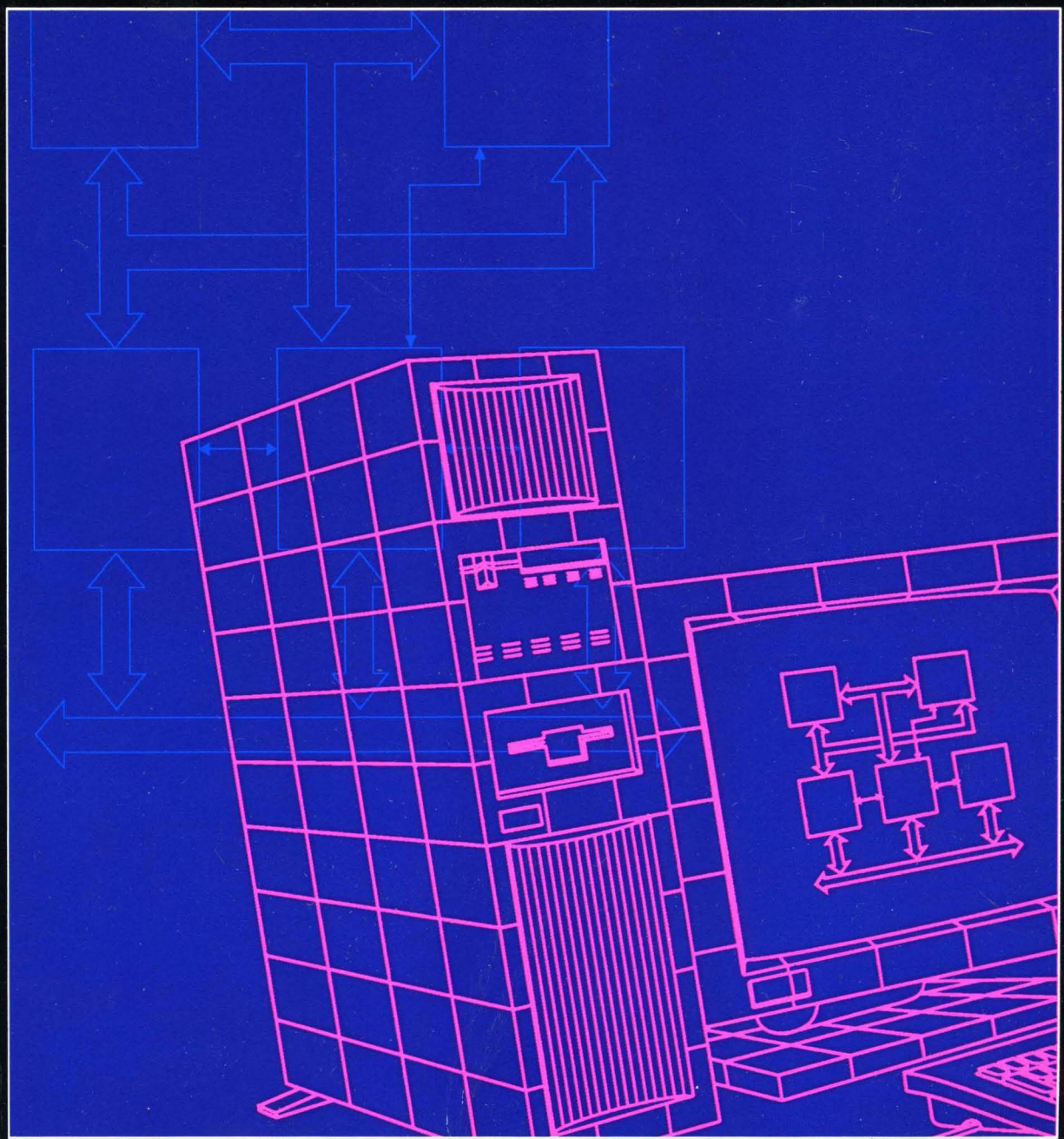
intel

Microprocessor and Peripheral Handbook

Volume I — Microprocessor



Order Number: 230843-006

LITERATURE

To order Intel Literature or obtain literature pricing information in the U.S. and Canada call or write Intel Literature Sales. In Europe and other international locations, please contact your local sales office or distributor.

INTEL LITERATURE SALES
P.O. BOX 58130
SANTA CLARA, CA 95052-8130

In the U.S. and Canada
call toll free
(800) 548-4725

CURRENT HANDBOOKS

Product line handbooks contain data sheets, application notes, article reprints and other design information.

TITLE	LITERATURE ORDER NUMBER
COMPLETE SET OF HANDBOOKS (Available in U.S. and Canada only)	**231003**
AUTOMOTIVE PRODUCTS HANDBOOK (Not included in handbook set)	231792
COMPONENTS QUALITY/RELIABILITY HANDBOOK	210997
EMBEDDED CONTROL APPLICATIONS HANDBOOK	270648
8-BIT EMBEDDED CONTROLLER HANDBOOK	270645
16-BIT EMBEDDED CONTROLLER HANDBOOK	270646
32-BIT EMBEDDED CONTROLLER HANDBOOK	270647
MEMORY COMPONENTS HANDBOOK	210830
MICROCOMMUNICATIONS HANDBOOK	231658
MICROCOMPUTER PROGRAMMABLE LOGIC HANDBOOK	296083
MICROPROCESSOR AND PERIPHERAL HANDBOOK (2 volume set)	230843
MILITARY PRODUCTS HANDBOOK (2 volume set. Not included in handbook set)	210461
OEM BOARDS AND SYSTEMS HANDBOOK	280407
PRODUCT GUIDE (Overview of Intel's complete product lines)	210846
SYSTEMS QUALITY/RELIABILITY HANDBOOK	231762
INTEL PACKAGING OUTLINES AND DIMENSIONS (Packaging types, number of leads, etc.)	231369
LITERATURE PRICE LIST (U.S. and Canada) (Comprehensive list of current Intel Literature)	210620
INTERNATIONAL LITERATURE GUIDE	E00029

CG/LIT/100188

About Our Cover:

The microprocessor has been the integral device in translating unlimited ideas into solid reality. Our microprocessors work in close conjunction with our family of peripherals to form complete microcomputer chip set solutions that will help you develop and build tomorrow's electronic systems.

intel

Intel the Microcomputer Company:

When Intel invented the microprocessor in 1971, it created the era of microcomputers. Whether used as microcontrollers in automobiles or microwave ovens, or as personal computers or supercomputers, Intel's microcomputers have always offered leading-edge technology. In the second half of the 1980s, Intel architectures have held at least a 75% market share of microprocessors at 16 bits and above. Intel continues to strive for the highest standards in memory, microcomputer components, modules, and systems to give its customers the best possible competitive advantages.

MICROPROCESSOR AND PERIPHERAL HANDBOOK

VOLUME I MICROPROCESSOR

1989

intel

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice.

Contact your local sales office to obtain the latest specifications before placing your order.

The following are trademarks of Intel Corporation and may only be used to identify Intel Products:

Above, BITBUS, COMMputer, CREDIT, Data Pipeline, ETOX, FASTPATH, Genius, i, ı, ICE, iCEL, iCS, iDBP, iDIS, I²ICE, iLBX, i_m, iMDDX, iMMX, Inboard, Insite, Intel, int$_e$l, Intel376, Intel386, Intel486, int$_e$lBOS, Intel Certified, Intelevision, int$_e$ligent Identifier, int$_e$ligent Programming, Intellec, Intellink, iOSP, iPDS, iPSC, iRMK, iRMX, iSBC, iSBX, iSDM, iSXM, KEPROM, Library Manager, MAPNET, MCS, Megachassis, MICROMAINFRAME, MULTIBUS, MULTICHANNEL, MULTIMODULE, ONCE, OpenNET, OTP, PC BUBBLE, Plug-A-Bubble, PROMPT, Promware, QUEST, QueX, Quick-Erase, Quick-Pulse Programming, Ripplemode, RMX/80, RUPI, Seamless, SLD, SugarCube, UPI, and VLSiCEL, and the combination of ICE, iCS, iRMX, iSBC, iSBX, iSXM, MCS, or UPI and a numerical suffix, 4-SITE, 376, 386, 486.

MDS is an ordering code only and is not used as a product name or trademark. MDS® is a registered trademark of Mohawk Data Sciences Corporation.

*MULTIBUS is a patented Intel bus.

Additional copies of this manual or other Intel literature may be obtained from:

Intel Corporation
Literature Sales
P.O. Box 58130
Santa Clara, CA 95052-8130

CUSTOMER SUPPORT

INTEL'S COMPLETE SUPPORT SOLUTION WORLDWIDE

Customer Support is Intel's complete support service that provides Intel customers with hardware support, software support, customer training, consulting services and network management services. For detailed information contact your local sales offices.

After a customer purchases any system hardware or software product, service and support become major factors in determining whether that product will continue to meet a customer's expectations. Such support requires an international support organization and a breadth of programs to meet a variety of customer needs. As you might expect, Intel's customer support is quite extensive. It can start with assistance during your development effort to network management. 100 Intel sales and service offices are located worldwide — in the U.S., Canada, Europe and the Far East. So wherever you're using Intel technology, our professional staff is within close reach.

HARDWARE SUPPORT SERVICES

Intel's hardware maintenance service, starting with complete on-site installation will boost your productivity from the start and keep you running at maximum efficiency. Support for system or board level products can be tailored to match your needs, from complete on-site repair and maintenance support economical carry-in or mail-in factory service.

Intel can provide support service for not only Intel systems and emulators, but also support for equipment in your development lab or provide service on your product to your end-user/customer.

SOFTWARE SUPPORT SERVICES

Software products are supported by our Technical Information Phone Service (TIPS) that has a special toll free number to provide you with direct, ready information on known, documented problems and deficiencies, as well as work-arounds, patches and other solutions.

Intel's software support consists of two levels of contracts. Standard support includes TIPS (Technical Information Phone Service), updates and subscription service (product-specific troubleshooting guides and; *COMMENTS Magazine*). Basic support consists of updates and the subscription service. Contracts are sold in environments which represent product groupings (e.g., iRMX® environment).

CONSULTING SERVICES

Intel provides field system engineering consulting services for any phase of your development or application effort. You can use our system engineers in a variety of ways ranging from assistance in using a new product, developing an application, personalizing training and customizing an Intel product to providing technical and management consulting. Systems Engineers are well versed in technical areas such as microcommunications, real-time applications, embedded microcontrollers, and network services. You know your application needs; we know our products. Working together we can help you get a successful product to market in the least possible time.

CUSTOMER TRAINING

Intel offers a wide range of instructional programs covering various aspects of system design and implementation. In just three to ten days a limited number of individuals learn more in a single workshop than in weeks of self-study. For optimum convenience, workshops are scheduled regularly at Training Centers worldwide or we can take our workshops to you for on-site instruction. Covering a wide variety of topics, Intel's major course categories include: architecture and assembly language, programming and operating systems, BITBUS™ and LAN applications.

NETWORK MANAGEMENT SERVICES

Today's networking products are powerful and extremely flexible. The return they can provide on your investment via increased productivity and reduced costs can be very substantial.

Intel offers complete network support, from definition of your network's physical and functional design, to implementation, installation and maintenance. Whether installing your first network or adding to an existing one, Intel's Networking Specialists can optimize network performance for you.

Table of Contents

Table of Contents (Continued)

Table of Contents (Continued)

Alphanumeric Index

Alphanumeric Index (Continued)

Any of the following products may appear in this publication. If so, it must be noted that such products have counterparts manufactured by Intel Puerto Rico, Inc., Intel Puerto Rico II, Inc., and/or Intel Singapore, Ltd. The product codes/part numbers of these counterpart products are listed below next to the corresponding Intel Corporation product codes/part numbers.

Intel Corporation Product Codes/ Part Numbers	Intel Puerto Rico, Inc. Intel Puerto Rico II, Inc. Product Codes/ Part Numbers	Intel Singapore, Ltd. Product Codes/ Part Numbers
376SKIT	p376SKIT	
903	p903	
904	p904	
913	p913	
914	p914	
923	p923	
924	p924	
952	p952	
953	p953	
954	p954	
ADAICE	pADAICE	
B386M1	pB386M1	
B386M2	pB386M2	
B386M4	pB386M4	
B386M8	pB386M8	
C044KIT	pC044KIT	
C252KIT	pC252KIT	
C28	pC28	
C32	pC32	
C452KIT	pC452KIT	
D86ASM	pD86ASM	
D86C86	pD86C86	
D86EDI	pD86EDI	
DCM9111	pDCM9111	
DOSNET	pDOSNET	
F1	pF1	
GUPILOGICIID	pGUPILOGICIID	
H4	pH4	
I044	pI044	
I252KIT	pI252KIT	
I452KIT	pI452KIT	
I86ASM	pI86ASM	
ICE386	pICE386	
III010	pIII010	
III086	pIII086	
III086	TIII086	
III111	pIII111	
III186	pIII186	
III186	TIII186	
III198	pIII198	
III212	pIII212	
III286	pIII286	
III286	TIII286	
III515	pIII515	
III520	TIII520	
III520	pIII520	
III531	pIII531	
III532	pIII532	
III533	pIII533	
III621	pIII621	
III707	pIII707	
III707	TIII707	
III815	pIII815	
INA961	pINA961	
IPAT86	pIPAT86	
KAS	pKAS	
KC	pKC	
KH	pKH	
KM1	pKM1	

Intel Corporation Product Codes/ Part Numbers	Intel Puerto Rico, Inc. Intel Puerto Rico II, Inc. Product Codes/ Part Numbers	Intel Singapore, Ltd. Product Codes/ Part Numbers
KM2	pKM2	
KM4	pKM4	
KM8	pKM8	
KNLAN	pKNLAN	
KT60	pKT60	
KW140	pKW140	
KW40	pKW40	
KW80	pKW80	
M1	pM1	
M2	pM2	
M4	pM4	
M8	pM8	
MDS610	pMDS610	
MDX3015	pMDX3015	
MDX3015	pMDX3015	
MDX3016	pMDX3016	
MDX3016	pMDX3016	
MDX457	pMDX457	
MDX457	pMDX457	
MDX458	pMDX458	
MDX458	pMDX458	
MSA96	pMSA96	
NLAN	pNLAN	
PCLINK		sPCLINK
PCX344A	pPCX344A	
R286ASM	pR286ASM	
R286EDI	pR286EDI	
R286PLM	pR286PLM	
R286SSC	pR286SSC	
R86FOR	pR86FOR	
RCB4410		sRCB4410
RCX920	pRCX920	
RMX286	pRMX286	
RMXNET	pRMXNET	
S301	pS301	
S386	pS386	
SBC010	pSBC010	
SBC012	pSBC012	sSBC012
SBC020	pSBC020	
SBC028	pSBC028	
SBC040	pSBC040	
SBC056	pSBC056	
SBC108	pSBC108	
SBC116	pSBC116	
SBC18603	pSBC18603	sSBC18603
SBC186410	pSBC186410	
SBC18651	pSBC18651	sSBC18651
SBC186530	pSBC186530	
SBC18678	pSBC18678	
SBC18848	pSBC18848	sSBC18848
SBC18856	pSBC18856	sSBC18856
SBC208	pSBC208	sSBC208
SBC214	pSBC214	
SBC215	pSBC215	
SBC220	pSBC220	sSBC220
SBC221	pSBC221	
SBC28610	pSBC28610	sSBC28610
SBC28612	pSBC28612	
SBC28614	pSBC28614	

Intel Corporation Product Codes/ Part Numbers	Intel Puerto Rico, Inc. Intel Puerto Rico II, Inc. Product Codes/ Part Numbers	Intel Singapore, Ltd. Product Codes/ Part Numbers
SBC28616	pSBC28616	
SBC300	pSBC300	
SBC301	pSBC301	
SBC302	pSBC302	
SBC304	pSBC304	
SBC307	pSBC307	
SBC314	pSBC314	
SBC322	pSBC322	
SBC324	pSBC324	
SBC337	pSBC337	
SBC341	pSBC341	
SBC386	pSBC386	sSBC386
SBC386116	pSBC386116	
SBC386120	pSBC386120	
SBC38621	pSBC38621	
SBC38622	pSBC38622	
SBC38624	pSBC38624	
SBC38628	pSBC38628	
SBC38631	pSBC38631	
SBC38632	pSBC38632	
SBC38634	pSBC38634	
SBC38638	pSBC38638	
SBC428	pSBC428	sSBC428
SBC464	pSBC464	
SBC517	pSBC517	
SBC519	pSBC519	sSBC519
SBC534	pSBC534	sSBC534
SBC548	pSBC548	
SBC550	TSBC550	
SBC550	pSBC550	
SBC550	pSBC550	
SBC552	pSBC552	
SBC556	pSBC556	sSBC556
SBC569	pSBC569	
SBC589	pSBC589	
SBC604	pSBC604	
SBC608	pSBC608	
SBC614	pSBC614	
SBC618	pSBC618	
SBC655	pSBC655	
SBC6611	pSBC6611	
SBC8010	pSBC8010	
SBC80204	pSBC80204	
SBC8024	pSBC8024	sSBC8024
SBC8030	pSBC8030	
SBC8605	pSBC8605	sSBC8605
SBC8612	pSBC8612	
SBC8614	pSBC8614	
SBC8630	pSBC8630	sSBC8630
SBC8635	pSBC8635	sSBC8635
SBC86C38		sSBC86C38
SBC8825	pSBC8825	sSBC8825
SBC8840	pSBC8840	
SBC8845	pSBC8845	sSBC8845
SBC905	pSBC905	
SBCLNK001	pSBCLNK001	

Intel Corporation Product Codes/ Part Numbers	Intel Puerto Rico, Inc. Intel Puerto Rico II, Inc. Product Codes/ Part Numbers	Intel Singapore, Ltd. Product Codes/ Part Numbers
SBCMEM310	pSBCMEM310	
SBCMEM312	pSBCMEM312	
SBCMEM320	pSBCMEM320	
SBCMEM340	pSBCMEM340	
SBE96	pSBE96	
SBX217	pSBX217	
SBX218	pSBX218	
SBX270	pSBX270	
SBX311	pSBX311	
SBX328	pSBX328	
SBX331	pSBX331	
SBX344	pSBX344	
SBX350	pSBX350	
SBX351	pSBX351	
SBX354	pSBX354	
SBX488	pSBX488	
SBX586		sSBX586
SCHEMAIIPLD	pSCHEMAIIPLD	
SCOM	pSCOM	
SDK51	pSDK51	
SDK85	pSDK85	
SDK86	pSDK86	
SXM217	pSXM217	
SXM28612	pSXM28612	
SXM386	pSXM386	
SXM544	pSXM544	
SXM552	pSXM552	
SXM951	pSXM951	
SXM955	pSXM955	
SYP120	pSYP120	
SYP301	pSYP301	
SYP302	pSYP302	
SYP31090	pSYP31090	
SYP311	pSYP311	
SYP3847	pSYP3847	
SYR286	pSYR286	
SYR86	pSYR86	
SYS120	pSYS120	
SYS310	pSYS310	
SYS311	pSYS311	
T60	pT60	
TA096	pTA096	
TA252	pTA252	
TA452	pTA452	
W140	pW140	
W280	pW280	
W40	pW40	
W80	pW80	
XNX286DOC	pXNX286DOC	
XNX286DOCB	pXNX286DOCB	
XNXIBASE	pXNXIBASE	
XNXIDB	pXNXIDB	
XNXIDESK	pXNXIDESK	
XNXIPLAN	pXNXIPLAN	
XNXIWORD	pXNXIWORD	

Overview

1

INTRODUCTION

Intel microprocessors and peripherals provide a complete solution in increasingly complex application environments. Quite often, a single peripheral device will replace anywhere from 20 to 100 TTL devices (and the associated design time that goes with them).

Built-in functions and standard Intel microprocessor/peripheral interface deliver very real *time* and *performance* advantages to the designer of microprocessor-based systems.

REDUCED TIME TO MARKET

When you can purchase an off-the-shelf solution that replaces a number of discrete devices, you're also replacing all the design, testing, and debug *time* that goes with them.

INCREASED RELIABILITY

At Intel, the rate of failure for devices is carefully tracked. Highest reliability is a tangible goal that translates to higher reliability for your product, reduced downtime, and reduced repair costs. And as more and more functions are intergrated on a single VLSI device, the resulting system requires less power, produces less heat, and requires fewer mechanical connections—again resulting in greater system reliability.

LOWER PRODUCTION COST

By minimizing design time, increasing reliability, and replacing numerous parts, microprocessor and peripheral solutions can contribute dramatically to lower product costs.

HIGHER SYSTEM PERFORMANCE

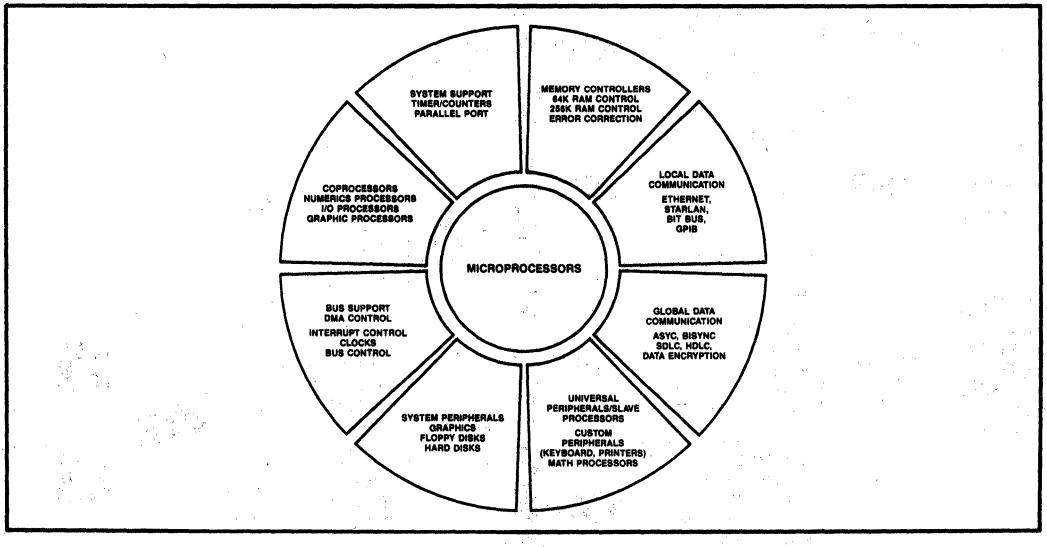
Intel microprocessors and peripherals provide the highest system performance for the demands of today's (and tomorrow's) microprocessor-based applications. For example, the 80386 32 bit offers the highest performance for multitasking, multiuser systems. Intel's peripheral products have been designed with the future in mind. They support all of Intel's 8, 16 and 32 bit processors.

HOW TO USE THE GUIDE

The following application guide illustrates the range of microprocessors and peripherals that can be used for the applictions in the vertical column of the left. The peripherals are grouped by the I/O function they control. CRT datacommunication, universal (user programmable), mass storage dynamic RAM controllers, and CPU/bus support.

An "X" in a horizontal application row indicates a potential peripheral or CPU, depending upon the features desired. For example, a conversational terminal could use either of the three display controllers, depending upon features like the number of characters per row or font capability. A "Y" indicates a likely candidate, for example, the 8272A Floppy Disk Controller in a small business computer.

The Intel microprocessor and peripherals family provides a broad range of time-saving, high performance solutions.



Get Your Kit Together!

Intel's Microsystem Components Kit Solution

MICRO-PROCESSOR
8088/80C88
8086/80C86
80186
80188
80286
386™μ
386SX™μ

SUPER CHIP SET
82310/11
82230/31
82335
82350

NUMERIC PROCESSORS
8087
80287
80387
80387SX

CPU SUPPORT
8231A
8253
8254/82C54
8255A/82C55A
8256AH
8279
82389
82370

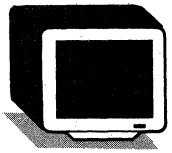
DMA
8253
8237
82285
82380
82560

CACHE MEMORY

CACHE CONTROL
82385

DYNAMIC RAM

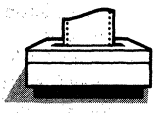
MEMORY SUPPORT
8203
8206
8207
82C08



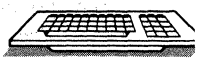
CRT CONTROL
82706
82716
82786

SPECIAL PERIPHERAL CONTROL
UPI™ 8041A/8741A
UPI™ 8042/8742
UPI™ 80/83/87C452

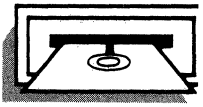
HARD COPY CONTROL
UPI™ 8042/8742



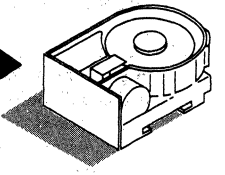
KEYBOARD CONTROL
8279-5
UPI™ 8042/8742



FLOPPY DISK CONTROL
8272A
82077



HARD DISK CONTROL
82064

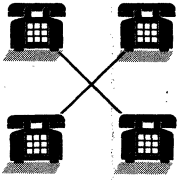


GLOBAL COMMUNICATIONS
8251A
82050
82510
8273
8274
8291A/92/94
82530
8044/8344/8744

LOCAL AREA NETWORKING
82C501
82586
82588
82560
82590/92
iMCC

INSTRUMENTATION BUS (GPIB)
8291
8292

TELE-COMMUNICATIONS
2910/11/12
29C13/C14/C16/C17
29C48
29C53AA
ISP 188
89024
89C024XE



January 1
Order Number: 230664-

8086 Microprocessor Family

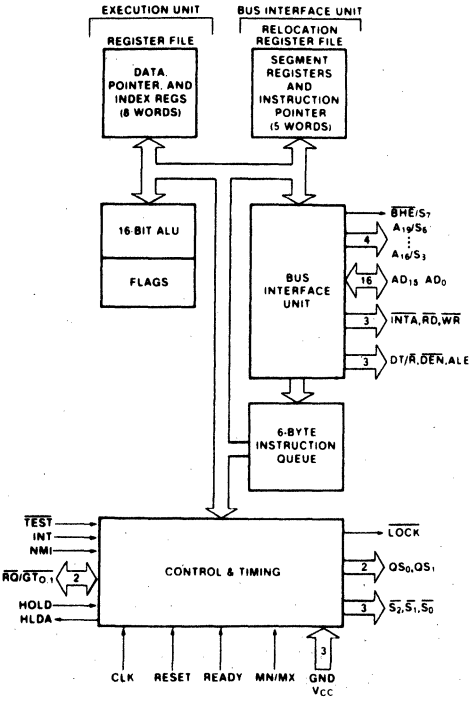
2

8086
16-BIT HMOS MICROPROCESSOR
8086/8086-2/8086-1*

- **Direct Addressing Capability 1 MByte of Memory**
- **Architecture Designed for Powerful Assembly Language and Efficient High Level Languages**
- **14 Word, by 16-Bit Register Set with Symmetrical Operations**
- **24 Operand Addressing Modes**
- **Bit, Byte, Word, and Block Operations**
- **8 and 16-Bit Signed and Unsigned Arithmetic in Binary or Decimal Including Multiply and Divide**
- **Range of Clock Rates: 5 MHz for 8086, 8 MHz for 8086-2, 10 MHz for 8086-1**
- **MULTIBUS® System Compatible Interface**
- **Available in EXPRESS — Standard Temperature Range — Extended Temperature Range**
- **Available in 40-Lead Cerdip and Plastic Package**
  (See Packaging Spec. Order #231369)

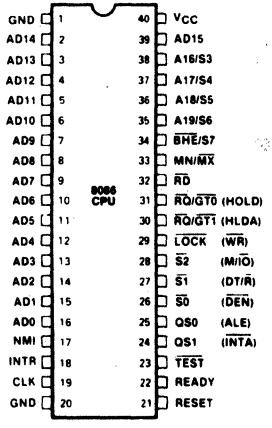
The Intel 8086 high performance 16-bit CPU is available in three clock rates: 5, 8 and 10 MHz. The CPU is implemented in N-Channel, depletion load, silicon gate technology (HMOS), and packaged in a 40-pin CERDIP or plastic package. The 8086 operates in both single processor and multiple processor configurations to achieve high performance levels.

*Changes from the 1985 handbook specification have been made for the 8086-1. See A.C. Characteristics TGVCH and TCLGL.



231455-1

Figure 1. 8086 CPU Block Diagram



231455-2

40 Lead

Figure 2. 8086 Pin Configuration

September 1988
Order Number: 231455-003

Table 1. Pin Description

The following pin function descriptions are for 8086 systems in either minimum or maximum mode. The "Local Bus" in these descriptions is the direct multiplexed bus interface connection to the 8086 (without regard to additional bus buffers).

Symbol	Pin No.	Type	Name and Function
AD_{15}–AD_0	2–16, 39	I/O	**ADDRESS DATA BUS:** These lines constitute the time multiplexed memory/IO address (T_1), and data (T_2, T_3, T_W, T_4) bus. A_0 is analogous to $\overline{BHE}$ for the lower byte of the data bus, pins D_7–D_0. It is LOW during T_1 when a byte is to be transferred on the lower portion of the bus in memory or I/O operations. Eight-bit oriented devices tied to the lower half would normally use A_0 to condition chip select functions. (See $\overline{BHE}$.) These lines are active HIGH and float to 3-state OFF during interrupt acknowledge and local bus "hold acknowledge".
A_{19}/S_6, A_{18}/S_5, A_{17}/S_4, A_{16}/S_3	35–38	O	**ADDRESS/STATUS:** During T_1 these are the four most significant address lines for memory operations. During I/O operations these lines are LOW. During memory and I/O operations, status information is available on these lines during T_2, T_3, T_W, T_4. The status of the interrupt enable FLAG bit (S_5) is updated at the beginning of each CLK cycle. A_{17}/S_4 and A_{16}/S_3 are encoded as shown. This information indicates which relocation register is presently being used for data accessing. These lines float to 3-state OFF during local bus "hold acknowledge."

A_{17}/S_4	A_{16}/S_3	Characteristics
0 (LOW)	0	Alternate Data
0	1	Stack
1 (HIGH)	0	Code or None
1	1	Data
S_6 is 0 (LOW)		

$\overline{BHE}/S_7$	34	O	**BUS HIGH ENABLE/STATUS:** During T_1 the bus high enable signal ($\overline{BHE}$) should be used to enable data onto the most significant half of the data bus, pins D_{15}–D_8. Eight-bit oriented devices tied to the upper half of the bus would normally use $\overline{BHE}$ to condition chip select functions. $\overline{BHE}$ is LOW during T_1 for read, write, and interrupt acknowledge cycles when a byte is to be transferred on the high portion of the bus. The S_7 status information is available during T_2, T_3, and T_4. The signal is active LOW, and floats to 3-state OFF in "hold". It is LOW during T_1 for the first interrupt acknowledge cycle.

$\overline{BHE}$	A_0	Characteristics
0	0	Whole word
0	1	Upper byte from/to odd address
1	0	Lower byte from/to even address
1	1	None

$\overline{RD}$	32	O	**READ:** Read strobe indicates that the processor is performing a memory of I/O read cycle, depending on the state of the S_2 pin. This signal is used to read devices which reside on the 8086 local bus. $\overline{RD}$ is active LOW during T_2, T_3 and T_W of any read cycle, and is guaranteed to remain HIGH in T_2 until the 8086 local bus has floated. This signal floats to 3-state OFF in "hold acknowledge".

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
READY	22	I	**READY:** is the acknowledgement from the addressed memory or I/O device that it will complete the data transfer. The READY signal from memory/IO is synchronized by the 8284A Clock Generator to form READY. This signal is active HIGH. The 8086 READY input is not synchronized. Correct operation is not guaranteed if the setup and hold times are not met.
INTR	18	I	**INTERRUPT REQUEST:** is a level triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored to via an interrupt vector lookup table located in system memory. It can be internally masked by software resetting the interrupt enable bit. INTR is internally synchronized. This signal is active HIGH.
$\overline{TEST}$	23	I	**$\overline{TEST}$:** input is examined by the "Wait" instruction. If the $\overline{TEST}$ input is LOW execution continues, otherwise the processor waits in an "Idle" state. This input is synchronized internally during each clock cycle on the leading edge of CLK.
NMI	17	I	**NON-MASKABLE INTERRUPT:** an edge triggered input which causes a type 2 interrupt. A subroutine is vectored to via an interrupt vector lookup table located in system memory. NMI is not maskable internally by software. A transition from LOW to HIGH initiates the interrupt at the end of the current instruction. This input is internally synchronized.
RESET	21	I	**RESET:** causes the processor to immediately terminate its present activity. The signal must be active HIGH for at least four clock cycles. It restarts execution, as described in the Instruction Set description, when RESET returns LOW. RESET is internally synchronized.
CLK	19	I	**CLOCK:** provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing.
V_{CC}	40		**V_{CC}:** +5V power supply pin.
GND	1, 20		**GROUND**
MN/$\overline{MX}$	33	I	**MINIMUM/MAXIMUM:** indicates what mode the processor is to operate in. The two modes are discussed in the following sections.

The following pin function descriptions are for the 8086/8288 system in maximum mode (i.e., MN/$\overline{MX}$ = V_{SS}). Only the pin functions which are unique to maximum mode are described; all other pin functions are as described above.

$\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$	26–28	O	**STATUS:** active during T_4, T_1, and T_2 and is returned to the passive state (1, 1, 1) during T_3 or during T_W when READY is HIGH. This status is used by the 8288 Bus Controller to generate all memory and I/O access control signals. Any change by $\overline{S_2}$, $\overline{S_1}$, or $\overline{S_0}$ during T_4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T_3 or T_W is used to indicate the end of a bus cycle.

Table 1. Pin Description (Continued)

<table>
<tr><th>Symbol</th><th>Pin No.</th><th>Type</th><th>Name and Function</th></tr>
<tr><td>$\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$ (Continued)</td><td>26–28</td><td>O</td><td>These signals float to 3-state OFF in "hold acknowledge". These status lines are encoded as shown.
<table>
<tr><th>$\overline{S_2}$</th><th>$\overline{S_1}$</th><th>$\overline{S_0}$</th><th>Characteristics</th></tr>
<tr><td>0 (LOW)</td><td>0</td><td>0</td><td>Interrupt Acknowledge</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Read I/O Port</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Write I/O Port</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Halt</td></tr>
<tr><td>1 (HIGH)</td><td>0</td><td>0</td><td>Code Access</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Read Memory</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Write Memory</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Passive</td></tr>
</table></td></tr>
<tr><td>$\overline{RQ}/\overline{GT_0}$, $\overline{RQ}/\overline{GT_1}$</td><td>30, 31</td><td>I/O</td><td>REQUEST/GRANT: pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional with $\overline{RQ}/\overline{GT_0}$ having higher priority than $\overline{RQ}/\overline{GT_1}$. $\overline{RQ}/\overline{GT}$ pins have internal pull-up resistors and may be left unconnected. The request/grant sequence is as follows (see Figure 9):
1. A pulse of 1 CLK wide from another local bus master indicates a local bus request ("hold") to the 8086 (pulse 1).
2. During a T_4 or T_1 clock cycle, a pulse 1 CLK wide from the 8086 to the requesting master (pulse 2), indicates that the 8086 has allowed the local bus to float and that it will enter the "hold acknowledge" state at the next CLK. The CPU's bus interface unit is disconnected logically from the local bus during "hold acknowledge".
3. A pulse 1 CLK wide from the requesting master indicates to the 8086 (pulse 3) that the "hold" request is about to end and that the 8086 can reclaim the local bus at the next CLK.
Each master-master exchange of the local bus is a sequence of 3 pulses. There must be one dead CLK cycle after each bus exchange. Pulses are active LOW.
If the request is made while the CPU is performing a memory cycle, it will release the local bus during T_4 of the cycle when all the following conditions are met:
1. Request occurs on or before T_2.
2. Current cycle is not the low byte of a word (on an odd address).
3. Current cycle is not the first acknowledge of an interrupt acknowledge sequence.
4. A locked instruction is not currently executing.
If the local bus is idle when the request is made the two possible events will follow:
1. Local bus will be released during the next clock.
2. A memory cycle will start within 3 clocks. Now the four rules for a currently active memory cycle apply with condition number 1 already satisfied.</td></tr>
<tr><td>$\overline{LOCK}$</td><td>29</td><td>O</td><td>$\overline{LOCK}$: output indicates that other system bus masters are not to gain control of the system bus while $\overline{LOCK}$ is active LOW. The $\overline{LOCK}$ signal is activated by the "LOCK" prefix instruction and remains active until the completion of the next instruction. This signal is active LOW, and floats to 3-state OFF in "hold acknowledge".</td></tr>
</table>

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
QS_1, QS_0	24, 25	O	**QUEUE STATUS:** The queue status is valid during the CLK cycle after which the queue operation is performed. QS_1 and QS_0 provide status to allow external tracking of the internal 8086 instruction queue.

QS_1	QS_0	Characteristics
0 (LOW)	0	No Operation
0	1	First Byte of Op Code from Queue
1 (HIGH)	0	Empty the Queue
1	1	Subsequent Byte from Queue

The following pin function descriptions are for the 8086 in minimum mode (i.e., MN/$\overline{MX}$ = V_{CC}). Only the pin functions which are unique to minimum mode are described; all other pin functions are as described above.

M/$\overline{IO}$	28	O	**STATUS LINE:** logically equivalent to S_2 in the maximum mode. It is used to distinguish a memory access from an I/O access. M/$\overline{IO}$ becomes valid in the T_4 preceding a bus cycle and remains valid until the final T_4 of the cycle (M = HIGH, IO = LOW). M/$\overline{IO}$ floats to 3-state OFF in local bus "hold acknowledge".
$\overline{WR}$	29	O	**WRITE:** indicates that the processor is performing a write memory or write I/O cycle, depending on the state of the M/$\overline{IO}$ signal. $\overline{WR}$ is active for T_2, T_3 and T_W of any write cycle. It is active LOW, and floats to 3-state OFF in local bus "hold acknowledge".
$\overline{INTA}$	24	O	**$\overline{INTA}$:** is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T_2, T_3 and T_W of each interrupt acknowledge cycle.
ALE	25	O	**ADDRESS LATCH ENABLE:** provided by the processor to latch the address into the 8282/8283 address latch. It is a HIGH pulse active during T_1 of any bus cycle. Note that ALE is never floated.
DT/$\overline{R}$	27	O	**DATA TRANSMIT/RECEIVE:** needed in minimum system that desires to use an 8286/8287 data bus transceiver. It is used to control the direction of data flow through the transceiver. Logically DT/$\overline{R}$ is equivalent to $\overline{S_1}$ in the maximum mode, and its timing is the same as for M/$\overline{IO}$. (T = HIGH, R = LOW.) This signal floats to 3-state OFF in local bus "hold acknowledge".
$\overline{DEN}$	26	O	**DATA ENABLE:** provided as an output enable for the 8286/8287 in a minimum system which uses the transceiver. $\overline{DEN}$ is active LOW during each memory and I/O access and for INTA cycles. For a read or $\overline{INTA}$ cycle it is active from the middle of T_2 until the middle of T_4, while for a write cycle it is active from the beginning of T_2 until the middle of T_4. $\overline{DEN}$ floats to 3-state OFF in local bus "hold acknowledge".
HOLD, HLDA	31, 30	I/O	**HOLD:** indicates that another master is requesting a local bus "hold." To be acknowledged, HOLD must be active HIGH. The processor receiving the "hold" request will issue HLDA (HIGH) as an acknowledgement in the middle of a T_4 or T_i clock cycle. Simultaneous with the issuance of HLDA the processor will float the local bus and control lines. After HOLD is detected as being LOW, the processor will LOWer the HLDA, and when the processor needs to run another cycle, it will again drive the local bus and control lines. The same rules as for $\overline{RQ}$/$\overline{GT}$ apply regarding when the local bus will be released. HOLD is not asynchronous input. External synchronization should be provided if the system cannot otherwise guarantee the setup time.

FUNCTIONAL DESCRIPTION

General Operation

The internal functions of the 8086 processor are partitioned logically into two processing units. The first is the Bus Interface Unit (BIU) and the second is the Execution Unit (EU) as shown in the block diagram of Figure 1.

These units can interact directly but for the most part perform as separate asynchronous operational processors. The bus interface unit provides the functions related to instruction fetching and queuing, operand fetch and store, and address relocation. This unit also provides the basic bus control. The overlap of instruction pre-fetching provided by this unit serves to increase processor performance through improved bus bandwidth utilization. Up to 6 bytes of the instruction stream can be queued while waiting for decoding and execution.

The instruction stream queuing mechanism allows the BIU to keep the memory utilized very efficiently. Whenever there is space for at least 2 bytes in the queue, the BIU will attempt a word fetch memory cycle. This greatly reduces "dead time" on the memory bus. The queue acts as a First-In-First-Out (FIFO) buffer, from which the EU extracts instruction bytes as required. If the queue is empty (following a branch instruction, for example), the first byte into the queue immediately becomes available to the EU.

The execution unit receives pre-fetched instructions from the BIU queue and provides un-relocated operand addresses to the BIU. Memory operands are passed through the BIU for processing by the EU, which passes results to the BIU for storage. See the Instruction Set description for further register set and architectural descriptions.

MEMORY ORGANIZATION

The processor provides a 20-bit address to memory which locates the byte being referenced. The memory is organized as a linear array of up to 1 million bytes, addressed as 00000(H) to FFFFF(H). The memory is logically divided into code, data, extra data, and stack segments of up to 64K bytes each, with each segment falling on 16-byte boundaries. (See Figure 3a.)

All memory references are made relative to base addresses contained in high speed segment registers. The segment types were chosen based on the addressing needs of programs. The segment register to be selected is automatically chosen according to the rules of the following table. All information in one segment type share the same logical attributes (e.g. code or data). By structuring memory into relocatable areas of similar characteristics and by automatically selecting segment registers, programs are shorter, faster, and more structured.

Word (16-bit) operands can be located on even or odd address boundaries and are thus not constrained to even boundaries as is the case in many 16-bit computers. For address and data operands, the least significant byte of the word is stored in the lower valued address location and the most significant byte in the next higher address location. The BIU automatically performs the proper number of memory accesses, one if the word operand is on an even byte boundary and two if it is on an odd byte boundary. Except for the performance penalty, this double access is transparent to the software. This performance penalty does not occur for instruction fetches, only word operands.

Physically, the memory is organized as a high bank (D_{15}–D_8) and a low bank (D_7–D_0) of 512K 8-bit bytes addressed in parallel by the processor's address lines A_{19}–A_1. Byte data with even addresses is transferred on the D_7–D_0 bus lines while odd addressed byte data (A_0 HIGH) is transferred on the D_{15}–D_8 bus lines. The processor provides two enable signals, $\overline{BHE}$ and A_0, to selectively allow reading from or writing into either an odd byte location, even byte location, or both. The instruction stream is fetched from memory as words and is addressed internally by the processor to the byte level as necessary.

Memory Reference Need	Segment Register Used	Segment Selection Rule
Instructions	CODE (CS)	Automatic with all instruction prefetch.
Stack	STACK (SS)	All stack pushes and pops. Memory references relative to BP base register except data references.
Local Data	DATA (DS)	Data references when: relative to stack, destination of string operation, or explicitly overridden.
External (Global) Data	EXTRA (ES)	Destination of string operations: explicitly selected using a segment override.

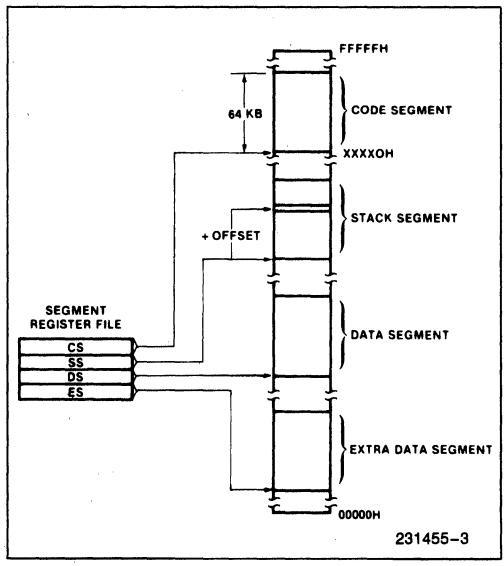


Figure 3a. Memory Organization

In referencing word data the BIU requires one or two memory cycles depending on whether or not the starting byte of the word is on an even or odd address, respectively. Consequently, in referencing word operands performance can be optimized by locating data on even address boundaries. This is an especially useful technique for using the stack, since odd address references to the stack may adversely affect the context switching time for interrupt processing or task multiplexing.

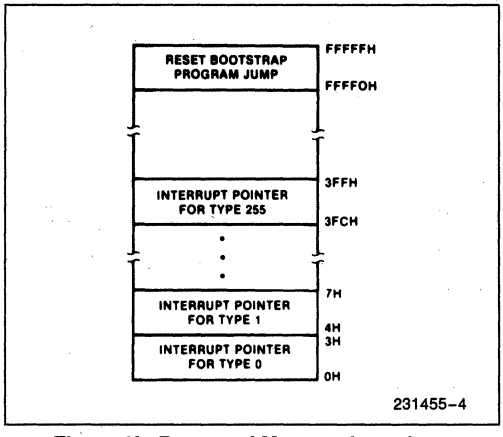


Figure 3b. Reserved Memory Locations

Certain locations in memory are reserved for specific CPU operations (see Figure 3b). Locations from address FFFF0H through FFFFFH are reserved for operations including a jump to the initial program loading routine. Following RESET, the CPU will always begin execution at location FFFF0H where the jump must be. Locations 00000H through 003FFH are reserved for interrupt operations. Each of the 256 possible interrupt types has its service routine pointed to by a 4-byte pointer element consisting of a 16-bit segment address and a 16-bit offset address. The pointer elements are assumed to have been stored at the respective places in reserved memory prior to occurrence of interrupts.

MINIMUM AND MAXIMUM MODES

The requirements for supporting minimum and maximum 8086 systems are sufficiently different that they cannot be done efficiently with 40 uniquely defined pins. Consequently, the 8086 is equipped with a strap pin (MN/$\overline{MX}$) which defines the system configuration. The definition of a certain subset of the pins changes dependent on the condition of the strap pin. When MN/$\overline{MX}$ pin is strapped to GND, the 8086 treats pins 24 through 31 in maximum mode. An 8288 bus controller interprets status information coded into $\overline{S_0}$, $\overline{S_2}$, $\overline{S_2}$ to generate bus timing and control signals compatible with the MULTIBUS® architecture. When the MN/$\overline{MX}$ pin is strapped to V_{CC}, the 8086 generates bus control signals itself on pins 24 through 31, as shown in parentheses in Figure 2. Examples of minimum mode and maximum mode systems are shown in Figure 4.

BUS OPERATION

The 8086 has a combined address and data bus commonly referred to as a time multiplexed bus. This technique provides the most efficient use of pins on the processor while permitting the use of a standard 40-lead package. This "local bus" can be buffered directly and used throughout the system with address latching provided on memory and I/O modules. In addition, the bus can also be demultiplexed at the processor with a single set of address latches if a standard non-multiplexed bus is desired for the system.

Each processor bus cycle consists of at least four CLK cycles. These are referred to as T_1, T_2, T_3 and T_4 (see Figure 5). The address is emitted from the processor during T_1 and data transfer occurs on the bus during T_3 and T_4. T_2 is used primarily for changing the direction of the bus during read operations. In the event that a "NOT READY" indication is given by the addressed device, "Wait" states (T_W) are inserted between T_3 and T_4. Each inserted "Wait" state is of the same duration as a CLK cycle. Periods

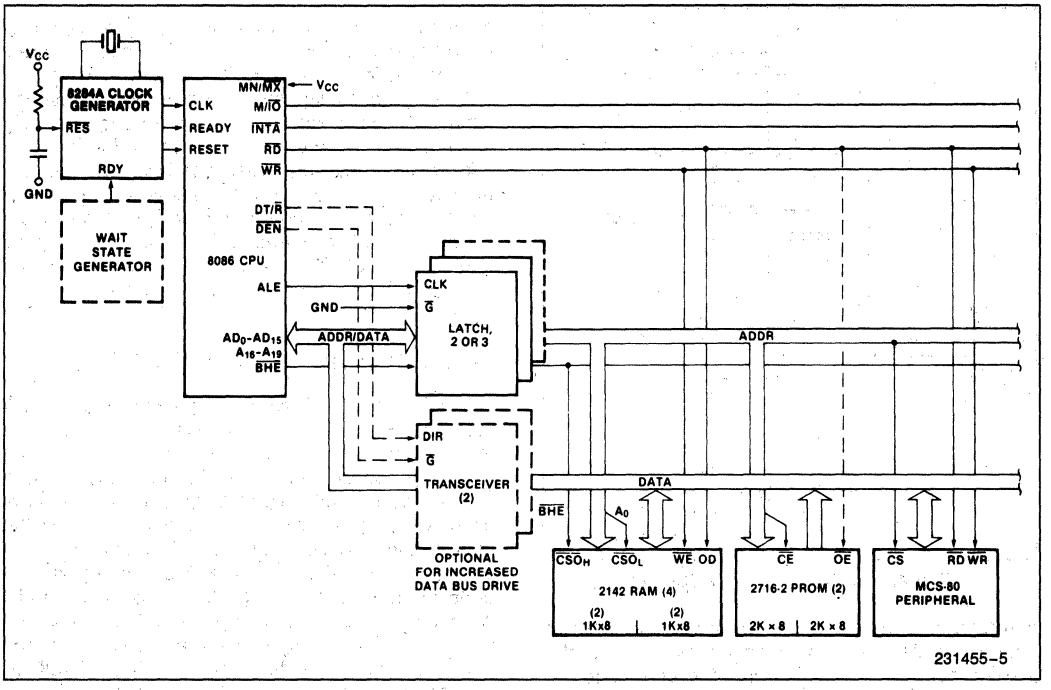


Figure 4a. Minimum Mode 8086 Typical Configuration

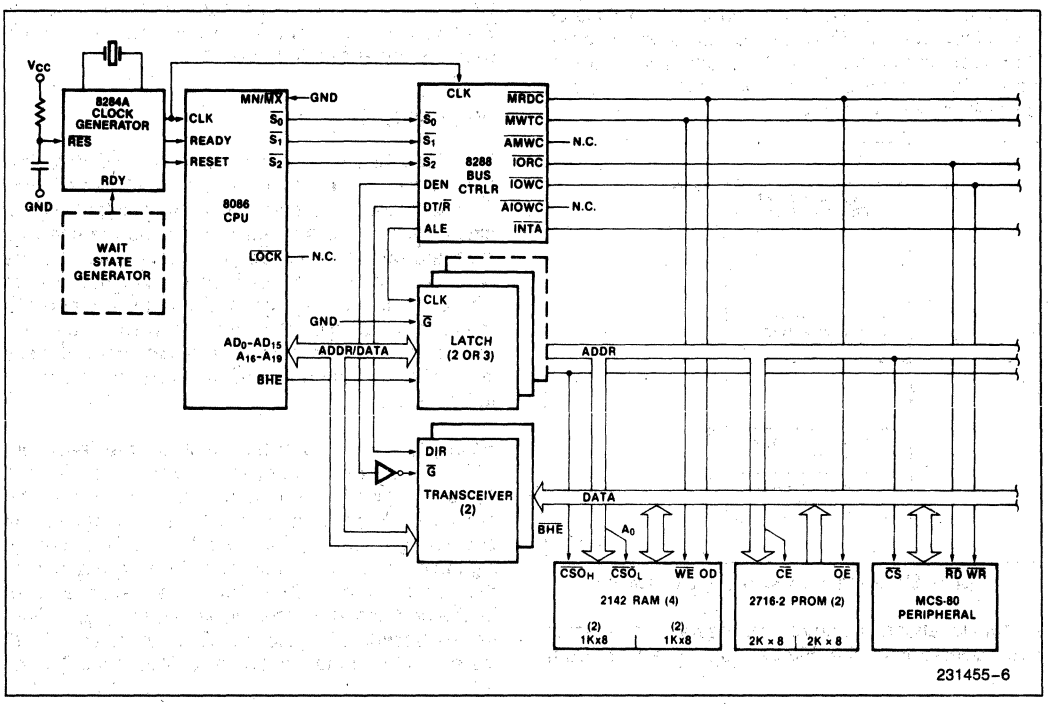


Figure 4b. Maximum Mode 8086 Typical Configuration

can occur between 8086 bus cycles. These are referred to as "Idle" states (T_I) or inactive CLK cycles. The processor uses these cycles for internal housekeeping.

During T_1 of any bus cycle the ALE (Address Latch Enable) signal is emitted (by either the processor or the 8288 bus controller, depending on the MN/$\overline{MX}$ strap). At the trailing edge of this pulse, a valid address and certain status information for the cycle may be latched.

Status bits $\overline{S_0}$, $\overline{S_1}$, and $\overline{S_2}$ are used, in maximum mode, by the bus controller to identify the type of bus transaction according to the following table:

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Characteristics
0 (LOW)	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1 (HIGH)	0	0	Instruction Fetch
1	0	1	Read Data from Memory
1	1	0	Write Data to Memory
1	1	1	Passive (no bus cycle)

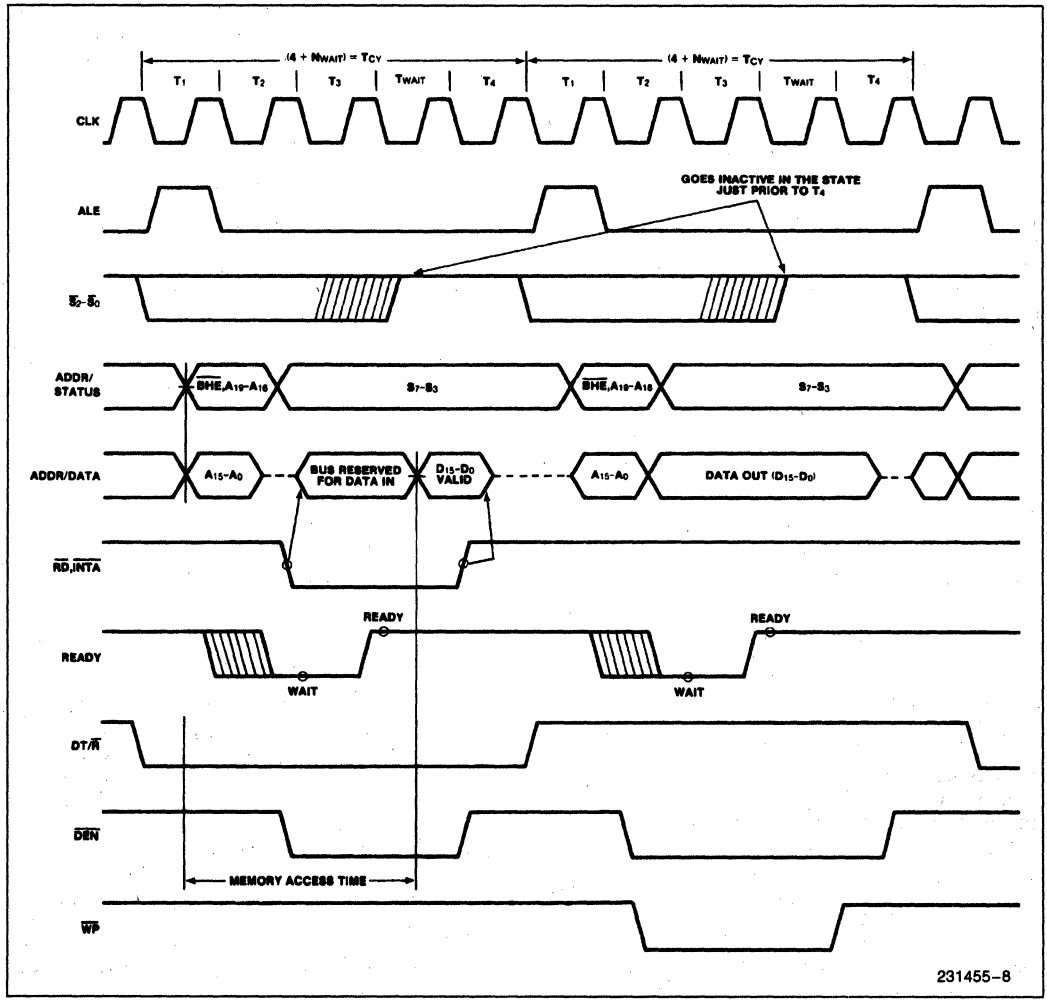


Figure 5. Basic System Timing

Status bits S_3 through S_7 are multiplexed with high-order address bits and the $\overline{BHE}$ signal, and are therefore valid during T_2 through T_4. S_3 and S_4 indicate which segment register (see Instruction Set description) was used for this bus cycle in forming the address, according to the following table:

S_4	S_3	Characteristics
0 (LOW)	0	Alternate Data (extra segment)
0	1	Stack
1 (HIGH)	0	Code or None
1	1	Data

S_5 is a reflection of the PSW interrupt enable bit. S_6 = 0 and S_7 is a spare status bit.

I/O ADDRESSING

In the 8086, I/O operations can address up to a maximum of 64K I/O byte registers or 32K I/O word registers. The I/O address appears in the same format as the memory address on bus lines A_{15}–A_0. The address lines A_{19}–A_{16} are zero in I/O operations. The variable I/O instructions which use register DX as a pointer have full address capability while the direct I/O instructions directly address one or two of the 256 I/O byte locations in page 0 of the I/O address space.

I/O ports are addressed in the same manner as memory locations. Even addressed bytes are transferred on the D_7–D_0 bus lines and odd addressed bytes on D_{15}–D_8. Care must be taken to assure that each register within an 8-bit peripheral located on the lower portion of the bus be addressed as even.

External Interface

PROCESSOR RESET AND INITIALIZATION

Processor initialization or start up is accomplished with activation (HIGH) of the RESET pin. The 8086 RESET is required to be HIGH for greater than 4 CLK cycles. The 8086 will terminate operations on the high-going edge of RESET and will remain dormant as long as RESET is HIGH. The low-going transition of RESET triggers an internal reset sequence for approximately 10 CLK cycles. After this interval the 8086 operates normally beginning with the instruction in absolute location FFFF0H (see Figure 3b). The details of this operation are specified in the Instruction Set description of the MCS-86 Family User's Manual. The RESET input is internally synchronized to the processor clock. At initialization the HIGH-to-LOW transition of RESET must occur no sooner than 50 μs after power-up, to allow complete initialization of the 8086.

NMI asserted prior to the 2nd clock after the end of RESET will not be honored. If NMI is asserted after that point and during the internal reset sequence, the processor may execute one instruction before responding to the interrupt. A hold request active immediately after RESET will be honored before the first instruction fetch.

All 3-state outputs float to 3-state OFF during RESET. Status is active in the idle state for the first clock after RESET becomes active and then floats to 3-state OFF. ALE and HLDA are driven low.

INTERRUPT OPERATIONS

Interrupt operations fall into two classes; software or hardware initiated. The software initiated interrupts and software aspects of hardware interrupts are specified in the Instruction Set description. Hardware interrupts can be classified as non-maskable or maskable.

Interrupts result in a transfer of control to a new program location. A 256-element table containing address pointers to the interrupt service program locations resides in absolute locations 0 through 3FFH (see Figure 3b), which are reserved for this purpose. Each element in the table is 4 bytes in size and corresponds to an interrupt "type". An interrupting device supplies an 8-bit type number, during the interrupt acknowledge sequence, which is used to "vector" through the appropriate element to the new interrupt service program location.

NON-MASKABLE INTERRUPT (NMI)

The processor provides a single non-maskable interrupt pin (NMI) which has higher priority than the maskable interrupt request pin (INTR). A typical use would be to activate a power failure routine. The NMI is edge-triggered on a LOW-to-HIGH transition. The activation of this pin causes a type 2 interrupt. (See Instruction Set description.)

NMI is required to have a duration in the HIGH state of greater than two CLK cycles, but is not required to be synchronized to the clock. Any high-going transition of NMI is latched on-chip and will be serviced at the end of the current instruction or between whole moves of a block-type instruction. Worst case response to NMI would be for multiply, divide, and variable shift instructions. There is no specification on the occurrence of the low-going edge; it may occur before, during, or after the servicing of NMI. Another high-going edge triggers another response if it occurs after the start of the NMI procedure. The signal must be free of logical spikes in general and be free of bounces on the low-going edge to avoid triggering extraneous responses.

MASKABLE NTERRUPT (INTR)

The 8086 provides a single interrupt request input (INTR) which can be masked internally by software with the resetting of the interrupt enable FLAG status bit. The interrupt request signal is level triggered. It is internally synchronized during each clock cycle on the high-going edge of CLK. To be responded to, INTR must be present (HIGH) during the clock period preceding the end of the current instruction or the end of a whole move for a block-type instruction. During the interrupt response sequence further interrupts are disabled. The enable bit is reset as part of the response to any interrupt (INTR, NMI, software interrupt or single-step), although the FLAGS register which is automatically pushed onto the stack reflects the state of the processor prior to the interrupt. Until the old FLAGS register is restored the enable bit will be zero unless specifically set by an instruction.

During the response sequence (Figure 6) the processor executes two successive (back-to-back) interrupt acknowledge cycles. The 8086 emits the LOCK signal from T_2 of the first bus cycle until T_2 of the second. A local bus "hold" request will not be honored until the end of the second bus cycle. In the second bus cycle a byte is fetched from the external interrupt system (e.g., 8259A PIC) which identifies the source (type) of the interrupt. This byte is multiplied by four and used as a pointer into the interrupt vector lookup table. An INTR signal left HIGH will be continually responded to within the limitations of the enable bit and sample period. The INTERRUPT RETURN instruction includes a FLAGS pop which returns the status of the original interrupt enable bit when it restores the FLAGS.

HALT

When a software "HALT" instruction is executed the processor indicates that it is entering the "HALT" state in one of two ways depending upon which mode is strapped. In minimum mode, the processor issues one ALE with no qualifying bus control signals. In maximum mode, the processor issues appropriate HALT status on $\overline{S}_2$, $\overline{S}_1$, and $\overline{S}_0$; and the 8288 bus controller issues one ALE. The 8086 will not leave the "HALT" state when a local bus "hold" is entered while in "HALT". In this case, the processor reissues the HALT indicator. An interrupt request or RESET will force the 8086 out of the "HALT" state.

READ/MODIFY/WRITE (SEMAPHORE) OPERATIONS VIA LOCK

The $\overline{LOCK}$ status information is provided by the processor when directly consecutive bus cycles are required during the execution of an instruction. This provides the processor with the capability of performing read/modify/write operations on memory (via the Exchange Register With Memory instruction, for example) without the possibility of another system bus master receiving intervening memory cycles. This is useful in multi-processor system configurations to accomplish "test and set lock" operations. The $\overline{LOCK}$ signal is activated (forced LOW) in the clock cycle following the one in which the software "LOCK" prefix instruction is decoded by the EU. It is deactivated at the end of the last bus cycle of the instruction following the "LOCK" prefix instruction. While $\overline{LOCK}$ is active a request on a RQ/GT pin will be recorded and then honored at the end of the LOCK.

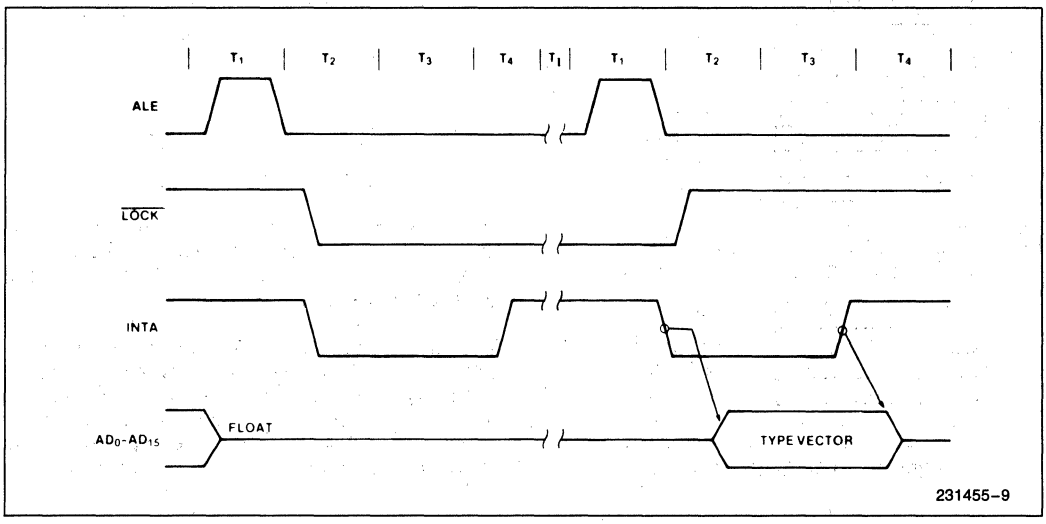


Figure 6. Interrupt Acknowledge Sequence

EXTERNAL SYNCHRONIZATION VIA TEST

As an alternative to the interrupts and general I/O capabilities, the 8086 provides a single software-testable input known as the $\overline{TEST}$ signal. At any time the program may execute a WAIT instruction. If at that time the $\overline{TEST}$ signal is inactive (HIGH), program execution becomes suspended while the processor waits for $\overline{TEST}$ to become active. It must remain active for at least 5 CLK cycles. The WAIT instruction is re-executed repeatedly until that time. This activity does not consume bus cycles. The processor remains in an idle state while waiting. All 8086 drivers go to 3-state OFF if bus "Hold" is entered. If interrupts are enabled, they may occur while the processor is waiting. When this occurs the processor fetches the WAIT instruction one extra time, processes the interrupt, and then re-fetches and re-executes the WAIT instruction upon returning from the interrupt.

Basic System Timing

Typical system configurations for the processor operating in minimum mode and in maximum mode are shown in Figures 4a and 4b, respectively. In minimum mode, the MN/$\overline{MX}$ pin is strapped to V_{CC} and the processor emits bus control signals in a manner similar to the 8085. In maximum mode, the MN/$\overline{MX}$ pin is strapped to V_{SS} and the processor emits coded status information which the 8288 bus controller uses to generate MULTIBUS compatible bus control signals. Figure 5 illustrates the signal timing relationships.

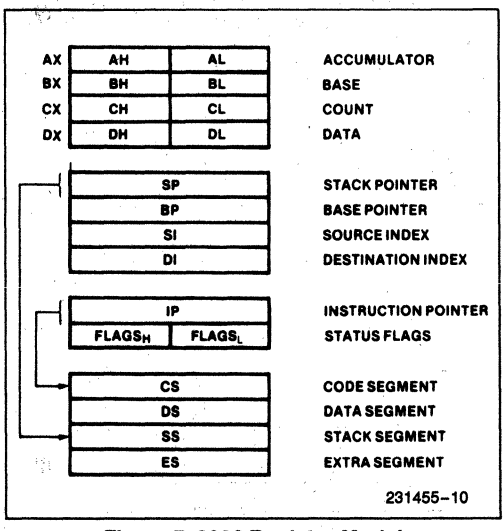


Figure 7. 8086 Register Model

SYSTEM TIMING—MINIMUM SYSTEM

The read cycle begins in T_1 with the assertion of the Address Latch Enable (ALE) signal. The trailing (low-going) edge of this signal is used to latch the address information, which is valid on the local bus at this time, into the address latch. The $\overline{BHE}$ and A_0 signals address the low, high, or both bytes. From T_1 to T_4 the M/$\overline{IO}$ signal indicates a memory or I/O operation. At T_2 the address is removed from the local bus and the bus goes to a high impedance state. The read control signal is also asserted at T_2. The read ($\overline{RD}$) signal causes the addressed device to enable its data bus drivers to the local bus. Some time later valid data will be available on the bus and the addressed device will drive the READY line HIGH. When the processor returns the read signal to a HIGH level, the addressed device will again 3-state its bus drivers. If a transceiver is required to buffer the 8086 local bus, signals DT/$\overline{R}$ and $\overline{DEN}$ are provided by the 8086.

A write cycle also begins with the assertion of ALE and the emission of the address. The M/$\overline{IO}$ signal is again asserted to indicate a memory or I/O write operation. In the T_2 immediately following the address emission the processor emits the data to be written into the addressed location. This data remains valid until the middle of T_4. During T_2, T_3, and T_W the processor asserts the write control signal. The write ($\overline{WR}$) signal becomes active at the beginning of T_2 as opposed to the read which is delayed somewhat into T_2 to provide time for the bus to float.

The $\overline{BHE}$ and A_0 signals are used to select the proper byte(s) of the memory/IO word to be read or written according to the following table:

$\overline{BHE}$	A0	Characteristics
0	0	Whole word
0	1	Upper byte from/to odd address
1	0	Lower byte from/to even address
1	1	None

I/O ports are addressed in the same manner as memory location. Even addressed bytes are transferred on the D_7–D_0 bus lines and odd addressed bytes on D_{15}–D_8.

The basic difference between the interrupt acknowledge cycle and a read cycle is that the interrupt acknowledge signal ($\overline{INTA}$) is asserted in place of the read ($\overline{RD}$) signal and the address bus is floated. (See Figure 6.) In the second of two successive INTA cycles, a byte of information is read from bus

lines D_7–D_0 as supplied by the inerrupt system logic (i.e., 8259A Priority Interrupt Controller). This byte identifies the source (type) of the interrupt. It is multiplied by four and used as a pointer into an interrupt vector lookup table, as described earlier.

BUS TIMING—MEDIUM SIZE SYSTEMS

For medium size systems the MN/$\overline{MX}$ pin is connected to V_{SS} and the 8288 Bus Controller is added to the system as well as a latch for latching the system address, and a transceiver to allow for bus loading greater than the 8086 is capable of handling. Signals ALE, DEN, and DT/$\overline{R}$ are generated by the 8288 instead of the processor in this configuration although their timing remains relatively the same. The 8086 status outputs ($\overline{S_2}$, $\overline{S_1}$, and $\overline{S_0}$) provide type-of-cycle information and become 8288 inputs. This bus cycle information specifies read (code, data, or I/O), write (data or I/O), interrupt acknowledge, or software halt. The 8288 thus issues control signals specifying memory read or write, I/O read or write, or interrupt acknowledge. The 8288 provides two types of write strobes, normal and advanced, to be applied as required. The normal write strobes have data valid at the leading edge of write. The advanced write strobes have the same timing as read strobes, and hence data isn't valid at the leading edge of write. The transceiver receives the usual DIR and $\overline{G}$ inputs from the 8288's DT/$\overline{R}$ and DEN.

The pointer into the interrupt vector table, which is passed during the second INTA cycle, can derive from an 8259A located on either the local bus or the system bus. If the master 8259A Priority Interrupt Controller is positioned on the local bus, a TTL gate is required to disable the transceiver when reading from the master 8259A during the interrupt acknowledge sequence and software "poll".

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature .....−65°C to +150°C
Voltage on Any Pin with
Respect to Ground.....−1.0V to +7V
Power Dissipation.....2.5W

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

(8086: T_A = 0°C to 70°C, V_{CC} = 5V ±10%)
(8086-1: T_A = 0°C to 70°C, V_{CC} = 5V ±5%)
(8086-2: T_A = 0°C to 70°C, V_{CC} = 5V ±5%)

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	+0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.5 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = − 400 μA
I_{CC}	Power Supply Current: 8086 8086-1 8086-2		340 360 350	mA	T_A = 25°C
I_{LI}	Input Leakage Current		±10	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{LO}	Output Leakage Current		±10	μA	0.45V ≤ V_{OUT} ≤ V_{CC}
V_{CL}	Clock Input Low Voltage	−0.5	+0.6	V	
V_{CH}	Clock Input High Voltage	3.9	V_{CC} + 1.0	V	
C_{IN}	Capacitance of Input Buffer (All input except AD_0–AD_{15}, $\overline{RQ}/\overline{GT}$)		15	pF	fc = 1 MHz
C_{IO}	Capacitance of I/O Buffer (AD_0–AD_{15}, $\overline{RQ}/\overline{GT}$)		15	pF	fc = 1 MHz

NOTES:
1. V_{IL} tested with MN/$\overline{MX}$ Pin = 0V.
2. V_{IH} tested with MN/$\overline{MX}$ Pin = 5V.
MN/$\overline{MX}$ Pin is a Strap Pin.

A.C. CHARACTERISTICS

(8086: T_A = 0°C to 70°C, V_{CC} = 5V ± 10%)
(8086-1: T_A = 0°C to 70°C, V_{CC} = 5V ± 5%)
(8086-2: T_A = 0°C to 70°C, V_{CC} = 5V ± 5%)

MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

Symbol	Parameter	8086		8086-1		8086-2		Units	Test Conditions
		Min	Max	Min	Max	Min	Max		
TCLCL	CLK Cycle Period	200	500	100	500	125	500	ns	
TCLCH	CLK Low Time	118		53		68		ns	
TCHCL	CLK High Time	69		39		44		ns	
TCH1CH2	CLK Rise Time		10		10		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10		10		10	ns	From 3.5V to 1.0V
TDVCL	Data in Setup Time	30		5		20		ns	
TCLDX	Data in Hold Time	10		10		10		ns	
TR1VCL	RDY Setup Time into 8284A (See Notes 1, 2)	35		35		35		ns	
TCLR1X	RDY Hold Time into 8284A (See Notes 1, 2)	0		0		0		ns	
TRYHCH	READY Setup Time into 8086	118		53		68		ns	
TCHRYX	READY Hold Time into 8086	30		20		20		ns	
TRYLCL	READY Inactive to CLK (See Note 3)	−8		−10		−8		ns	
THVCH	HOLD Setup Time	35		20		20		ns	
TINVCH	INTR, NMI, $\overline{\text{TEST}}$ Setup Time (See Note 2)	30		15		15		ns	
TILIH	Input Rise Time (Except CLK)		20		20		20	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK)		12		12		12	ns	From 2.0V to 0.8V

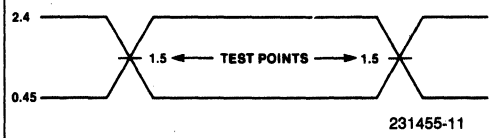
A.C. CHARACTERISTICS (Continued)

TIMING RESPONSES

Symbol	Parameter	8086		8086-1		8086-2		Units	Test Conditions
		Min	Max	Min	Max	Min	Max		
TCLAV	Address Valid Delay	10	110	10	50	10	60	ns	*C_L = 20–100 pF for all 8086 Outputs (In addition to 8086 selfload)
TCLAX	Address Hold Time	10		10		10		ns	
TCLAZ	Address Float Delay	TCLAX	80	10	40	TCLAX	50	ns	
TLHLL	ALE Width	TCLCH-20		TCLCH-10		TCLCH-10		ns	
TCLLH	ALE Active Delay		80		40		50	ns	
TCHLL	ALE Inactive Delay		85		45		55	ns	
TLLAX	Address Hold Time	TCHCL-10		TCHCL-10		TCHCL-10		ns	
TCLDV	Data Valid Delay	10	110	10	50	10	60	ns	
TCHDX	Data Hold Time	10		10		10		ns	
TWHDX	Data Hold Time After WR	TCLCH-30		TCLCH-25		TCLCH-30		ns	
TCVCTV	Control Active Delay 1	10	110	10	50	10	70	ns	
TCHCTV	Control Active Delay 2	10	110	10	45	10	60	ns	
TCVCTX	Control Inactive Delay	10	110	10	50	10	70	ns	
TAZRL	Address Float to READ Active	0		0		0		ns	
TCLRL	$\overline{RD}$ Active Delay	10	165	10	70	10	100	ns	
TCLRH	$\overline{RD}$ Inactive Delay	10	150	10	60	10	80	ns	
TRHAV	$\overline{RD}$ Inactive to Next Address Active	TCLCL-45		TCLCL-35		TCLCL-40		ns	
TCLHAV	HLDA Valid Delay	10	160	10	60	10	100	ns	
TRLRH	$\overline{RD}$ Width	2TCLCL-75		2TCLCL-40		2TCLCL-50		ns	
TWLWH	$\overline{WR}$ Width	2TCLCL-60		2TCLCL-35		2TCLCL-40		ns	
TAVAL	Address Valid to ALE Low	TCLCH-60		TCLCH-35		TCLCH-40		ns	
TOLOH	Output Rise Time		20		20		20	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time		12		12		12	ns	From 2.0V to 0.8V

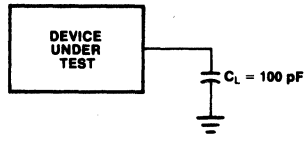
NOTES:
1. Signal at 8284A shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T2 state. (8 ns into T3).

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 1.5V for both a Logic "1" and "0".

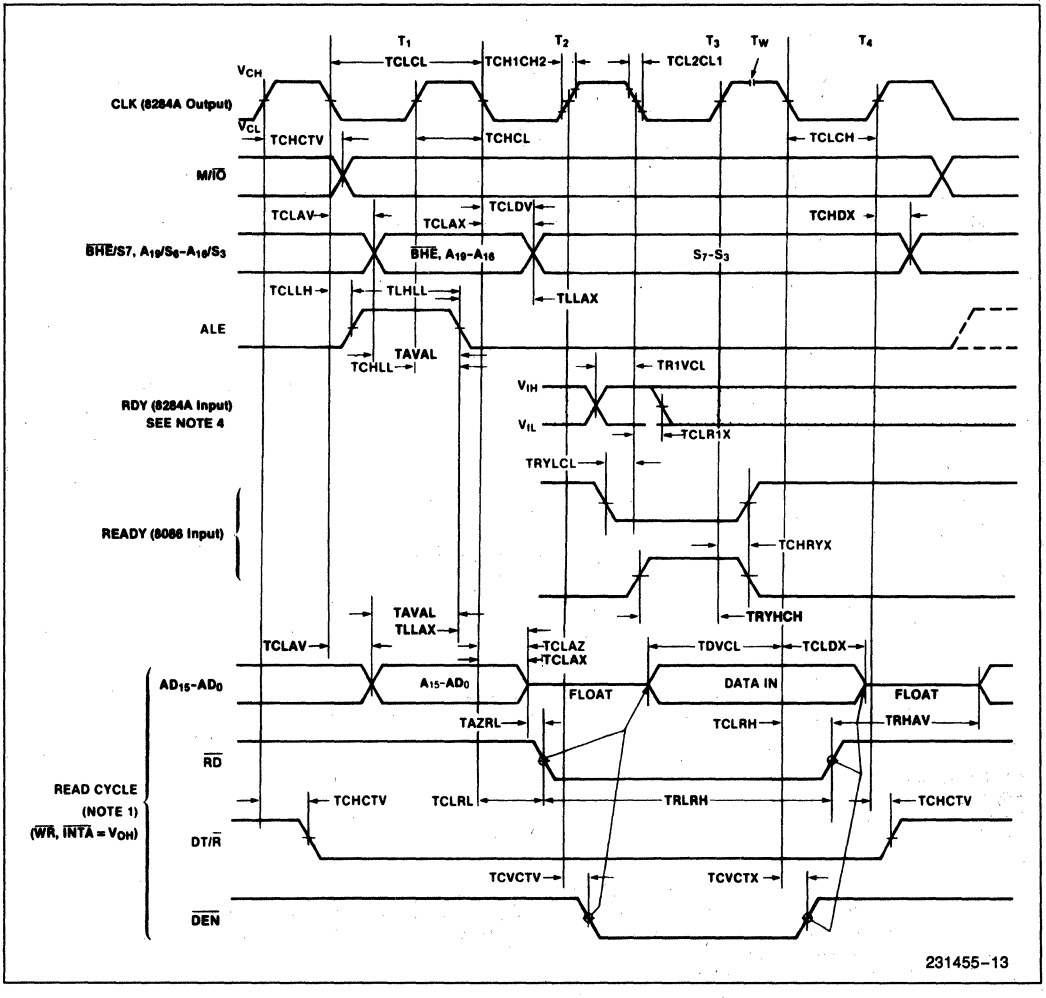
A.C. TESTING LOAD CIRCUIT



C_L Includes Jig Capacitance

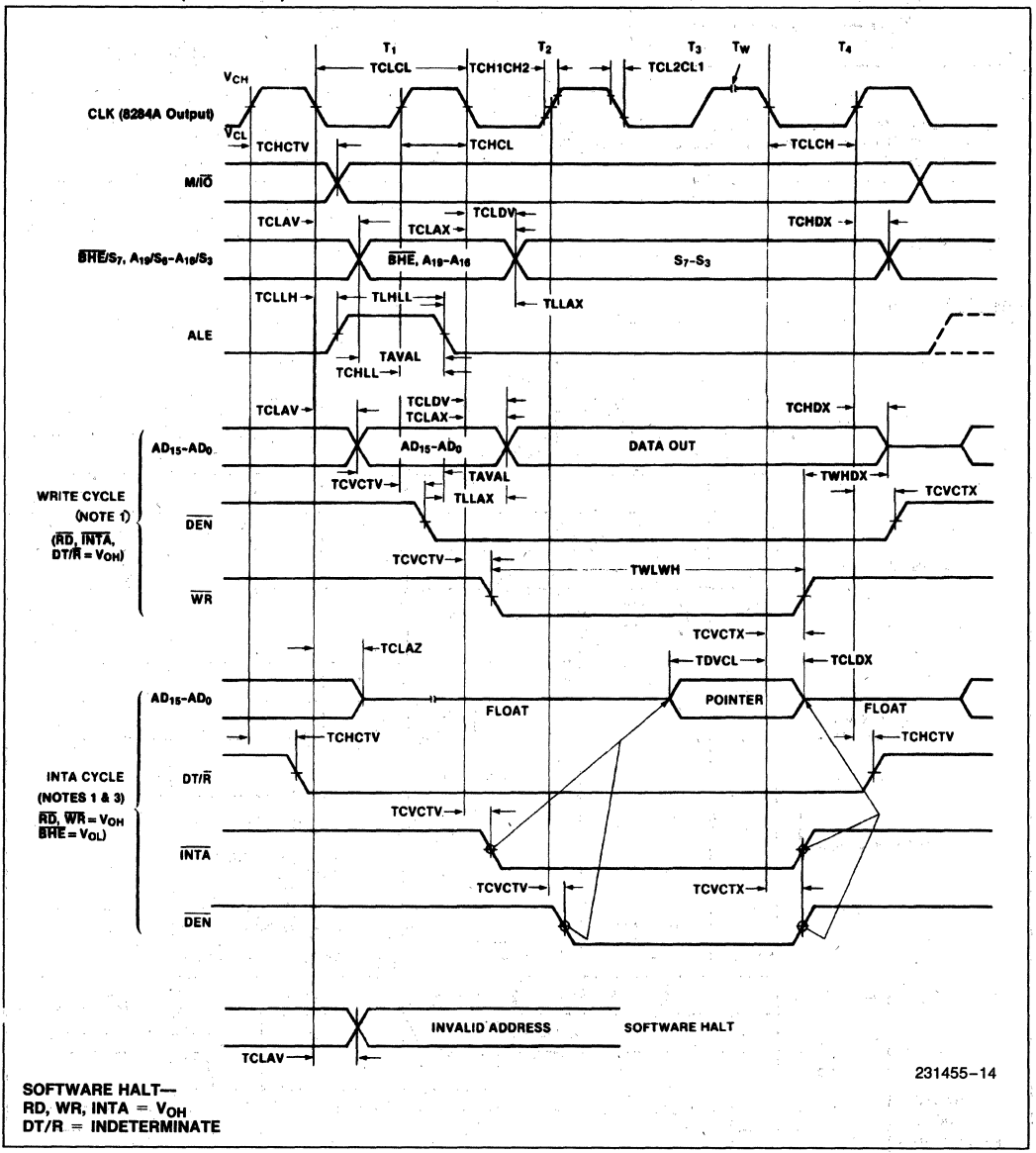
WAVEFORMS

MINIMUM MODE



WAVEFORMS (Continued)

MINIMUM MODE (Continued)



SOFTWARE HALT—
RD, WR, INTA = V_{OH}
DT/R = INDETERMINATE

NOTES:
1. All signals switch between V_{OH} and V_{OL} unless otherwise specified.
2. RDY is sampled near the end of T_2, T_3, T_W to determine if T_W machines states are to be inserted.
3. Two INTA cycles run back-to-back. The 8086 LOCAL ADDR/DATA BUS is floating during both INTA cycles. Control signals shown for second INTA cycle.
4. Signals at 8284A are shown for reference only.
5. All timing measurements are made at 1.5V unless otherwise noted.

A.C. CHARACTERISTICS

MAX MODE SYSTEM (USING 8288 BUS CONTROLLER) TIMING REQUIREMENTS

Symbol	Parameter	8086		8086-1		8086-2		Units	Test Conditions
		Min	Max	Min	Max	Min	Max		
TCLCL	CLK Cycle Period	200	500	100	500	125	500	ns	
TCLCH	CLK Low Time	118		53		68		ns	
TCHCL	CLK High Time	69		39		44		ns	
TCH1CH2	CLK Rise Time		10		10		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10		10		10	ns	From 3.5V to 1.0V
TDVCL	Data in Setup Time	30		5		20		ns	
TCLDX	Data in Hold Time	10		10		10		ns	
TR1VCL	RDY Setup Time into 8284A (Notes 1, 2)	35		35		35		ns	
TCLR1X	RDY Hold Time into 8284A (Notes 1, 2)	0		0		0		ns	
TRYHCH	READY Setup Time into 8086	118		53		68		ns	
TCHRYX	READY Hold Time into 8086	30		20		20		ns	
TRYLCL	READY Inactive to CLK (Note 4)	−8		−10		−8		ns	
TINVCH	Setup Time for Recognition (INTR, NMI, $\overline{TEST}$) (Note 2)	30		15		15		ns	
TGVCH	$\overline{RQ}/\overline{GT}$ Setup Time (Note 5)	30		15		15		ns	
TCHGX	$\overline{RQ}$ Hold Time into 8086	40		20		30		ns	
TILIH	Input Rise Time (Except CLK)		20		20		20	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK)		12		12		12	ns	From 2.0V to 0.8V

A.C. CHARACTERISTICS (Continued)

TIMING RESPONSES

Symbol	Parameter	8086		8086-1		8086-2		Units	Test Conditions
		Min	Max	Min	Max	Min	Max		
TCLML	Command Active Delay (See Note 1)	10	35	10	35	10	35	ns	
TCLMH	Command Inactive Delay (See Note 1)	10	35	10	35	10	35	ns	
TRYHSH	READY Active to Status Passive (See Note 3)		110		45		65	ns	
TCHSV	Status Active Delay	10	110	10	45	10	60	ns	
TCLSH	Status Inactive Delay	10	130	10	55	10	70	ns	
TCLAV	Address Valid Delay	10	110	10	50	10	60	ns	
TCLAX	Address Hold Time	10		10		10		ns	
TCLAZ	Address Float Delay	TCLAX	80	10	40	TCLAX	50	ns	
TSVLH	Status Valid to ALE High (See Note 1)		15		15		15	ns	
TSVMCH	Status Valid to MCE High (See Note 1)		15		15		15	ns	
TCLLH	CLK Low to ALE Valid (See Note 1)		15		15		15	ns	C_L = 20–100 pF for all 8086 Outputs (In addition to 8086 self-load)
TCLMCH	CLK Low to MCE High (See Note 1)		15		15		15	ns	
TCHLL	ALE Inactive Delay (See Note 1)		15		15		15	ns	
TCLMCL	MCE Inactive Delay (See Note 1)		15		15		15	ns	
TCLDV	Data Valid Delay	10	110	10	50	10	60	ns	
TCHDX	Data Hold Time	10		10		10		ns	
TCVNV	Control Active Delay (See Note 1)	5	45	5	45	5	45	ns	
TCVNX	Control Inactive Delay (See Note 1)	10	45	10	45	10	45	ns	
TAZRL	Address Float to READ Active	0		0		0		ns	
TCLRL	RD Active Delay	10	165	10	70	10	100	ns	
TCLRH	RD Inactive Delay	10	150	10	60	10	80	ns	

A.C. CHARACTERISTICS (Continued)

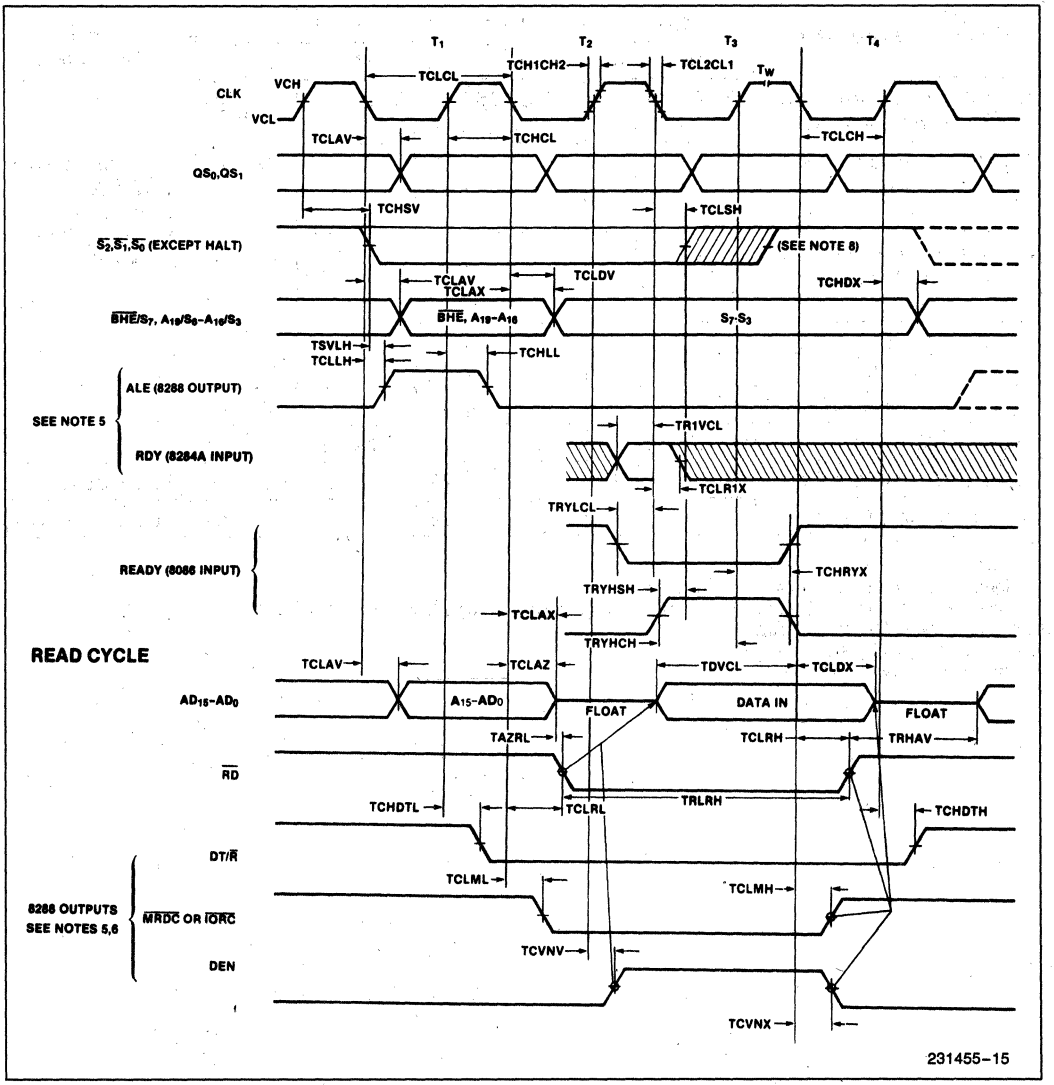
TIMING RESPONSES (Continued)

Symbol	Parameter	8086		8086-1		8086-2		Units	Test Conditions
		Min	Max	Min	Max	Min	Max		
TRHAV	RD Inactive to Next Address Active	TCLCL-45		TCLCL-35		TCLCL-40		ns	C_L = 20–100 pF for all 8086 Outputs (In addition to 8086 self-load)
TCHDTL	Direction Control Active Delay (Note 1)		50		50		50	ns	
TCHDTH	Direction Control Inactive Delay (Note 1)		30		30		30	ns	
TCLGL	GT Active Delay (Note 5)	0	85	0	38	0	50	ns	
TCLGH	GT Inactive Delay	0	85	0	45	0	50	ns	
TRLRH	RD Width	2TCLCL-75		2TCLCL-40		2TCLCL-50		ns	
TOLOH	Output Rise Time		20		20		20	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time		12		12		12	ns	From 2.0V to 0.8V

NOTES:
1. Signal at 8284A or 8288 shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T3 and wait states.
4. Applies only to T2 state (8 ns into T3).
5. Change from 1985 Handbook.

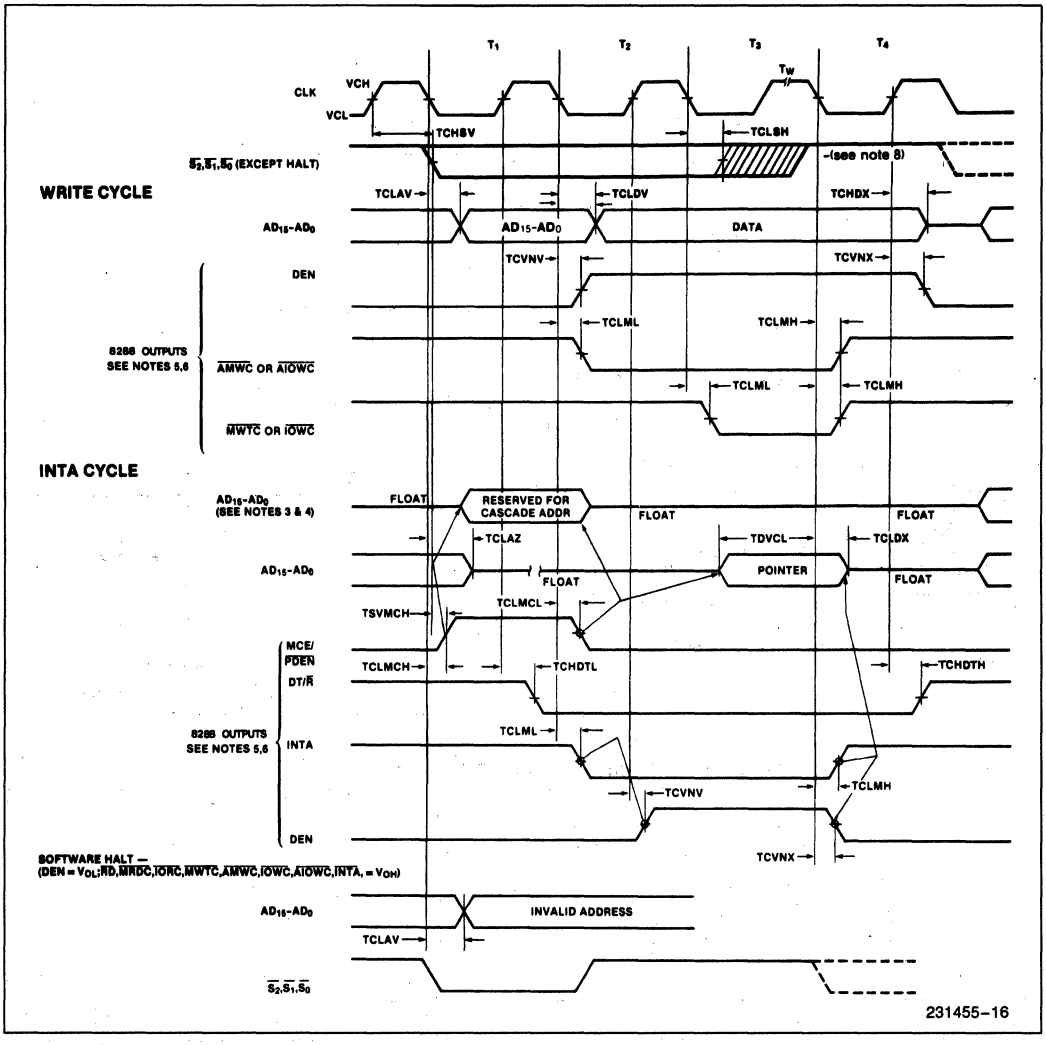
WAVEFORMS

MAXIMUM MODE



WAVEFORMS (Continued)

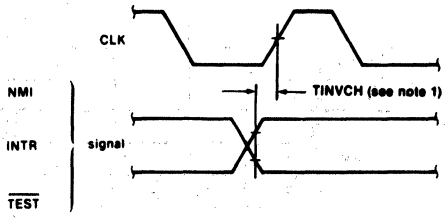
MAXIMUM MODE (Continued)



NOTES:
1. All signals switch between V_{OH} and V_{OL} unless otherwise specified.
2. RDY is sampled near the end of T_2, T_3, T_W to determine if T_W machines states are to be inserted.
3. Cascade address is valid between first and second INTA cycle.
4. Two INTA cycles run back-to-back. The 8086 LOCAL ADDR/DATA BUS is floating during both INTA cycles. Control for pointer address is shown for second INTA cycle.
5. Signals at 8284A or 8288 are shown for reference only.
6. The issuance of the 8288 command and control signals ($\overline{MRDC}$, $\overline{MWTC}$, $\overline{AMWC}$, $\overline{IORC}$, $\overline{IOWC}$, $\overline{AIOWC}$, $\overline{INTA}$ and DEN) lags the active high 8288 CEN.
7. All timing measurements are made at 1.5V unless otherwise noted.
8. Status inactive in state just prior to T_4.

WAVEFORMS (Continued)

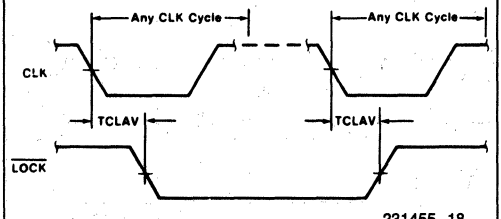
ASYNCHRONOUS SIGNAL RECOGNITION



231455-17

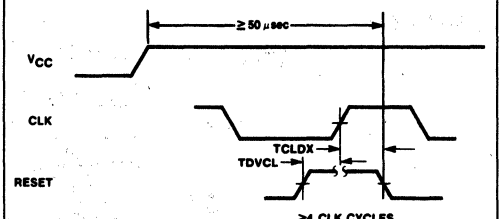
NOTE:
1. Setup requirements for asynchronous signals only to guarantee recognition at next CLK.

BUS LOCK SIGNAL TIMING (MAXIMUM MODE ONLY)



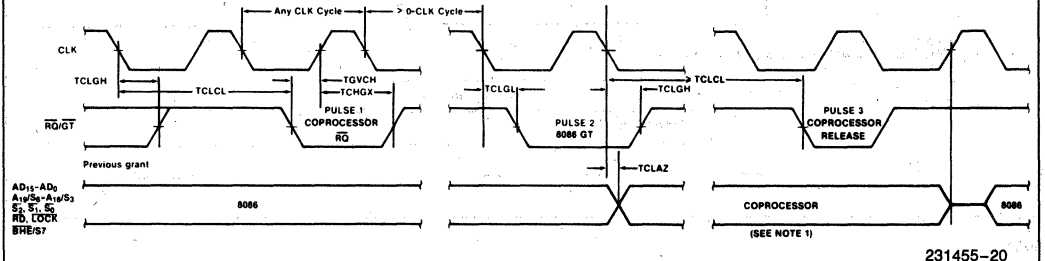
231455-18

RESET TIMING



231455-19

REQUEST/GRANT SEQUENCE TIMING (MAXIMUM MODE ONLY)



231455-20

NOTE:
The coprocessor may not drive the buses outside the region shown without risking contention.

WAVEFORMS (Continued)

HOLD/HOLD ACKNOWLEDGE TIMING (MINIMUM MODE ONLY)

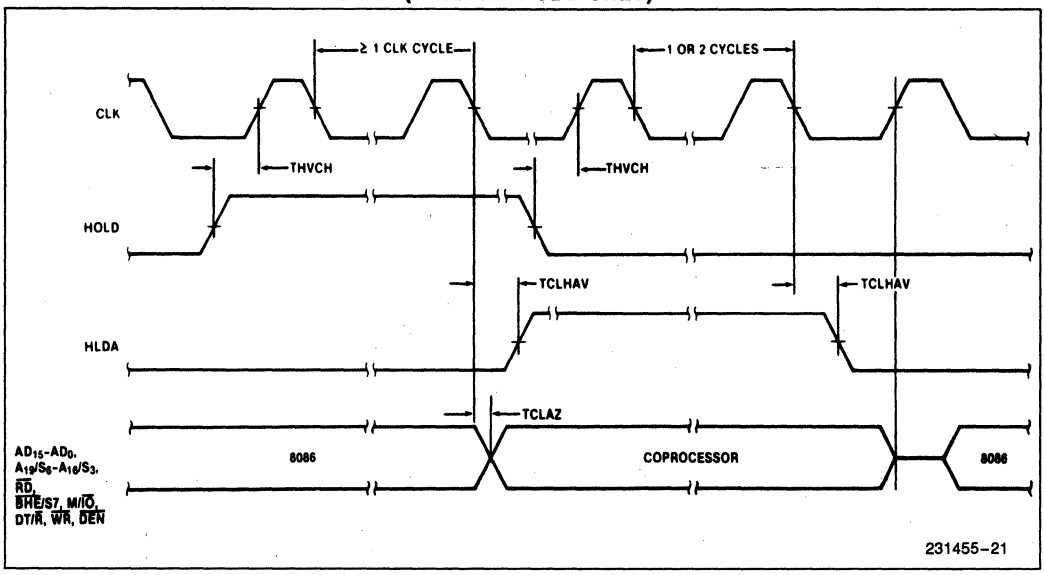


Table 2. Instruction Set Summary

Mnemonic and Description	Instruction Code			
DATA TRANSFER				
MOV = Move:	76543210	76543210	76543210	76543210
Register/Memory to/from Register	100010dw	mod reg r/m		
Immediate to Register/Memory	1100011w	mod 000 r/m	data	data if w = 1
Immediate to Register	1011wreg	data	data if w = 1	
Memory to Accumulator	1010000w	addr-low	addr-high	
Accumulator to Memory	1010001w	addr-low	addr-high	
Register/Memory to Segment Register	10001110	mod 0 reg r/m		
Segment Register to Register/Memory	10001100	mod 0 reg r/m		
PUSH = Push:				
Register/Memory	11111111	mod 110 r/m		
Register	01010reg			
Segment Register	000reg110			
POP = Pop:				
Register/Memory	10001111	mod 000 r/m		
Register	01011reg			
Segment Register	000reg111			
XCHG = Exchange:				
Register/Memory with Register	1000011w	mod reg r/m		
Register with Accumulator	10010reg			
IN = Input from:				
Fixed Port	1110010w	port		
Variable Port	1110110w			
OUT = Output to:				
Fixed Port	1110011w	port		
Variable Port	1110111w			
XLAT = Translate Byte to AL	11010111			
LEA = Load EA to Register	10001101	mod reg r/m		
LDS = Load Pointer to DS	11000101	mod reg r/m		
LES = Load Pointer to ES	11000100	mod reg r/m		
LAHF = Load AH with Flags	10011111			
SAHF = Store AH into Flags	10011110			
PUSHF = Push Flags	10011100			
POPF = Pop Flags	10011101			

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
ARITHMETIC	76543210	76543210	76543210	76543210
ADD = Add:				
Reg./Memory with Register to Either	000000dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod 000 r/m	data	data if s: w = 01
Immediate to Accumulator	0000010w	data	data if w = 1	
ADC = Add with Carry:				
Reg./Memory with Register to Either	000100dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod 010 r/m	data	data if s: w = 01
Immediate to Accumulator	0001010w	data	data if w = 1	
INC = Increment:				
Register/Memory	1111111w	mod 000 r/m		
Register	01000reg			
AAA = ASCII Adjust for Add	00110111			
BAA = Decimal Adjust for Add	00100111			
SUB = Subtract:				
Reg./Memory and Register to Either	001010dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod 101 r/m	data	data if s w = 01
Immediate from Accumulator	0010110w	data	data if w = 1	
SSB = Subtract with Borrow				
Reg./Memory and Register to Either	000110dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod 011 r/m	data	data if s w = 01
Immediate from Accumulator	0001110w	data	data if w = 1	
DEC = Decrement:				
Register/memory	1111111w	mod 001 r/m		
Register	01001 reg			
NEG = Change sign	1111011w	mod 011 r/m		
CMP = Compare:				
Register/Memory and Register	001110dw	mod reg r/m		
Immediate with Register/Memory	100000sw	mod 111 r/m	data	data if s w = 01
Immediate with Accumulator	0011110w	data	data if w = 1	
AAS = ASCII Adjust for Subtract	00111111			
DAS = Decimal Adjust for Subtract	00101111			
MUL = Multiply (Unsigned)	1111011w	mod 100 r/m		
IMUL = Integer Multiply (Signed)	1111011w	mod 101 r/m		
AAM = ASCII Adjust for Multiply	11010100	00001010		
DIV = Divide (Unsigned)	1111011w	mod 110 r/m		
IDIV = Integer Divide (Signed)	1111011w	mod 111 r/m		
AAD = ASCII Adjust for Divide	11010101	00001010		
CBW = Convert Byte to Word	10011000			
CWD = Convert Word to Double Word	10011001			

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
LOGIC	76543210	76543210	76543210	76543210
NOT = Invert	1111011w	mod 010 r/m		
SHL/SAL = Shift Logical/Arithmetic Left	110100vw	mod 100 r/m		
SHR = Shift Logical Right	110100vw	mod 101 r/m		
SAR = Shift Arithmetic Right	110100vw	mod 111 r/m		
ROL = Rotate Left	110100vw	mod 000 r/m		
ROR = Rotate Right	110100vw	mod 001 r/m		
RCL = Rotate Through Carry Flag Left	110100vw	mod 010 r/m		
RCR = Rotate Through Carry Right	110100vw	mod 011 r/m		
AND = **And:**				
Reg./Memory and Register to Either	001000dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 100 r/m	data	data if w = 1
Immediate to Accumulator	0010010w	data	data if w = 1	
TEST = **And Function to Flags, No Result:**				
Register/Memory and Register	1000010w	mod reg r/m		
Immediate Data and Register/Memory	1111011w	mod 000 r/m	data	data if w = 1
Immediate Data and Accumulator	1010100w	data	data if w = 1	
OR = **Or:**				
Reg./Memory and Register to Either	000010dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 001 r/m	data	data if w = 1
Immediate to Accumulator	0000110w	data	data if w = 1	
XOR = **Exclusive or:**				
Reg./Memory and Register to Either	001100dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 110 r/m	data	data if w = 1
Immediate to Accumulator	0011010w	data	data if w = 1	
STRING MANIPULATION				
REP = Repeat	1111001z			
MOVS = Move Byte/Word	1010010w			
CMPS = Compare Byte/Word	1010011w			
SCAS = Scan Byte/Word	1010111w			
LODS = Load Byte/Wd to AL/AX	1010110w			
STOS = Stor Byte/Wd from AL/A	1010101w			
CONTROL TRANSFER				
CALL = **Call:**				
Direct within Segment	11101000	disp-low	disp-high	
Indirect within Segment	11111111	mod 010 r/m		
Direct Intersegment	10011010	offset-low	offset-high	
		seg-low	seg-high	
Indirect Intersegment	11111111	mod 011 r/m		

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code		
JMP = Unconditional Jump:	76543210	76543210	76543210
Direct within Segment	11101001	disp-low	disp-high
Direct within Segment-Short	11101011	disp	
Indirect within Segment	11111111	mod 100 r/m	
Direct Intersegment	11101010	offset-low	offset-high
		seg-low	seg-high
Indirect Intersegment	11111111	mod 101 r/m	
RET = Return from CALL:			
Within Segment	11000011		
Within Seg Adding Immed to SP	11000010	data-low	data-high
Intersegment	11001011		
Intersegment Adding Immediate to SP	11001010	data-low	data-high
JE/JZ = Jump on Equal/Zero	01110100	disp	
JL/JNGE = Jump on Less/Not Greater or Equal	01111100	disp	
JLE/JNG = Jump on Less or Equal/ Not Greater	01111110	disp	
JB/JNAE = Jump on Below/Not Above or Equal	01110010	disp	
JBE/JNA = Jump on Below or Equal/ Not Above	01110110	disp	
JP/JPE = Jump on Parity/Parity Even	01111010	disp	
JO = Jump on Overflow	01110000	disp	
JS = Jump on Sign	01111000	disp	
JNE/JNZ = Jump on Not Equal/Not Zero	01110101	disp	
JNL/JGE = Jump on Not Less/Greater or Equal	01111101	disp	
JNLE/JG = Jump on Not Less or Equal/ Greater	01111111	disp	
JNB/JAE = Jump on Not Below/Above or Equal	01110011	disp	
JNBE/JA = Jump on Not Below or Equal/Above	01110111	disp	
JNP/JPO = Jump on Not Par/Par Odd	01111011	disp	
JNO = Jump on Not Overflow	01110001	disp	
JNS = Jump on Not Sign	01111001	disp	
LOOP = Loop CX Times	11100010	disp	
LOOPZ/LOOPE = Loop While Zero/Equal	11100001	disp	
LOOPNZ/LOOPNE = Loop While Not Zero/Equal	11100000	disp	
JCXZ = Jump on CX Zero	11100011	disp	
INT = Interrupt			
Type Specified	11001101	type	
Type 3	11001100		
INTO = Interrupt on Overflow	11001110		
IRET = Interrupt Return	11001111		

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code	
	76543210	76543210
PROCESSOR CONTROL		
CLC = Clear Carry	11111000	
CMC = Complement Carry	11110101	
STC = Set Carry	11111001	
CLD = Clear Direction	11111100	
STD = Set Direction	11111101	
CLI = Clear Interrupt	11111010	
STI = Set Interrupt	11111011	
HLT = Halt	11110100	
WAIT = Wait	10011011	
ESC = Escape (to External Device)	11011xxx	mod x x x r/m
LOCK = Bus Lock Prefix	11110000	

NOTES:
AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value
Greater = more positive;
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
if w = 1 then word instruction; if w = 0 then byte instruction
if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16 bits, disp-high is absent
if mod = 10 then DISP = disp-high; disp-low
if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)
*except if mod = 00 and r/m = 110 then EA = disp-high; disp-low.

if s w = 01 then 16 bits of immediate data form the operand
if s w = 11 then an immediate data byte is sign extended to form the 16-bit operand
if v = 0 then "count" = 1; if v = 1 then "count" in (CL)
x = don't care
z is used for string primitives for comparison with ZF FLAG

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:
FLAGS = X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

DATA SHEET REVISION REVIEW

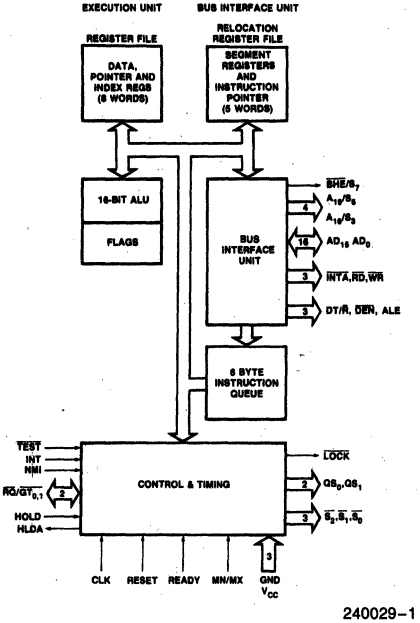
The following list represents key differences between this and the -002 data sheet. Please review this summary carefully.

1. In the Pin Description Table (Table 1), the description of the HLDA signal being issued has been corrected. HLDA will be issued in the middle of either the T_4 or T_i state.

80C86A
16-BIT CHMOS MICROPROCESSOR

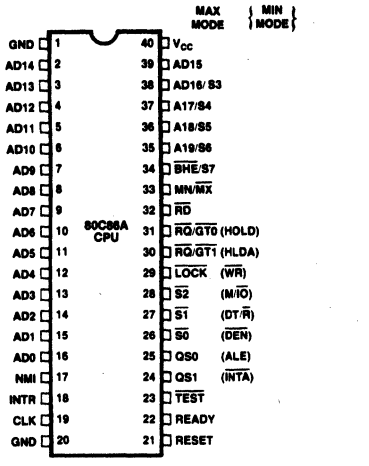
- **Pin-for-Pin and Functionally Compatible to Industry Standard HMOS 8086**
- **Fully Static Design with Frequency Range from D.C. to:**
  — 8 MHz for 80C86A-2
- **Low Power Operation**
  — Operating I_{CC} = 10 mA/MHz
  — Standby I_{CCS} = 500 μA max
- **Bus-Hold Circuitry Eliminates Pull-Up Resistors**
- **Direct Addressing Capability of 1 MByte of Memory**
- **Architecture Designed for Powerful Assembly Language and Efficient High Level Languages**
- **24 Operand Addressing Modes**
- **Byte, Word and Block Operations**
- **8 and 16-Bit Signed and Unsigned Arithmetic**
  — Binary or Decimal
  — Multiply and Divide
- **Available in 40-Lead Plastic DIP**

The Intel 80C86A is a high performance, CHMOS version of the industry standard HMOS 8086 16-bit CPU. The 80C86A available in 8 MHz clock rates, offers two modes of operation: MINimum for small systems and MAXimum for larger applications such as multiprocessing. It is available in 40-pin DIP package.



240029-1

Figure 1. 80C86A CPU Block Diagram



240029-2

Figure 2. 80C86A 40-Lead DIP Configuration

September 1988
Order Number: 240029-002

Table 1. Pin Description

The following pin function descriptions are for 80C86AA systems in either minimum or maximum mode. The "Local Bus" in these descriptions is the direct multiplexed bus interface connection to the 80C86A (without regard to additional bus buffers).

Symbol	Pin No.	Type	Name and Function
AD_{15}–AD_0	2–16, 39	I/O	**ADDRESS DATA BUS:** These lines constitute the time multiplexed memory/IO address (T_1) and data (T_2, T_3, T_W, T_4) bus. A_0 is analogous to $\overline{BHE}$ for the lower byte of the data bus, pins D_7–D_0. It is LOW during T_1 when a byte is to be transferred on the lower portion of the bus in memory or I/O operations. Eight-bit oriented devices tied to the lower half would normally use A_0 to condition chip select functions. (See $\overline{BHE}$.) These lines are active HIGH and float to 3-state OFF(1) during interrupt acknowledge and local bus "hold acknowledge."
A_{19}/S_6, A_{18}/S_5, A_{17}/S_4, A_{16}/S_3	35–38	O	**ADDRESS/STATUS:** During T_1 these are the four most significant address lines for memory operations. During I/O operations these lines are LOW. During memory and I/O operations, status information is available on these lines during T_2, T_3, T_W, and T_4. The status of the interrupt enable FLAG bit (S_5) is updated at the beginning of each CLK cycle. A_{17}/S_4 and A_{16}/S_3 are encoded as shown. This information indicates which relocation register is presently being used for data accessing. These lines float to 3-state OFF(1) during local bus "hold acknowledge."

A_{17}/S_4	A_{16}/S_3	Characteristics
0 (LOW)	0	Alternate Data
0	1	Stack
1 (HIGH)	0	Code or None
1	1	Data
S_6 is 0 (LOW)		

$\overline{BHE}/S_7$	34	O	**BUS HIGH ENABLE/STATUS:** During T_1 the bus high enable signal ($\overline{BHE}$) should be used to enable data onto the most significant half of the data bus, pins D_{15}–D_8. Eight-bit oriented devices tied to the upper half of the bus would normally use $\overline{BHE}$ to condition chip select functions. $\overline{BHE}$ is LOW during T_1 for read, write, and interrupt acknowledge cycles when a byte is to be transferred on the high portion of the bus. The S_7 status information is available during T_2, T_3, and T_4. The signal is active LOW, and floats to 3-state OFF(1) in "hold." It is LOW during T_1 for the first interrupt acknowledge cycle.

$\overline{BHE}$	A_0	Characteristics
0	0	Whole word
0	1	Upper byte from/ to odd address
1	0	Lower byte from/ to even address
1	1	None

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
$\overline{RD}$	32	O	**READ:** Read strobe indicates that the processor is performing a memory of I/O read cycle, depending on the state of the S_2 pin. This signal is used to read devices which reside on the 80C86A local bus. $\overline{RD}$ is active LOW during T_2, T_3 and T_W of any read cycle, and is guaranteed to remain HIGH in T_2 until the 80C86A local bus has floated. This floats to 3-state OFF in "hold acknowledge."
READY	22	I	**READY:** is the acknowledgement from the addressed memory or I/O device that it will complete the data transfer. The READY signal from memory/IO is synchronized by the 82C84A Clock Generator to form READY. This signal is active HIGH. The 80C86A READY input is not synchronized. Correct operation is not guaranteed if the setup and hold times are not met.
INTR	18	I	**INTERRUPT REQUEST:** is a level triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored to via an interrupt vector lookup table located in system memory. It can be internally masked by software resetting the interrupt enable bit. INTR is internally synchronized. This signal is active HIGH.
$\overline{TEST}$	23	I	**$\overline{TEST}$:** input is examined by the "Wait" instruction. If the $\overline{TEST}$ input is LOW execution continues, otherwise the processor waits in an "Idle" state. This input is synchronized internally during each clock cycle on the leading edge of CLK.
NMI	17	I	**NON-MASKABLE INTERRUPT:** an edge triggered input which causes a type 2 interrupt. A subroutine is vectored to via an interrupt vector lookup table located in system memory. NMI is not maskable internally by software. A transition from a LOW to HIGH initiates the interrupt at the end of the current instruction. This input is internally synchronized.
RESET	21	I	**RESET:** causes the processor to immediately terminate its present activity. The signal must be active HIGH for at least four clock cycles. It restarts execution, as described in the Instruction Set description, when RESET returns LOW. RESET is internally synchronized.
CLK	19	I	**CLOCK:** provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing.
V_{CC}	40		**V_{CC}:** +5V power supply pin.
GND	1, 20		**GROUND:** Both must be connected.
MN/$\overline{MX}$	33	I	**MINIMUM/MAXIMUM:** indicates what mode the processor is to operate in. The two modes are discussed in the following sections.

Table 1. Pin Description (Continued)

The following pin function descriptions are for the 80C86A/82C88 system in maximum mode (i.e., MN/$\overline{MX}$ = V_{SS}). Only the pin functions which are unique to maximum mode are described; all other pin functions are as described above.

<table>
<tr><th>Symbol</th><th>Pin No.</th><th>Type</th><th>Name and Function</th></tr>
<tr><td>$\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$</td><td>26–28</td><td>O</td><td>STATUS: active during T_4, T_1, and T_2 and is returned to the passive state (1,1,1) during T_3 or during T_W when READY is HIGH. This status is used by the 82C88 Bus Controller to generate all memory and I/O access control signals. Any change by $\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$ during T_4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T_3 or T_W is used to indicate the end of a bus cycle.
These signals float to 3-state OFF(1) in "hold acknowledge." These status lines are encoded as shown.
<table>
<tr><th>$\overline{S_2}$</th><th>$\overline{S_1}$</th><th>$\overline{S_0}$</th><th>Characteristics</th></tr>
<tr><td>0 (LOW)</td><td>0</td><td>0</td><td>Interrupt Acknowledge</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Read I/O Port</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Write I/O Port</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Halt</td></tr>
<tr><td>1 (HIGH)</td><td>0</td><td>0</td><td>Code Access</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Read Memory</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Write Memory</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Passive</td></tr>
</table>
</td></tr>
<tr><td>$\overline{RQ}/\overline{GT_0}$, $\overline{RQ}/\overline{GT_1}$</td><td>30, 31</td><td>I/O</td><td>REQUEST/GRANT: pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional with $\overline{RQ}/\overline{GT_0}$ having higher priority than $\overline{RQ}/\overline{GT_1}$. $\overline{RQ}/\overline{GT}$ has an internal pull-up resistor so may be left unconnected. The request/grant sequence is as follows (see timing diagram):
1. A pulse of 1 CLK wide from another local bus master indicates a local bus request ("hold") to the 80C86A (pulse 1).
2. During a T_4 or T_1 clock cycle, a pulse 1 CLK wide from the 80C86A to the requesting master (pulse 2), indicates that the 80C86A has allowed the local bus to float and that it will enter the "hold acknowledge" state at the next CLK. The CPU's bus interface unit is disconnected logically from the local bus during "hold acknowledge."
3. A pulse 1 CLK wide from the requesting master indicates to the 80C86A (pulse 3) that the "hold" request is about to end and that 80C86A can reclaim the local bus at the next CLK.
Each master-master exchange of the local bus is a sequence of 3 pulses. There must be one dead CLK cycle after each bus exchange. Pulses are active LOW.
If the request is made while the CPU is performing a memory cycle, it will release the local bus during T_4 of the cycle when all the following conditions are met:
1. Request occurs on or before T_2.
2. Current cycle is not the low byte of a word (on an odd address).
3. Current cycle is not the first acknowledge of an interrupt acknowledge sequence.
4. A locked instruction is not currently executing.</td></tr>
</table>

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
			If the local bus is idle when the request is made the two possible events will follow: 1. Local bus will be released during the next clock. 2. A memory cycle will start within 3 clocks. Now the four rules for a currently active memory cycle apply with condition number 1 already satisfied.
$\overline{LOCK}$	29	O	**$\overline{LOCK}$:** output indicates that other system bus masters are not to gain control of the system bus while $\overline{LOCK}$ is active LOW. The $\overline{LOCK}$ signal is activated by the "LOCK" prefix instruction and remains active until the completion of the next instruction. This signal is active LOW, and floats to 3-state OFF(1) in "hold acknowledge."
QS_1, QS_0	24, 25	O	**QUEUE STATUS:** The queue status is valid during the CLK cycle after which the queue operation is performed. QS_1 and QS_0 provide status to allow external tracking of the internal 80C86A instruction queue.

QS_1	QS_0	Characteristics
0 (LOW)	0	No Operation
0	1	First Byte of Op Code from Queue
1 (HIGH)	0	Empty the Queue
1	1	Subsequent Byte from Queue

The following pin function descriptions are for the 80C86A in minimum mode (i.e., MN/$\overline{MX}$ = V_{CC}). Only the pin functions which are unique to minimum mode are described; all other pin functions are described above.

M/$\overline{IO}$	28	O	**STATUS LINE:** logically equivalent to S_2 in the maximum mode. It is used to distinguish a memory access from an I/O access. M/$\overline{IO}$ becomes valid in the T_4 preceding a bus cycle and remains valid until the final T_4 of the cycle (M = HIGH, IO = LOW). M/$\overline{IO}$ floats to 3-state OFF(1) in local bus "hold acknowledge."
$\overline{WR}$	29	O	**WRITE:** indicates that the processor is performing a write memory or write I/O cycle, depending on the state of the M/$\overline{IO}$ signal. $\overline{WR}$ is active for T_2, T_3 and T_W of any write cycle. It is active LOW, and floats to 3-state OFF(1) in local bus "hold acknowledge."
$\overline{INTA}$	24	O	$\overline{INTA}$ is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T_2, T_3 and T_W of each interrupt acknowledge cycle.
ALE	25	O	**ADDRESS LATCH ENABLE:** provided by the processor to latch the address into an address latch. It is a HIGH pulse active during T_1 of any bus cycle. Note that ALE is never floated.
DT/$\overline{R}$	27	O	**DATA TRANSMIT/RECEIVE:** needed in minimum system that desires to use a data bus transceiver. It is used to control the direction of data flow through the transceiver. Logically DT/$\overline{R}$ is equivalent to $\overline{S_1}$ in the maximum mode, and its timing is the same as for M/$\overline{IO}$. (T = HIGH, R = LOW.) This signal floats to 3-state OFF(1) in local bus "hold acknowledge."

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
$\overline{DEN}$	26	O	**DATA ENABLE:** provided as an output enable for the transceiver in a minimum system which uses the transceiver. $\overline{DEN}$ is active LOW during each memory and I/O access and for INTA cycles. For a read or $\overline{INTA}$ cycle it is active from the middle of T_2 until the middle of T_4, while for a write cycle it is active from the beginning of T_2 until the middle of T_4. $\overline{DEN}$ floats to 3-state OFF(1) in local bus "hold acknowledge."
HOLD, HLDA	31, 30	I/O	**HOLD:** indicates that another master is requesting a local bus "hold." To be acknowledged, HOLD must be active HIGH. The processor receiving the "hold" request will issue HLDA (HIGH) as an acknowledgement in the middle of a T_4 or T_i clock cycle. Simultaneous with the issuance of HLDA the processor will float the local bus and control lines. After HOLD is detected as being LOW, the processor will LOWer the HLDA, and when the processor needs to run another cycle, it will again drive the local bus and control lines. The same rules as for $\overline{RQ}/\overline{GT}$ apply regarding when the local bus will be released. HOLD is not an asynchronous input. External synchronization should be provided if the system cannot otherwise guarantee the setup time.

NOTE:
1. See the section on Bus Hold Circuitry.

FUNCTIONAL DESCRIPTION

STATIC OPERATION

All 80C86A circuitry is of static design. Internal registers, counters and latches are static and require no refresh as with dynamic circuit design. This eliminates the minimum operating frequency restriction placed on other microprocessors. The CMOS 80C86A can operate from DC to the appropriate upper frequency limit. The processor clock may be stopped in either state (high/low) and held there indefinitely. This type of operation is especially useful for system debug or power critical applications.

The 80C86A can be single stepped using only the CPU clock. This state can be maintained as long as is necessary. Single step clock operation allows simple interface circuitry to provide critical information for bringing up your system.

Static design also allows very low frequency operation. In a power critical situation, this can provide extremely low power operation since 80C86A power dissipation is directly related to operating frequency. As the system frequency is reduced, so is the operating power until, ultimately, at a DC input frequency, the 80C86A power requirement is the standby current.

INTERNAL ARCHITECTURE

The internal functions of the 80C86A processor are partitioned logically into two processing units. The first is the Bus Interface Unit (BIU) and the second is the Execution Unit (EU) as shown in the block diagram of Figure 1.

These units can interact directly but for the most part perform as separate asynchronous operational processors. The bus interface unit provides the functions related to instruction fetching and queuing, operand fetch and store, and address relocation. This unit also provides the basic bus control. The overlap of instruction pre-fetching provided by this unit serves to increase processor performance through improved bus bandwidth utilization. Up to 6 bytes of the instruction stream can be queued while waiting for decoding and execution.

The instruction stream queuing mechanism allows the BIU to keep the memory utilized very efficiently. Whenever there is space for at least 2 bytes in the queue, the BIU will attempt a word fetch memory cycle. This greatly reduces "dead time" on the memory bus. The queue acts as a First-In-First Out (FIFO) buffer, from which the EU extracts instruction bytes as required. If the queue is empty (following a branch instruction, for example), the first byte into the queue immediately becomes available to the EU.

Memory Reference Need	Segment Register Used	Segment Selection Rule
Instructions	CODE (CS)	Automatic with all instruction prefetch.
Stack	STACK (SS)	All stack pushes and pops. Memory references relative to BP base register except data references.
Local Data	DATA (DS)	Data references when: relative to stack, destination of string operation, or explicitly overridden.
External (Global) Data	EXTRA (ES)	Destination of string operations: Explicitly selected using a segment override.

The execution units receives pre-fetched instructions from the BIU queue and provides un-relocated operand addresses to the BIU. Memory operands are passed through the BIU for processing by the EU, which passes results to the BIU for storage. See the Instruction Set description for further register set and architectural descriptions.

MEMORY ORGANIZATION

The processor provides a 20-bit address to memory which locates the byte being referenced. The memory is organized as a linear array of up to 1 million bytes, addressed as 00000(H) to FFFFF(H). The memory is logically divided into code, data, extra data, and stack segments of up to 64k bytes each, with each segment falling on 16-byte boundaries. (See Figure 3a.)

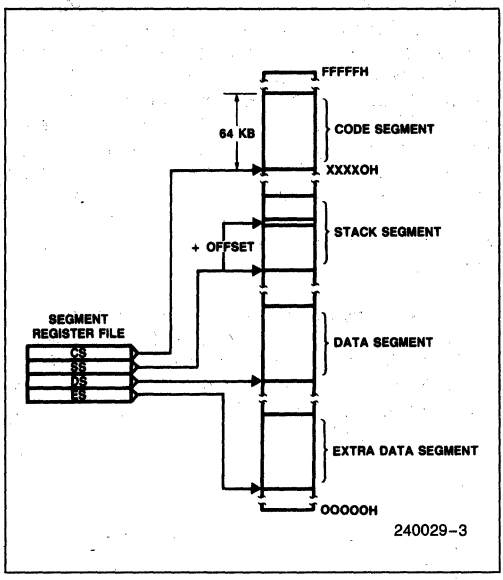


Figure 3a. Memory Organization

All memory references are made relative to base addresses contained in high speed segment registers. The segment types were chosen based on the addressing needs of programs. The segment register to be selected is automatically chosen according to the rules of the following table. All information in one segment type share the same logical attributes (e.g. code or data). By structuring memory into relocatable areas of similar characteristics and by automatically selecting segment registers, programs are shorter, faster, and more structured.

Word (16-bit) operands can be located on even or odd address boundaries and are thus not constrained to even boundaries as is the case in many 16-bit computers. For address and data operands, the least significant byte of the word is stored in the lower valued address location and the most significant byte in the next higher address location. The BIU automatically performs the proper number of memory accesses, one if the word operand is on an even byte boundary and two if it is on an odd byte boundary. Except for the performance penalty, this double access is transparent to the software. This performance penalty does not occur for instruction fetches, only word operands.

Physically, the memory is organized as a high bank (D_{15}–D_8) and a low bank (D_7–D_0) of 512k 8-bit bytes addressed in parallel by the processor's address lines.

A_{19}–A_1. Byte data with even addresses is transferred on the D_7-D_0 bus lines while odd addressed byte data (A_0 HIGH) is transferred on the D_{15}–D_8 bus lines. The processor provides two enable signals, $\overline{BHE}$ and A_0, to selectively allow reading from or writing into either an odd byte location, even byte location, or both. The instruction stream is fetched from memory as words and is addressed internally by the processor to the byte level as necessary.

In referencing word data the BIU requires one or two memory cycles depending on whether or not the starting byte of the word is on an even or odd address, respectively. Consequently, in referencing

word operands performance can be optimized by locating data on even address boundaries. This is an especially useful technique for using the stack, since odd address references to the stack may adversely affect the context switching time for interrupt processing or task multiplexing.

Certain locations in memory are reserved for specific CPU operations (see Figure 3b.) Locations from address FFFF0H through FFFFFH are reserved for operations including a jump to the initial program loading routine. Following RESET, the CPU will always begin execution at location FFFF0H where the jump must be. Locations 00000H through 003FFH are reserved for interrupt operations. Each of the 256 possible interrupt types has its service routine pointed to by a 4-byte pointer element consisting of a 16-bit segment address and a 16-bit offset address. The pointer elements are assumed to have been stored at the respective places in reserved memory prior to occurrence of interrupts.

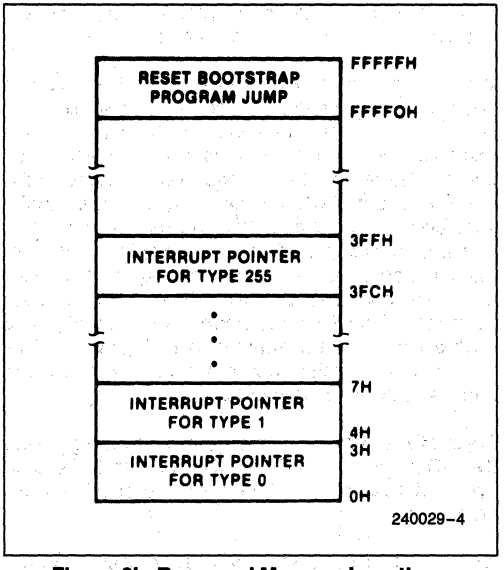


Figure 3b. Reserved Memory Locations

MINIMUM AND MAXIMUM MODES

The requirements for supporting minimum and maximum 80C86A systems are sufficiently different that they cannot be done efficiently with 40 uniquely defined pins. Consequently, the 80C86A is equipped with a strap pin (MN/$\overline{MX}$) which defines the system configuration. The definition of a certain subset of the pins changes dependent on the condition of the strap pin. When MN/$\overline{MX}$ pin is strapped to GND, the 80C86A treats pins 24 through 31 in maximum mode. An 82C88 bus controller interprets status information coded into $\overline{S}_0$, $\overline{S}_1$, $\overline{S}_2$ to generate bus timing and control signals compatible with the MULTIBUS® architecture. When the MN/$\overline{MX}$ pin is strapped to V_{CC}, the 80C86A generates bus control signals itself on pins 24 through 31, as shown in parentheses in Figure 2. Examples of minimum mode and maximum mode systems are shown in Figure 4.

BUS OPERATION

The 80C86A has a combined address and data bus commonly referred to as a time multiplexed bus. This technique provides the most efficient use of pins on the processor. This "local bus" can be buffered directly and used throughout the system with address latching provided on memory and I/O modules. In addition, the bus can also be demultiplexed at the processor with a single set of address latches if a standard non-multiplexed bus is desired for the system.

Each processor bus cycle consists of at least four CLK cycles. These are referred to as T_1, T_2, T_3 and T_4 (see Figure 5). The address is emitted from the processor during T_1 and data transfer occurs on the bus during T_3 and T_4. T_2 is used primarily for changing the direction of the bus during read operations. In the event that a "NOT READY" indication is given by the addressed device, "Wait" states (T_W) are inserted between T_3 and T_4. Each inserted "Wait" state is of the same duration as a CLK cycle. Periods can occur between 80C86A bus cycles. These are referred to as "Idle" states (T_1) or inactive CLK cycles. The processor uses these cycles for internal housekeeping.

During T_1 of any bus cycle the ALE (Address Latch Enable) signal is emitted (by either the processor or the 82C88 bus controller, depending on the MN/$\overline{MX}$ strap). At the trailing edge of this pulse, a valid address and certain status information for the cycle may be latched.

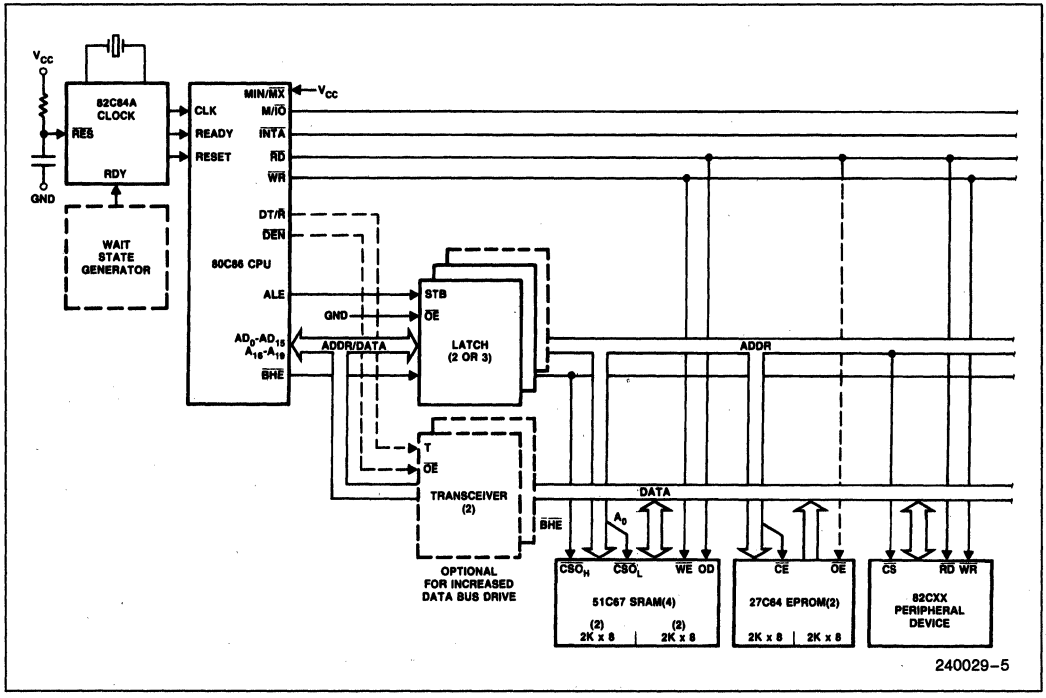


Figure 4a. Minimum Mode iAPX 80C86A Typical Configuration

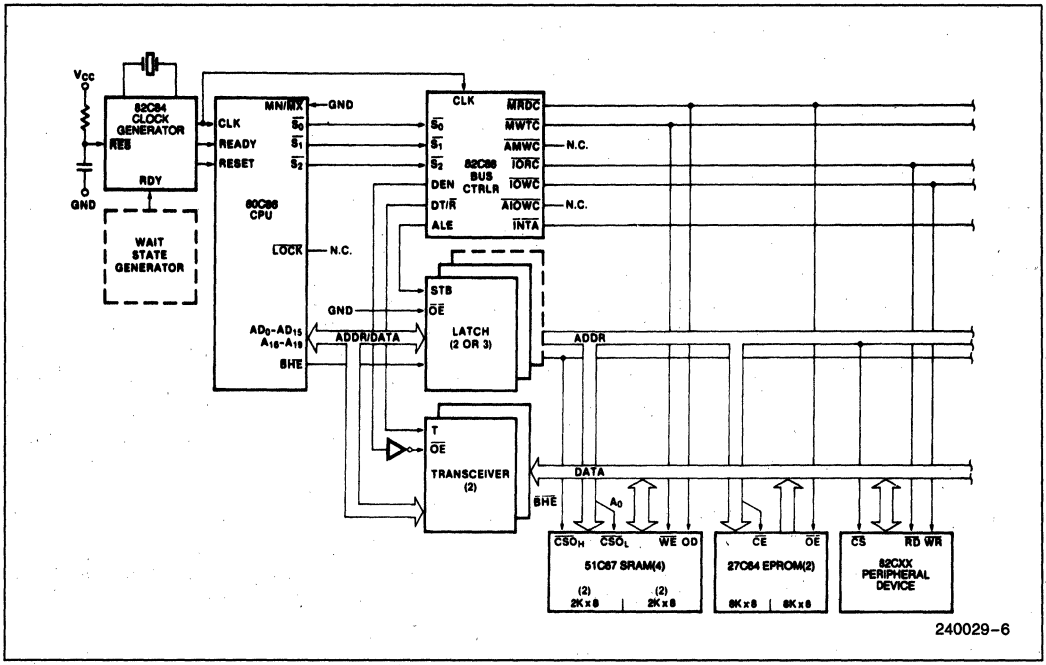


Figure 4b. Maximum Mode 80C86A Typical Configuration

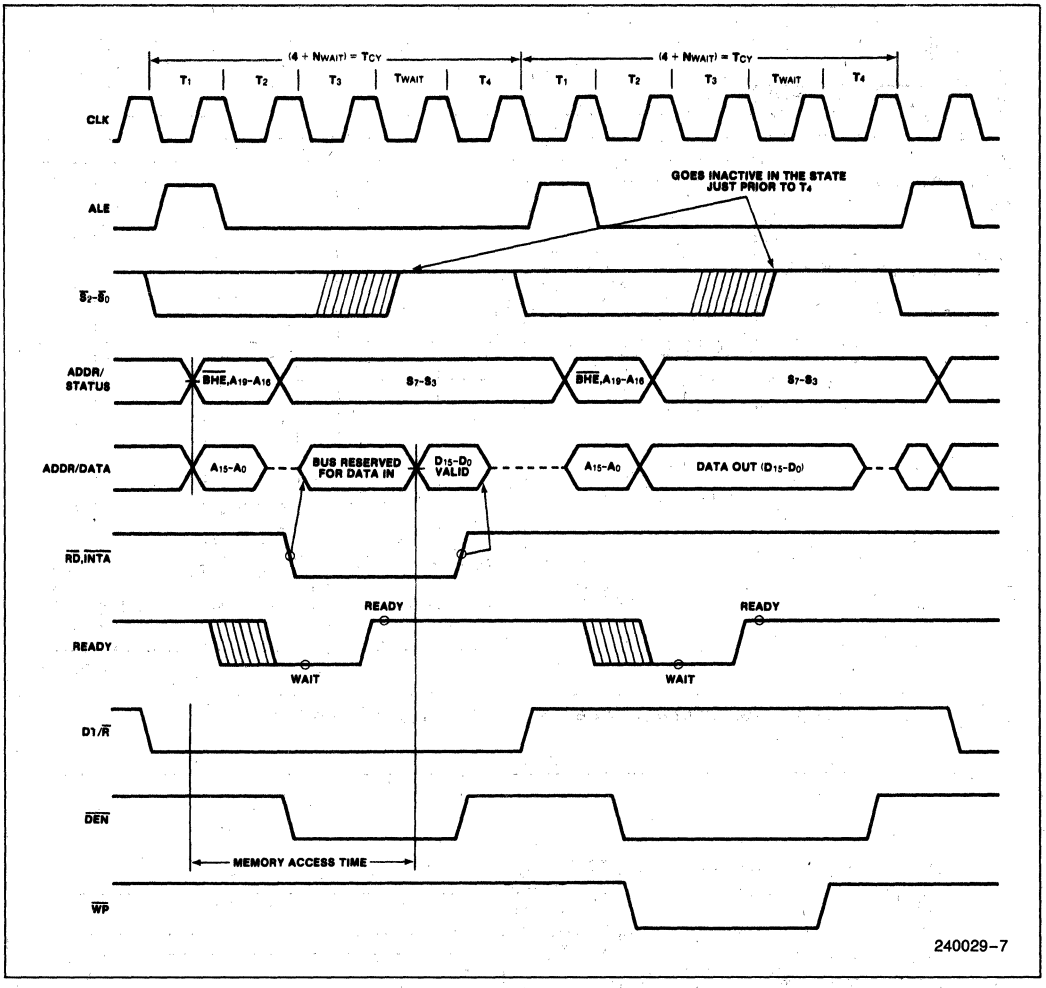


Figure 5. Basic System Timing

Status bits $\overline{S}_0$, $\overline{S}_1$, and $\overline{S}_2$ are used, in maximum mode, by the bus controller to identify the type of bus transaction according to the following table:

$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	Characteristics
0 (LOW)	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1 (HIGH)	0	0	Instruction Fetch
1	0	1	Read Data from Memory
1	1	0	Write Data to Memory
1	1	1	Passive (no bus cycle)

Status bits S_3 through S_7 are multiplexed with high-order address bits and the $\overline{BHE}$ signal, and are therefore valid during T_2 through T_4. S_3 and S_4 indicate which segment register (see Instruction Set description) was used for this bus cycle in forming the address, according to the following table:

S_4	S_3	Characteristics
0 (LOW)	0	Alternate Data (extra segment)
0	1	Stack
1 (HIGH)	0	Code or None
1	1	Data

S_5 is a reflection of the PSW interrupt enable bit. $S_6 = 0$ and S_7 is a spare status pin.

I/O ADDRESSING

In the 80C86A, I/O operations can address up to a maximum of 64k I/O byte registers or 32k I/O word registers. The I/O address appears in the same format as the memory address on bus lines A_{15}–A_0. The address lines A_{19}–A_{16} are zero in I/O operations. The variable I/O instructions which use register DX as a pointer have full address capability while the direct I/O instructions directly address one or two of the 256 I/O byte locations in page 0 of the I/O address space.

I/O ports are addressed in the same manner as memory locations. Even addressed bytes are transferred on the D_7–D_0 bus lines and odd addressed bytes on D_{15}–D_8. Care must be taken to assure that each register within an 8-bit peripheral located on the lower portion of the bus be addressed as even.

EXTERNAL INTERFACE

PROCESSOR RESET AND INITIALIZATION

Processor initialization or start up is accomplished with activation (HIGH) of the RESET pin. The 80C86A RESET is required to be HIGH for four or more CLK cycles. The 80C86A will terminate operations on the high-going edge of RESET and will remain dormant as long as RESET is HIGH. The low-going transition of RESET triggers an internal reset sequence for approximately 7 CLK cycles. After this interval the 80C86A operates normally beginning with the instruction in absolute location FFFF0H (see Figure 3b). The details of this operation are specified in the Instruction Set description of the MCS®-86 Family User's Manual. The RESET input is internally synchronized to the processor clock. At initialization the HIGH-to-LOW transition of RESET must occur no sooner than 50 μs after power-up, to allow complete initialization of the 80C86A.

NMI asserted prior to the 2nd clock after the end of RESET will not be honored. If NMI is asserted after that point and during the internal reset sequence, the processor may execute one instruction before responding to the interrupt. A hold request active immediately after RESET will be honored before the first instruction fetch.

All 3-state outputs float to 3-state OFF(1) during RESET. Status is active in the idle state for the first clock after RESET becomes active and then floats to 3-state OFF(1). ALE and HLDA are driven low.

NOTE:
1. See the section on Bus Hold Circuitry.

BUS HOLD CIRCUITRY

To avoid high current conditions caused by floating inputs to CMOS devices and eliminate the need for pull-up/down resistors, "bus-hold" circuitry has been used on the 80C86A pins 2–16, 26–32, and 34–39 (Figures 6a, 6b). These circuits will maintain the last valid logic state if no driving source is present (i.e. an unconnected pin or a driving source which goes to a high impedance state). To overdrive the "bus hold" circuits, an external driver must be capable of supplying 350 μA minimum sink or source current at valid input voltage levels. Since this "bus hold" circuitry is active and not a "resistive" type element, the associated power supply current is negligible and power dissipation is significantly reduced when compared to the use of passive pull-up resistors.

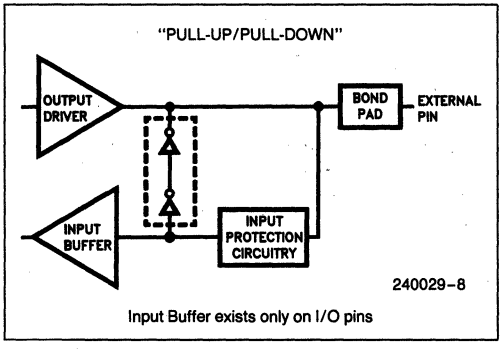


Figure 6a. Bus hold circuitry pin 2-16, 34-39.

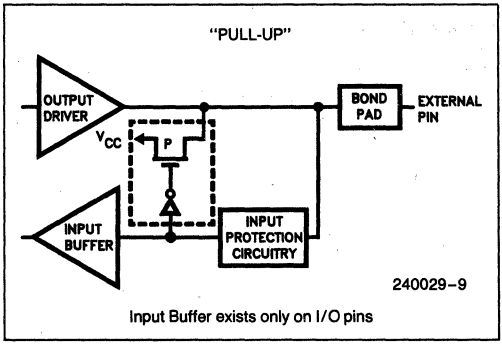


Figure 6b. Bus hold circuitry pin 26-32.

INTERRUPT OPERATIONS

Interrupt operations fall into two classes; software or hardware initiated. The software initiated interrupts and software aspects of hardware interrupts are specified in the Instruction Set description. Hardware interrupts can be classified as non-maskable or maskable.

Interrupts result in a transfer of control to a new program location. A 256-element table containing address pointers to the interrupt service program locations resides in absolute locations 0 through 3FFH (see Figure 3b), which are reserved for this purpose. Each element in the table is 4 bytes in size and corresponds to an interrupt "type". An interrupting device supplies an 8-bit type number, during the interrupt acknowledge sequence, which is used to "vector" through the appropriate element to the new interrupt service program location.

NON-MASKABLE INTERRUPT (NMI)

The processor provides a single non-maskable interrupt pin (NMI) which has higher priority than the maskable interrupt request pin (INTR). A typical use would be to activate a power failure routine. The NMI is edge-triggered on a LOW-to-HIGH transition. The activation of this pin causes a type 2 interrupt. (See Instruction Set description.) NMI is required to have a duration in the HIGH state of greater than two CLK cycles, but is not required to be synchronized to the clock. Any high-going transition of NMI is latched on-chip and will be serviced at the end of the current instruction or between whole moves of a block-type instruction. Worst case response to NMI would be for multiply, divide and variable shift instructions. There is no specification on the occurrence of the low-going edge; it may occur before, during, or after the servicing of NMI. Another high-going edge triggers another response if it occurs after the start of the NMI procedure. The signal must be free of logical spikes in general and be free of bounces on the low-going edge to avoid triggering extraneous responses.

MASKABLE INTERRUPT (INTR)

The 80C86A provides a single interrupt request input (INTR) which can be masked internally by software with the resetting of the interrupt enable FLAG status bit. The interrupt request signal is level triggered. It is internally synchronized during each clock cycle on the high-going edge of CLK. To be responded to, INTR must be present (HIGH) during the clock period preceding the end of the current instruction or the end of a whole move for a block-type instruction. During the interrupt response sequence further interrupts are disabled. The enable bit is reset as part of the response to any interrupt (INTR, NMI, software interrupt or single-step), although the FLAGS register which is automatically pushed onto the stack reflects the state of the processor prior to the interrupt. Until the old FLAGS register is restored the enable bit will be zero unless specifically set by an instruction.

During the response sequence (Figure 7) the processor executes two successive (back-to-back) interrupt acknowledge cycles. The 80C86A emits the LOCK signal from T_2 of the first bus cycle until T_2 of the second. A local bus "hold" request will not be honored until the end of the second bus cycle. In the second bus cycle a byte is fetched from the external interrupt system (e.g., 82C59 PIC) which identifies the source (type) of the interrupt. This byte is multiplied by four and used as a pointer into the interrupt vector lookup table. An INTR signal left HIGH will be continually responded to within the limitations of the enable bit and sample period. The INTERRUPT RETURN instruction includes a FLAGS pop which returns the status of the original interrupt enable bit when it restores the FLAGS.

HALT

When a software "HALT" instruction is executed the processor indicates that it is entering the "HALT" state in one of two ways depending upon which mode is strapped. In minimum mode, the processor issues one ALE with no qualifying bus control signals. In Maximum Mode, the processor issues appropriate HALT status on $\overline{S}_2$, $\overline{S}_1$ and $\overline{S}_0$ and the 82C88 bus controller issues one ALE. The 80C86A will not leave the "HALT" state when a local bus "hold" is entered while in "HALT". In this case, the processor reissues the HALT indicator. An interrupt request or RESET will force the 80C86A out of the "HALT" state.

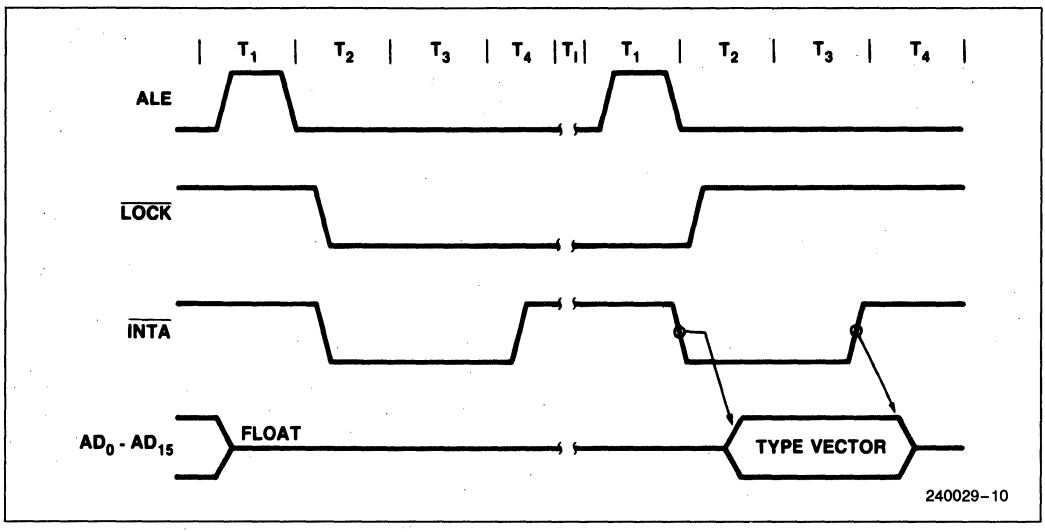


Figure 7. Interrupt Acknowledge Sequence

READ/MODIFY/WRITE (SEMAPHORE) OPERATIONS VIA LOCK

The $\overline{LOCK}$ status information is provided by the processor when directly consecutive bus cycles are required during the execution of an instruction. This provides the processor with the capability of performing read/modify/write operations on memory (via the Exchange Register With Memory instruction, for example) without the possibility of another system bus master receiving intervening memory cycles. This is useful in mutliprocessor system configurations to accomplish "test and set lock" operations. The $\overline{LOCK}$ signal is activated (forced LOW) in the clock cycle following the one in which the software "LOCK" prefix instruction is decoded by the EU. It is deactivated at the end of the last bus cycle of the instruction following the "LOCK" prefix instruction. While $\overline{LOCK}$ is active a request on a RQ/GT pin will be recorded and then honored at the end of the LOCK.

EXTERNAL SYNCHRONIZATION VIA TEST

As an alternative to the interrupts and general I/O capabilities, the 80C86A provides a single software-testable input known as the $\overline{TEST}$ signal. At any time the program may execute a WAIT instruction. If at that time the $\overline{TEST}$ signal is inactive (HIGH), program execution becomes suspended while the processor waits for $\overline{TEST}$ to become active. It must remain active for at least 5 CLK cycles. The WAIT instruction is re-executed repeatedly until that time. This activity does not consume bus cycles. The processor remains in an idle state while waiting. All 80C86A drivers go to 3-state OFF if bus "Hold" is entered. If interrupts are enabled, they may occur while the processor is waiting. When this occurs the processor fetches the WAIT instruction one extra time, processes the interrupt, and then re-fetches and re-executes the WAIT instruction upon returning from the interrupt.

BASIC SYSTEM TIMING

Typical system configurations for the processor operating in minimum mode and in maximum mode are shown in Figures 4a and 4b, respectively. In minimum mode, the MN/$\overline{MX}$ pin is strapped to V_{CC} and the processor emits bus control signals in a manner similar to the 8085. In maximum mode, the MN/$\overline{MX}$ pin is strapped to V_{SS} and the processor emits coded status information which the 82C88 bus controller uses to generate MULTIBUS compatible bus control signals. Figure 5 illustrates the signal timing relationships.

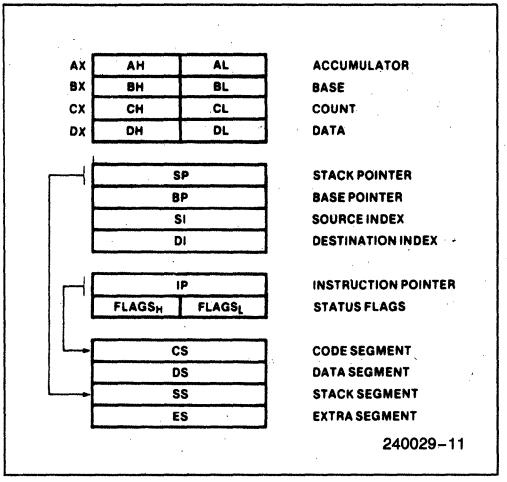


Figure 8. 80C86A Register Model

SYSTEM TIMING—MINIMUM SYSTEM

The read cycle begins in T_1 with the assertion of the Address Latch Enable (ALE) signal. The trailing (low-going) edge of this signal is used to latch the address information, which is valid on the local bus at this time, into a latch. The $\overline{BHE}$ and A_0 signals address the low, high, or both bytes. From T_1 to T_4 the M/$\overline{IO}$ signal indicates a memory or I/O operation. At T_2 the address is removed from the local bus and the bus goes to a high impedance state. The read control signal is also asserted at T_2. The read ($\overline{RD}$) signal causes the addressed device to enable its data bus drivers to the local bus. Some time later valid data will be available on the bus and the addressed device will drive the READY line HIGH. When the processor returns the read signal to a HIGH level, the addressed device will again 3-state its bus drivers. If a transceiver is required to buffer the 80C86A local bus, signals DT/$\overline{R}$ and $\overline{DEN}$ are provided by the 80C86A.

A write cycle also begins with the assertion of ALE and the emission of the address. The M/$\overline{IO}$ signal is again asserted to indicate a memory or I/O write operation. In the T_2 immediately following the address emission the processor emits the data to be written into the addressed location. This data remains valid until the middle of T_4. During T_2, T_3, and T_W the processor asserts the write control signal. The write ($\overline{WR}$) signal becomes active at the beginning of T_2 as opposed to the read which is delayed somewhat into T_2 to provide time for the bus to float.

The $\overline{BHE}$ and A_0 signals are used to select the proper byte(s) of the memory/IO word to be read or written according to the following table:

$\overline{BHE}$	A0	Characteristics
0	0	Whole word
0	1	Upper byte from/ to odd address
1	0	Lower byte from/ to even address
1	1	None

I/O ports are addressed in the same manner as memory location. Even addressed bytes are transferred on the D_7–D_0 bus lines and odd addressed bytes on D_{15}–D_8.

The basic difference between the interrupt acknowledge cycle and a read cycle is that the interrupt acknowledge signal ($\overline{INTA}$) is asserted in place of the read ($\overline{RD}$) signal and the address bus is floated. (See Figure 7.) In the second of two successive INTA cycles, a byte of information is read from bus lines D_7–D_0 as supplied by the interrupt system logic (i.e., 82C59A Priority Interrupt Controller). This byte identifies the source (type) of the interrupt. It is multiplied by four and used as a pointer into an interrupt vector lookup table, as described earlier.

BUS TIMING—MEDIUM SIZE SYSTEMS

For medium size systems the MN/$\overline{MX}$ pin is connected to V_{SS} and the 82C88 Bus Controller is added to the system as well as a latch for latching the system address, and a transceiver to allow for bus loading greater than the 80C86A is capable of handling. Signals ALE, DEN, and DT/$\overline{R}$ are generated by the 82C88 instead of the processor in this configuration although their timing remains relatively the same. The 80C86A status outputs ($\overline{S}_2$, $\overline{S}_1$, and $\overline{S}_0$) provide type-of-cycle information and become 82C88 inputs. This bus cycle information specifies read (code, data, or I/O), write (data or I/O), interrupt acknowledge, or software halt. The 82C88 thus issues control signals specifying memory read or write, I/O read or write, or interrupt acknowledge. The 82C88 provides two types of write strobes, normal and advanced, to be applied as required. The normal write strobes have data valid at the leading edge of write. The advanced write strobes have the same timing as read strobes, and hence data isn't valid at the leading edge of write. The transceiver receives the usual T and OE inputs from the 82C88 DT/$\overline{R}$ and DEN.

The pointer into the interrupt vector table, which is passed during the second INTA cycle, can derive from an 82C59A located on either the local bus or the system bus. If the master 82C59A Priority Interrupt Controller is positioned on the local bus, a TTL gate is required to disable the transceiver when reading from the master 82C59A during the interrupt acknowledge sequence and software "poll".

ABSOLUTE MAXIMUM RATINGS*

Supply Voltage
(With respect to ground) ..... −0.5 to 7.0V
Input Voltage Applied
(w.r.t. ground) ..... −0.5 to V_{CC} + 0.5V
Output Voltage Applied
(w.r.t. ground) ..... −0.5 to V_{CC} + 0.5V
Power Dissipation 1.0W
Storage Temperature ..... −65°C to 150°C
Ambient Temperature Under Bias ..... 0°C to 70°C

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

(T_A = 0°C to 70°C, V_{CC} = 5V ±5%)

Symbol	Parameter	80C86A-2		Units	Test Conditions
		Min	Max		
V_{IL}	Input Low Voltage	−0.5	+0.8	V	
V_{IH}	Input High Voltage (All inputs except clock)	2.0		V	
V_{CH}	Clock Input High Voltage	V_{CC}−0.8		V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.5 mA
V_{OH}	Output High Voltage	3.0 V_{CC}−0.4		V	I_{OH} = −2.5 mA I_{OH} = −100 μA
I_{CC}	Power Supply Current		10 mA/MHz		V_{IL} = GND, V_{IH} = V_{CC}
I_{CCS}	Standby Supply Current		500	μA	V_{IN} = V_{CC} or GND Outputs Unloaded CLK = GND or V_{CC}
I_{LI}	Input Leakage Current		±1.0	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{BHL}	Input Leakage Current (Bus Hold Low)	50	400	μA	V_{IN} = 0.8V (Note 4)
I_{BHH}	Input Leakage Current (Bus Hold High)	−50	−400	μA	V_{IN} = 3.0V (Note 5)
I_{BHLO}	Bus Hold Low Overdrive		600	μA	(Note 2)
I_{BHHO}	Bus Hold High Overdrive		−600	μA	(Note 3)
I_{LO}	Output Leakage Current		±10	μA	V_{OUT} = GND or V_{CC}
C_{IN}	Capacitance of Input Buffer (All inputs except AD_0–AD_{15}, $\overline{RQ}/\overline{GT}$)		5	pF	(Note 1)
C_{IO}	Capacitance of I/O Buffer (AD_0–AD_{15}, $\overline{RQ}/\overline{GT}$)		20	pF	(Note 1)
C_{OUT}	Output Capacitance		15	pF	(Note 1)

NOTES:
1. Characterization conditions are a) Frequency = 1 MHz; b) Unmeasured pins at GND; c) V_{IN} at +5.0V or GND.
2. An external driver must source at least I_{BHLO} to switch this node from LOW to HIGH.
3. An external driver must sink at least I_{BHHO} to switch this node from HIGH to LOW.
4. Test Condition is to lower V_{IN} to GND and then raise V_{IN} to 0.8V on pins 2-16 & 34-39.
5. Test Condition is to raise V_{IN} to V_{CC} and then lower V_{IN} to 3.0V on pins 2-16, 26-32 & 34-39.

A.C. CHARACTERISTICS

(T_A = 0°C to 70°C, V_{CC} = 5V ±5%)

MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

Symbol	Parameter	80C86A-2		Units	Test Conditions
		Min	Max		
TCLCL	CLK Cycle Period	125	D.C.	ns	
TCLCH	CLK Low Time	68		ns	
TCHCL	CLK High Time	44		ns	
TCH1CH2	CLK Rise Time		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10	ns	From 3.5V to 1.0V
TDVCL	Data in Setup Time	20		ns	
TCLDX	Data in Hold Time	10		ns	
TR1VCL	RDY Setup Time into 82C84A (Notes 1, 2)	35		ns	
TCLR1X	RDY Hold Time into 82C84A (Notes 1, 2)	0		ns	
TRYHCH	READY Setup Time into 80C86A	68		ns	
TCHRYX	READY Hold Time into 80C86A	20		ns	
TRYLCL	READY Inactive to CLK (Note 3)	−8		ns	
THVCH	HOLD Setup Time	20		ns	
TINVCH	INTR, NMI, $\overline{\text{TEST}}$ Setup Time (Note 2)	15		ns	
TILIH	Input Rise Time (Except CLK)		15	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK)		15	ns	From 2.0V to 0.8V

A.C. CHARACTERISTICS (Continued)

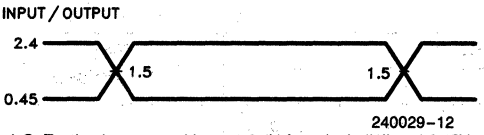
(T_A = 0°C to 70°C, V_{CC} = 5V ±5%)

Timing Responses

Symbol	Parameter	80C86A-2		Units	Test Conditions
		Min	Max		
TCLAV	Address Valid Delay	10	60	ns	
TCLAX	Address Hold Time	10		ns	
TCLAZ	Address Float Delay	TCLAX	50	ns	
TLHLL	ALE Width	TCLCH−10		ns	
TCLLH	ALE Active Delay		50	ns	
TCHLL	ALE Inactive Delay		55	ns	
TLLAX	Address Hold Time to ALE Inactive	TCHCL−10		ns	
TCLDV	Data Valid Delay	10	60	ns	
TCHDX	Data Hold Time	10		ns	
TWHDX	Data Hold Time After WR	TCLCH−30		ns	
TCVCTV	Control Active Delay 1	10	70	ns	
TCHCTV	Control Active Delay 2	10	60	ns	
TCVCTX	Control Inactive Delay	10	70	ns	
TAZRL	Address Float to READ Active	0		ns	
TCLRL	$\overline{RD}$ Active Delay	10	100	ns	
TCLRH	$\overline{RD}$ Inactive Delay	10	80	ns	
TRHAV	$\overline{RD}$ Inactive to Next Address Active	TCLCL−40		ns	
TCLHAV	HLDA Valid Delay	10	100	ns	
TRLRH	$\overline{RD}$ Width	2TCLCL−50		ns	
TWLWH	$\overline{WR}$ Width	2TCLCL−40		ns	
TAVAL	Address Valid to ALE Low	TCLCH−40		ns	
TOLOH	Output Rise Time		15	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time		15	ns	From 2.0V to 0.8V

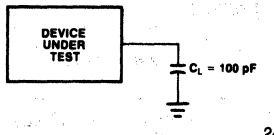
NOTES:
1. Signal at 82C84A shown for reference only. See 82C84A data sheet for the most recent specifications.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T2 state. (8 ns into T3).

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. Testing inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0". Timing measurements are made at 1.5V.

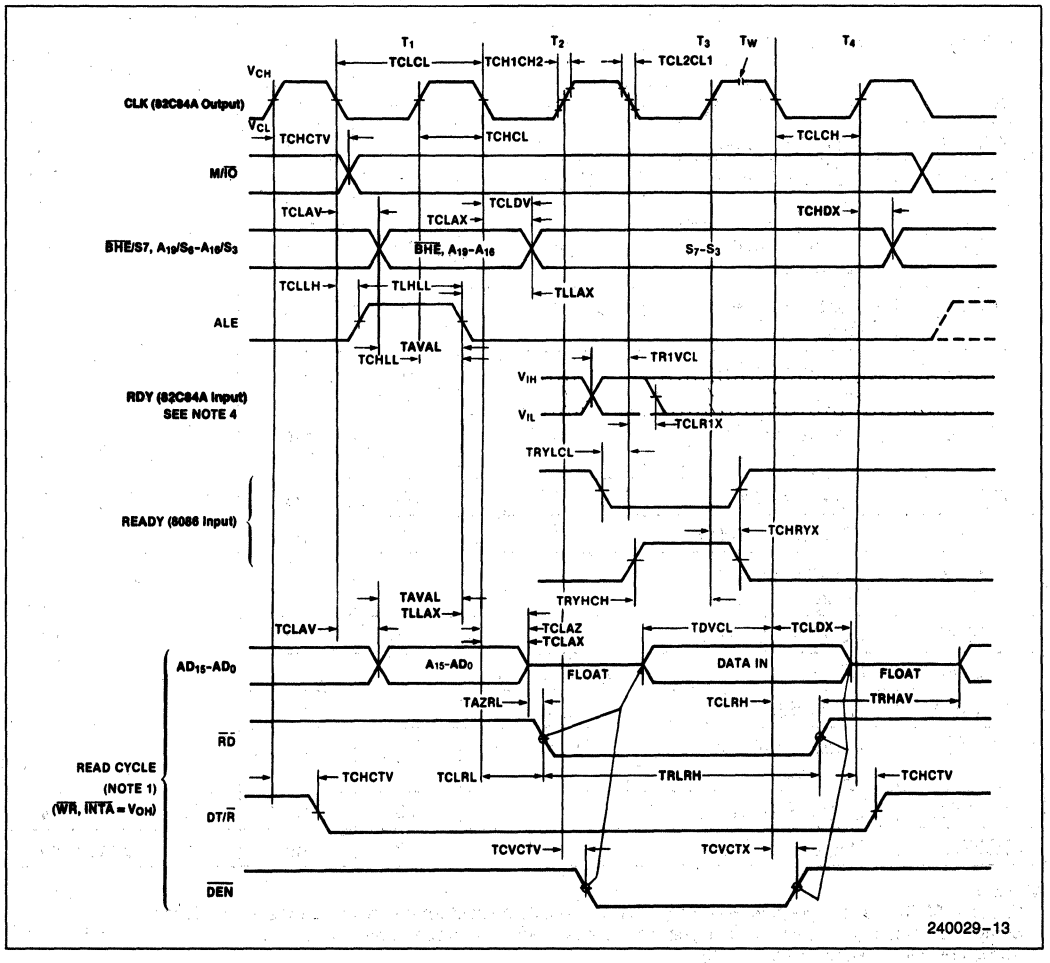
A.C. TESTING LOAD CIRCUIT



C_L Includes Jig Capacitance

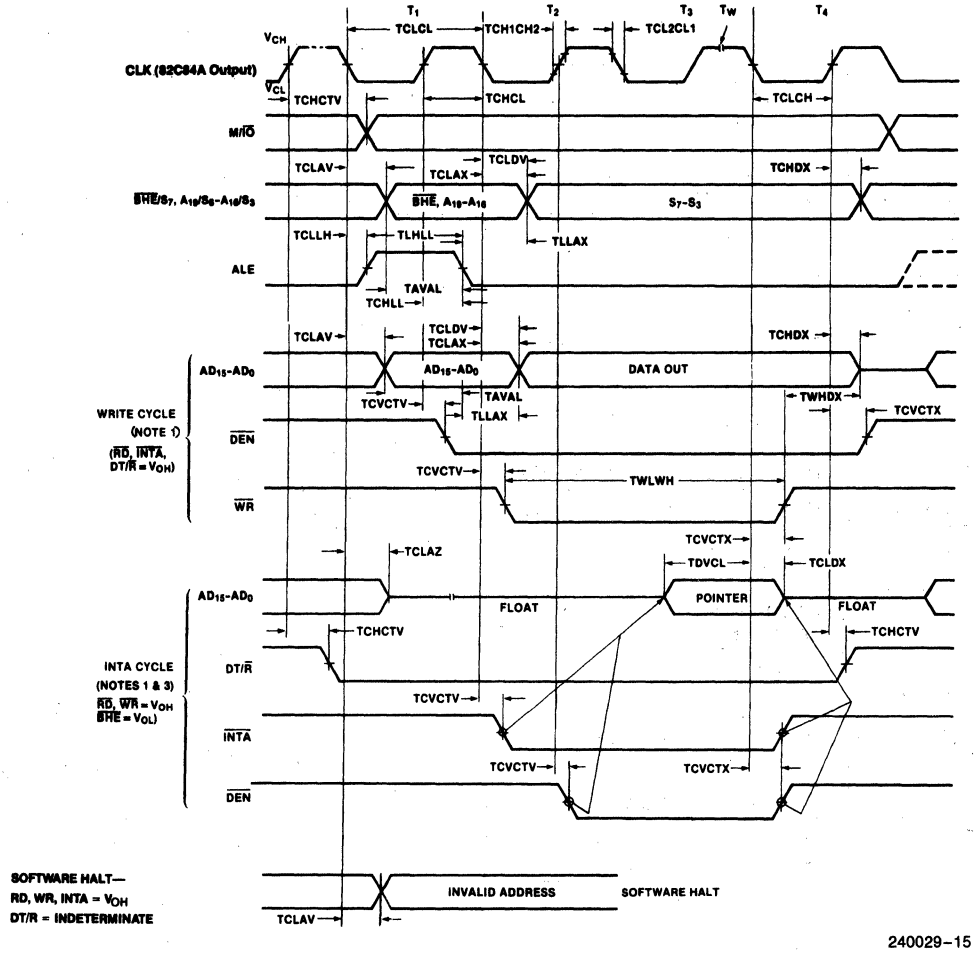
WAVEFORMS

MINIMUM MODE



WAVEFORMS (Continued)

MINIMUM MODE (Continued)



NOTES:
1. All output timing measurements are made at 1.5V unless otherwise noted.
2. RDY is sampled near the end of T_2, T_3, T_W to determine if T_W machines states are to be inserted.
3. Two INTA cycles run back-to-back. The 80C86A local ADDR/DATA BUS is floating during both INTA cycles. Control signals shown for second INTA cycle.
4. Signals at 82C84A are shown for reference only.

A.C. CHARACTERISTICS

MAX MODE SYSTEM (USING 82C88 BUS CONTROLLER) TIMING REQUIREMENTS

Symbol	Parameter	80C86A-2		Units	Test Conditions
		Min	Max		
TCLCL	CLK Cycle Period	125	D.C.	ns	
TCLCH	CLK Low Time	68		ns	
TCHCL	CLK High Time	44		ns	
TCH1CH2	CLK Rise Time		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10	ns	From 3.5V to 1.0V
TDVCL	Data in Setup Time	20		ns	
TCLDX	Data in Hold Time	10		ns	
TR1VCL	RDY Setup Time into 82C84A (Notes 1, 2)	35		ns	
TCLR1X	RDY Hold Time into 82C84A (Notes 1, 2)	0		ns	
TRYHCH	READY Setup Time into 80C86A	68		ns	
TCHRYX	READY Hold Time into 80C86A	20		ns	
TRYLCL	READY Inactive to CLK (Note 4)	−8		ns	
TINVCH	Setup Time for Recognition (INTR, NMI, $\overline{TEST}$) (Note 2)	15		ns	
TGVCH	$\overline{RQ}/\overline{GT}$ Setup Time	15		ns	
TCHGX	$\overline{RQ}$ Hold Time into 80C86A	30		ns	
TILIH	Input Rise Time (Except CLK) (Note 5)		15	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK) (Note 5)		15	ns	From 2.0V to 0.8V

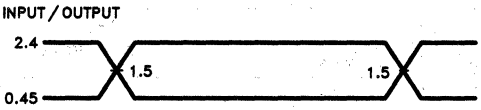
A.C. CHARACTERISTICS (Continued)

TIMING RESPONSES

Symbol	Parameter	80C86A-2		Units	Test Conditions
		Min	Max		
TCLML	Command Active Delay (Note 1)	5	35	ns	
TCLMH	Command Inactive Delay (Note 1)	5	35	ns	
TRYHSH	READY Active to Status Passive (Note 3)		65	ns	
TCHSV	Status Active Delay	10	60	ns	
TCLSH	Status Inactive Delay	10	70	ns	
TCLAV	Address Valid Delay	10	60	ns	
TCLAX	Address Hold Time	10		ns	
TCLAZ	Address Float Delay	TCLAX	50	ns	
TSVLH	Status Valid to ALE High (Note 1)		20	ns	
TSVMCH	Status Valid to MCE High (Note 1)		30	ns	
TCLLH	CLK Low to ALE Valid (Note 1)		20	ns	
TCLMCH	CLK Low to MCE High (Note 1)		25	ns	
TCHLL	ALE Inactive Delay (Note 1)	4	18	ns	
TCLDV	Data Valid Delay	10	60	ns	
TCHDX	Data Hold Time	10		ns	
TCVNV	Control Active Delay (Note 1)	5	45	ns	
TCVNX	Control Inactive Delay (Note 1)	10	45	ns	
TAZRL	Address Float to Read Active	0		ns	
TCLRL	RD Active Delay	10	100	ns	
TCLRH	RD Inactive Delay	10	80	ns	
TRHAV	RD Inactive to Next Address Active	TCLCL−40		ns	
TCHDTL	Direction Control Active Delay (Note 1)		50	ns	
TCHDTH	Direction Control Inactive Delay (Note 1)		30	ns	
TCLGL	GT Active Delay	0	50	ns	
TCLGH	GT Inactive Delay	0	50	ns	
TRLRH	RD Width	2TCLCL−50		ns	
TOLOH	Output Rise Time		15	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time		15	ns	From 2.0V to 0.8V

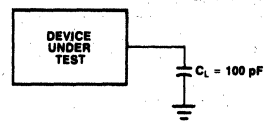
NOTES:
1. Signal at 82C84A or 82C88 shown for reference only. See 82C84A and 82C88 for the most recent specifications.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T3 and wait states.
4. Applies only to T2 state (8 ns into T3).
5. These parameters are characterized and not 100% tested.

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. Testing inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0". Timing measurements are made at 1.5V

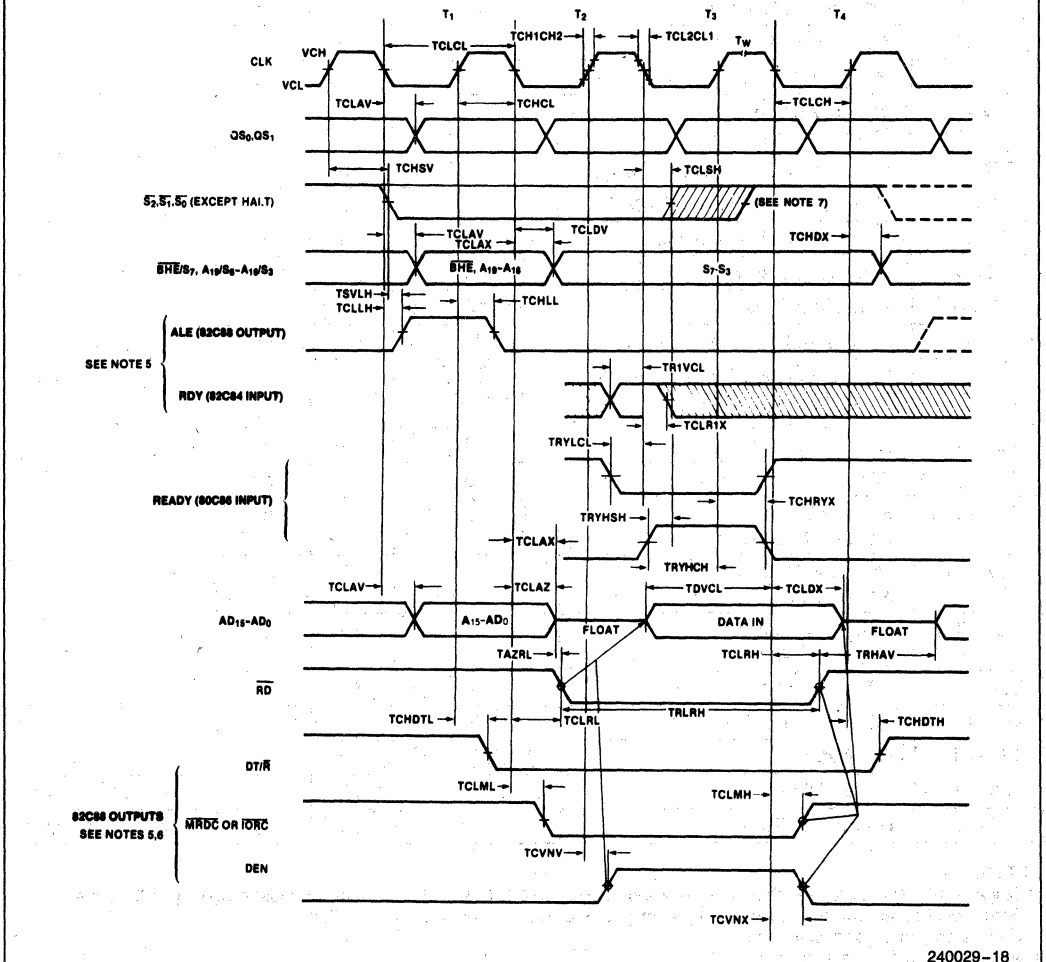
A.C. TESTING LOAD CIRCUIT



C_L Includes Jig Capacitance

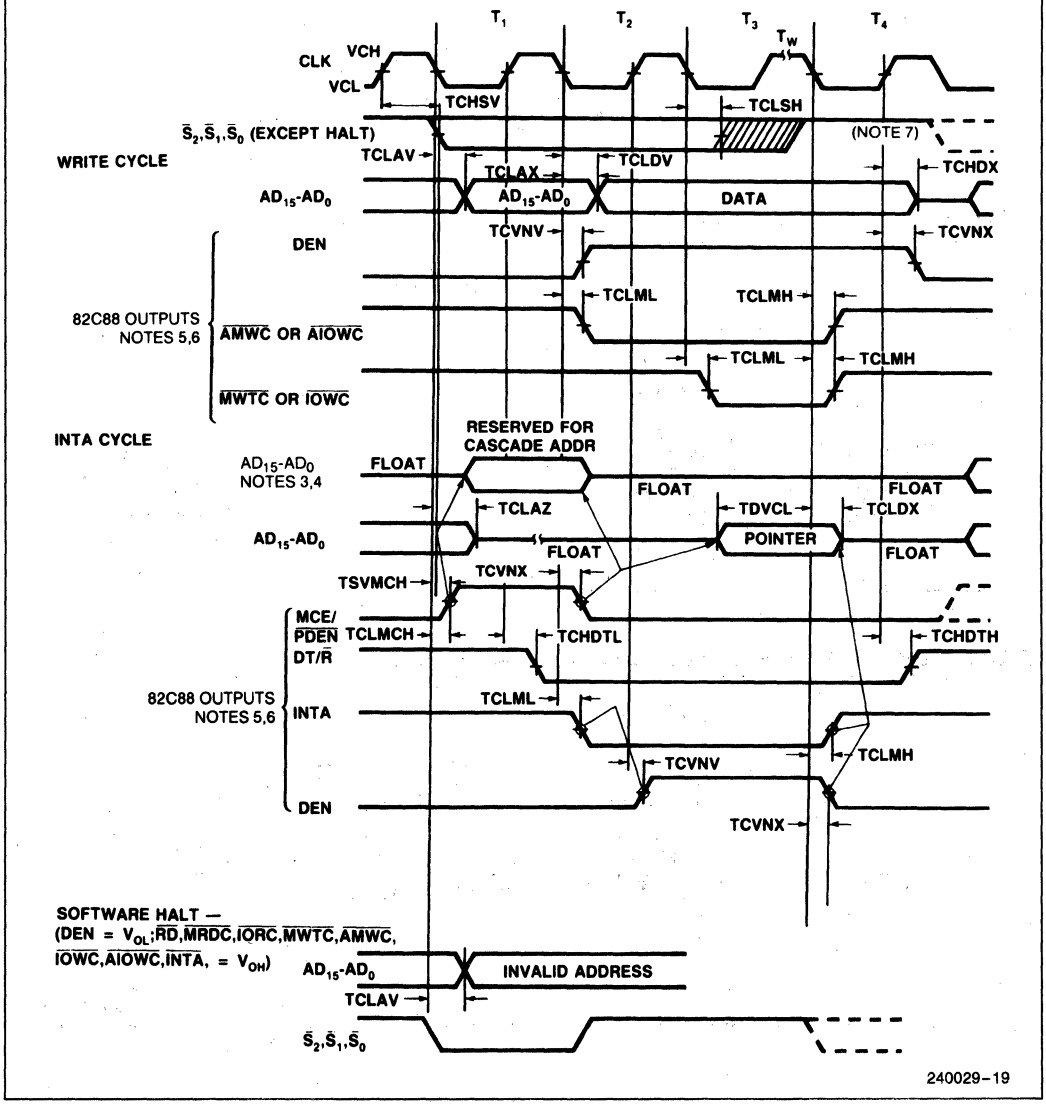
WAVEFORMS

MAXIMUM MODE



WAVEFORMS (Continued)

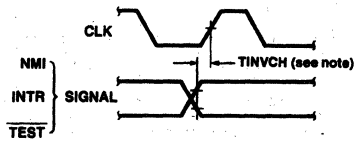
MAXIMUM MODE (Continued)



NOTES:
1. All timing measurements are made at 1.5V unless otherwise noted.
2. RDY is sampled near the end of T_2, T_3, T_W to determine if T_W machines states are to be inserted.
3. Cascade address is valid between first and second INTA cycle.
4. Two INTA cycles run back-to-back. The 80C86A local ADDR/DATA BUS is floating during both INTA cycles. Control for pointer address is shown for second INTA cycle.
5. Signals at 82C84A or 82C88 are shown for reference only.
6. The issuance of the 82C88 command and control signals ($\overline{MRDC}$, $\overline{MWTC}$, $\overline{AMWC}$, $\overline{IORC}$, $\overline{IOWC}$, $\overline{AIOWC}$, $\overline{INTA}$ and DEN) lags the active high 82C88 CEN.
7. Status inactive in state just prior to T_4.

WAVEFORMS (Continued)

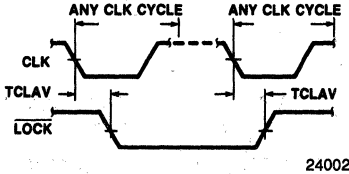
ASYNCHRONOUS SIGNAL RECOGNITION



240029-20

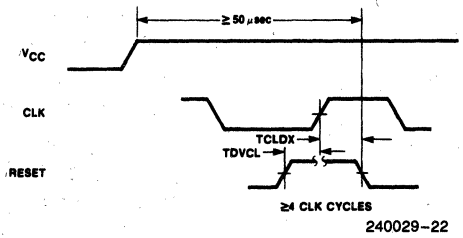
NOTE: Setup requirements for asynchronous signals only to guarantee recognition at next CLK.

BUS LOCK SIGNAL TIMING (MAXIMUM MODE ONLY)



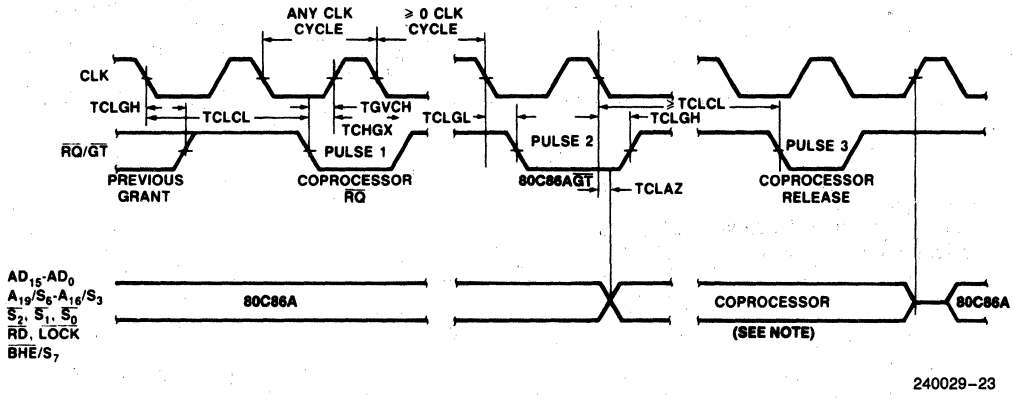
240029-21

RESET TIMING



240029-22

REQUEST/GRANT SEQUENCE TIMING (MAXIMUM MODE ONLY)

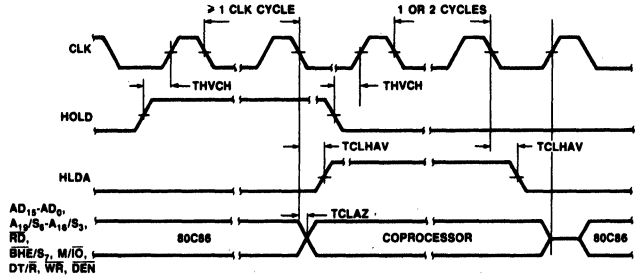


240029-23

NOTE: The coprocessor may not drive the buses outside the region shown without risking contention.

WAVEFORMS (Continued)

HOLD/HOLD ACKNOWLEDGE TIMING (MINIMUM MODE ONLY)



240029-24

Table 2. Instruction Set Summary

Mnemonic and Description	Instruction Code			
DATA TRANSFER				
MOV = Move:	76543210	76543210	76543210	76543210
Register/Memory to/from Register**	100010dw	mod reg r/m		
Immediate to Register/Memory	1100011w	mod 000 r/m	data	data if w = 1
Immediate to Register	1011 w reg	data	data if w = 1	
Memory to Accumulator	1010000w	addr-low	addr-high	
Accumulator to Memory	1010001w	addr-low	addr-high	
Register/Memory to Segment Register**	10001110	mod 0 reg r/m		
Segment Register to Register/Memory	10001100	mod 0 reg r/m		
PUSH = Push:				
Register/Memory	11111111	mod 110 r/m		
Register	01010 reg			
Segment Register	000 reg 110			
POP = Pop:				
Register/Memory	10001111	mod 000 r/m		
Register	01011 reg			
Segment Register	000 reg 111			
XCHG = Exchange:				
Register/Memory with Register	1000011w	mod reg r/m		
Register with Accumulator	10010 reg			
IN = Input from:				
Fixed Port	1110010w	port		
Variable Port	1110110w			
OUT = Output to:				
Fixed Port	1110011w	port		
Variable Port	1110111w			
XLAT = Translate Byte to AL	11010111			
LEA = Load EA to Register	10001101	mod reg r/m		
LDS = Load Pointer to DS	11000101	mod reg r/m		
LES = Load Pointer to ES	11000100	mod reg r/m		
LAHF = Load AH with Flags	10011111			
SAHF = Store AH into Flags	10011110			
PUSHF = Push Flags	10011100			
POPF = Pop Flags	10011101			

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
ARITHMETIC	76543210	76543210	76543210	76543210
ADD = Add:				
Reg./Memory with Register to Either	000000dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod 000 r/m	data	data if s w = 01
Immediate to Accumulator	0000010w	data	data if w = 1	
ADC = Add with Carry:				
Reg./Memory with Register to Either	000100dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod 010 r/m	data	data if s w = 01
Immediate to Accumulator	0001010w	data	data if w = 1	
INC = Increment:				
Register/Memory	1111111w	mod 000 r/m		
Register	01000reg			
AAA = ASCII Adjust for Add	00110111			
DAA = Decimal Adjust for Add	00100111			
SUB = Subtract:				
Reg./Memory and Register to Either	001010dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod 101 r/m	data	data if s w = 01
Immediate from Accumulator	0010110w	data	data if w = 1	
SBB = Subtract with Borrow				
Reg./Memory and Register to Either	000110dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod 011 r/m	data	data if s w = 01
Immediate from Accumulator	0001110w	data	data if w = 1	
DEC = Decrement:				
Register/Memory	1111111w	mod 001 r/m		
Register	01001 reg			
NEG = Change Sign	1111011w	mod 011 r/m		
CMP = Compare:				
Register/Memory and Register	001110dw	mod reg r/m		
Immediate with Register/Memory	100000sw	mod 111 r/m	data	data if s w = 01
Immediate with Accumulator	0011110w	data	data if w = 1	
AAS = ASCII Adjust for Subtract	00111111			
DAS = Decimal Adjust for Subtract	00101111			
MUL = Multiply (Unsigned)	1111011w	mod 100 r/m		
IMUL = Integer Multiply (Signed)	1111011w	mod 101 r/m		
AAM = ASCII Adjust for Multiply	11010100	00001010		
DIV = Divide (Unsigned)	1111011w	mod 110 r/m		
IDIV = Integer Divide (Signed)	1111011w	mod 111 r/m		
AAD = ASCII Adjust for Divide	11010101	00001010		
CBW = Convert Byte to Word	10011000			
CWD = Convert Word to Double Word	10011001			

8086/8088 Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
LOGIC	76543210	76543210	76543210	76543210
NOT = Invert	1111011w	mod010r/m		
SHL/SAL = Shift Logical/Arithmetic Left	110100vw	mod100r/m		
SHR = Shift Logical Right	110100vw	mod101r/m		
SAR = Shift Arithmetic Right	110100vw	mod111r/m		
ROL = Rotate Left	110100vw	mod000r/m		
ROR = Rotate Right	110100vw	mod001r/m		
RCL = Rotate Through Carry Flag Left	110100vw	mod010r/m		
RCR = Rotate Through Carry Right	110100vw	mod011r/m		
AND = **And:**				
Reg./Memory and Register to Either	001000dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod100r/m	data	data if w = 1
Immediate to Accumulator	0010010w	data	data if w = 1	
TEST = **And Function to Flags, No Result:**				
Register/Memory and Register	1000010w	mod reg r/m		
Immediate Data and Register/Memory	1111011w	mod000r/m	data	data if w = 1
Immediate Data and Accumulator	1010100w	data	data if w = 1	
OR = **Or:**				
Reg./Memory and Register to Either	000010dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod001r/m	data	data if w = 1
Immediate to Accumulator	0000110w	data	data if w = 1	
XOR = **Exclusive OR:**				
Reg./Memory and Register to Either	001100dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod110r/m	data	data if w = 1
Immediate to Accumulator	0011010w	data	data if w = 1	
STRING MANIPULATION				
REP = Repeat	1111001z			
MOVS = Move Byte/Word	1010010w			
CMPS = Compare Byte/Word	1010011w			
SCAS = Scan Byte/Word	1010111w			
LODS = Load Byte/Wd to AL/AX	1010110w			
STOS = Stor Byte/Wd from AL/A	1010101w			
CONTROL TRANSFER				
CALL = **Call:**				
Direct Within Segment	11101000	disp-low	disp-high	
Indirect Within Segment	11111111	mod010r/m		
Direct Intersegment	10011010	offset-low	offset-high	
		seg-low	seg-high	
Indirect Intersegment	11111111	mod011r/m		

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code		
CONTROL TRANSFER (Continued)			
JMP = Unconditional Jump:	76543210	76543210	76543210
Direct Within Segment	11101001	disp-low	disp-high
Direct Within Segment-Short	11101011	disp	
Indirect Within Segment	11111111	mod 100 r/m	
Direct Intersegment	11101010	offset-low	offset-high
		seg-low	seg-high
Indirect Intersegment	11111111	mod 101 r/m	
RET = Return from CALL:			
Within Segment	11000011		
Within Seg. Adding Immed to SP	11000010	data-low	data-high
Intersegment	11001011		
Intersegment Adding Immediate to SP	11001010	data-low	data-high
JE/JZ = Jump on Equal/Zero	01110100	disp	
JL/JNGE = Jump on Less/Not Greater or Equal	01111100	disp	
JLE/JNG = Jump on Less or Equal/ Not Greater	01111110	disp	
JB/JNAE = Jump on Below/Not Above or Equal	01110010	disp	
JBE/JNA = Jump on Below or Equal/ Not Above	01110110	disp	
JP/JPE = Jump on Parity/Parity Even	01111010	disp	
JO = Jump on Overflow	01110000	disp	
JS = Jump on Sign	01111000	disp	
JNE/JNZ = Jump on Not Equal/Not Zero	01110101	disp	
JNL/JGE = Jump on Not Less/Greater or Equal	01111101	disp	
JNLE/JG = Jump on Not Less or Equal/ Greater	01111111	disp	
JNB/JAE = Jump on Not Below/Above or Equal	01110011	disp	
JNBE/JA = Jump on Not Below or Equal/Above	01110111	disp	
JNP/JPO = Jump on Not Par/Par Odd	01111011	disp	
JNO = Jump on Not Overflow	01110001	disp	
JNS = Jump on Not Sign	01111001	disp	
LOOP = Loop CX Times	11100010	disp	
LOOPZ/LOOPE = Loop While Zero/Equal	11100001	disp	
LOOPNZ/LOOPNE = Loop While Not Zero/Equal	11100000	disp	
JCXZ = Jump on CX Zero	11100011	disp	
INT = Interrupt			
Type Specified	11001101	type	
Type 3	11001100		
INTO = Interrupt on Overflow	11001110		
IRET = Interrupt Return	11001111		

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code	
PROCESSOR CONTROL	76543210	76543210
CLC = Clear Carry	11111000	
CMC = Complement Carry	11110101	
STC = Set Carry	11111001	
CLD = Clear Direction	11111100	
STD = Set Direction	11111101	
CLI = Clear Interrupt	11111010	
STI = Set Interrupt	11111011	
HLT = Halt	11110100	
WAIT = Wait	10011011	
ESC = Escape (to External Device)	11011xxx	mod x x x r/m
LOCK = Bus Lock Prefix	11110000	

NOTES:
AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value.
Greater = more positive:
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
if w = 1 then word instruction; if w = 0 then byte instruction
if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16 bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low
if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)
*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.
**MOV CS, REG/MEMORY not allowed.
if s w = 01 then 16 bits of immediate data form the operand
if s w = 11 then an immediate data byte is sign extended to form the 16-bit operand
if v = 0 then "count" = 1; if v = 1 then "count" in (CL) register
x = don't care
z is used for string primitives for comparison with ZF FLAG

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:
FLAGS =
X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

DATA SHEET REVISION REVIEW

The following list represents key differences between this and the -001 data sheet. Please review this summary carefully.

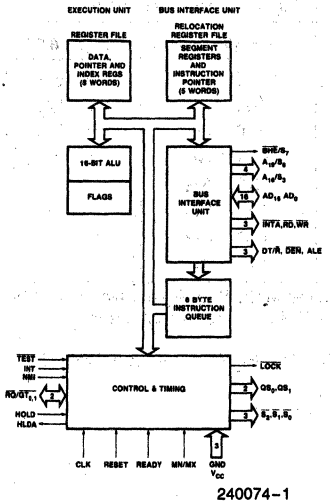
1. In the Pin Description Table (Table 1), the description of the HLDA signal being issued has been corrected. HLDA will be issued in the middle of either the T_4 or T_i state.

80C86AL
16-BIT CHMOS MICROPROCESSOR

- **Pin-for-Pin and Functionally Compatible to Industry Standard HMOS 8086**
- **Fully Static Design with Frequency Range from D.C. to:**
  - **— 5 MHz for 80C86AL**
  - **— 8 MHz for 80C86AL-2**
- **Low Power Operation**
  - **— Operating I_{CC} = 10 mA/MHz**
  - **— Standby I_{CCS} = 500 μA Max**
- **Bus-Hold Circuitry Eliminates Pull-Up Resistors**
- **Direct Addressing Capability of 1 MByte of Memory**
- **Architecture Designed for Powerful Assembly Language and Efficient High Level Languages**
- **24 Operand Addressing Modes**
- **Byte, Word and Block Operations**
- **8 and 16-Bit Signed and Unsigned Arithmetic**
  - **— Binary or Decimal**
  - **— Multiply and Divide**
- **Available in 40-Lead Plastic DIP and 44-Lead PLCC Packages**

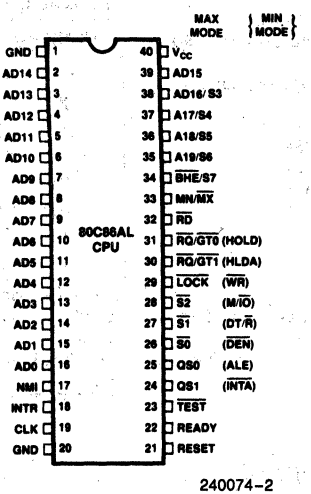
(See Packaging Spec., Order #231369)

The Intel 80C86AL is a high performance, CHMOS version of the industry standard HMOS 8086 16-bit CPU. It is available in 5 and 8 MHz clock rates. The 80C86AL offers two modes of operation: MINimum for small systems and MAXimum for larger applications such as multiprocessing. It is available in 40-pin DIP and 44-pin plastic leaded chip carrier (PLCC) package.



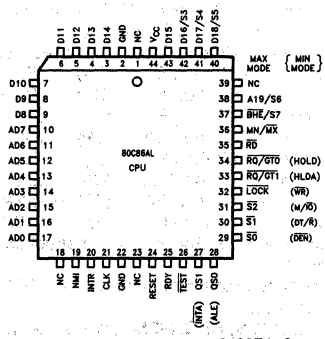
240074-1

Figure 1. 80C86AL CPU Block Diagram



240074-2

Figure 2a. 80C86AL 40-Lead P-DIP Configuration



240074-3

Figure 2b. 80C86AL 44-Lead PLCC Configuration

September 1988
Order Number: 240074-002

Table 1. Pin Description

The following pin function descriptions are for 80C86AL systems in either minimum or maximum mode. The "Local Bus" in these descriptions is the direct multiplexed bus interface connection to the 80C86AL (without regard to additional bus buffers).

Symbol	P-DIP Config. Pin No.	Type	Name and Function
AD_{15}–AD_0	2–16, 39	I/O	**ADDRESS DATA BUS:** These lines constitute the time multiplexed memory/IO address (T_1) and data (T_2, T_3, T_W, T_4) bus. A_0 is analogous to $\overline{BHE}$ for the lower byte of the data bus, pins D_7–D_0. It is LOW during T_1 when a byte is to be transferred on the lower portion of the bus in memory or I/O operations. Eight-bit oriented devices tied to the lower half would normally use A_0 to condition chip select functions. (See $\overline{BHE}$.) These lines are active HIGH and float to 3-state OFF(1) during interrupt acknowledge and local bus "hold acknowledge."
A_{19}/S_6, A_{18}/S_5, A_{17}/S_4, A_{16}/S_3	35–38	O	**ADDRESS/STATUS:** During T_1 these are the four most significant address lines for memory operations. During I/O operations these lines are LOW. During memory and I/O operations, status information is available on these lines during T_2, T_3, T_W, and T_4. The status of the interrupt enable FLAG bit (S_5) is updated at the beginning of each CLK cycle. A_{17}/S_4 and A_{16}/S_3 are encoded as shown. This information indicates which relocation register is presently being used for data accessing. These lines float to 3-state OFF(1) during local bus "hold acknowledge."

A_{17}/S_4	A_{16}/S_3	Characteristics
0 (LOW)	0	Alternate Data
0	1	Stack
1 (HIGH)	0	Code or None
1	1	Data
S_6 is 0 (LOW)		

$\overline{BHE}/S_7$	34	O	**BUS HIGH ENABLE/STATUS:** During T_1 the bus high enable signal ($\overline{BHE}$) should be used to enable data onto the most significant half of the data bus, pins D_{15}–D_8. Eight-bit oriented devices tied to the upper half of the bus would normally use $\overline{BHE}$ to condition chip select functions. $\overline{BHE}$ is LOW during T_1 for read, write, and interrupt acknowledge cycles when a byte is to be transferred on the high portion of the bus. The S_7 status information is available during T_2, T_3, and T_4. The signal is active LOW, and floats to 3-state OFF(1) in "hold." It is LOW during T_1 for the first interrupt acknowledge cycle.

$\overline{BHE}$	A_0	Characteristics
0	0	Whole word
0	1	Upper byte from/ to odd address
1	0	Lower byte from/ to even address
1	1	None

Table 1. Pin Description (Continued)

Symbol	P-DIP Config. Pin No.	Type	Name and Function
$\overline{RD}$	32	O	**READ:** Read strobe indicates that the processor is performing a memory of I/O read cycle, depending on the state of the S_2 pin. This signal is used to read devices which reside on the 80C86AL local bus. $\overline{RD}$ is active LOW during T_2, T_3 and T_W of any read cycle, and is guaranteed to remain HIGH in T_2 until the 80C86AL local bus has floated. This floats to 3-state OFF in "hold acknowledge."
READY	22	I	**READY:** is the acknowledgement from the addressed memory or I/O device that it will complete the data transfer. The READY signal from memory/IO is synchronized by the 82C84A Clock Generator to form READY. This signal is active HIGH. The 80C86AL READY input is not synchronized. Correct operation is not guaranteed if the setup and hold times are not met.
INTR	18	I	**INTERRUPT REQUEST:** is a level triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored to via an interrupt vector lookup table located in system memory. It can be internally masked by software resetting the interrupt enable bit. INTR is internally synchronized. This signal is active HIGH.
$\overline{TEST}$	23	I	**$\overline{TEST}$:** input is examined by the "Wait" instruction. If the $\overline{TEST}$ input is LOW execution continues, otherwise the processor waits in an "Idle" state. This input is synchronized internally during each clock cycle on the leading edge of CLK.
NMI	17	I	**NON-MASKABLE INTERRUPT:** an edge triggered input which causes a type 2 interrupt. A subroutine is vectored to via an interrupt vector lookup table located in system memory. NMI is not maskable internally by software. A transition from a LOW to HIGH initiates the interrupt at the end of the current instruction. This input is internally synchronized.
RESET	21	I	**RESET:** causes the processor to immediately terminate its present activity. The signal must be active HIGH for at least four clock cycles. It restarts execution, as described in the Instruction Set description, when RESET returns LOW. RESET is internally synchronized.
CLK	19	I	**CLOCK:** provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing.
V_{CC}	40		**V_{CC}:** +5V power supply pin.
GND	1, 20		**GROUND:** Both must be connected.
MN/$\overline{MX}$	33	I	**MINIMUM/MAXIMUM:** indicates what mode the processor is to operate in. The two modes are discussed in the following sections.

Table 1. Pin Description (Continued)

The following pin function descriptions are for the 80C86AL/82C88 system in maximum mode (i.e., MN/$\overline{MX}$ = V_{SS}). Only the pin functions which are unique to maximum mode are described; all other pin functions are as described above.

<table>
<tr><th>Symbol</th><th>P-DIP Config. Pin No.</th><th>Type</th><th>Name and Function</th></tr>
<tr><td>$\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$</td><td>26–28</td><td>O</td><td>STATUS: active during T_4, T_1, and T_2 and is returned to the passive state (1,1,1) during T_3 or during T_W when READY is HIGH. This status is used by the 82C88 Bus Controller to generate all memory and I/O access control signals. Any change by $\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$ during T_4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T_3 or T_W is used to indicate the end of a bus cycle.
These signals float to 3-state OFF(1) in "hold acknowledge." These status lines are encoded as shown.
<table>
<tr><th>$\overline{S_2}$</th><th>$\overline{S_1}$</th><th>$\overline{S_0}$</th><th>Characteristics</th></tr>
<tr><td>0 (LOW)</td><td>0</td><td>0</td><td>Interrupt Acknowledge</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Read I/O Port</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Write I/O Port</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Halt</td></tr>
<tr><td>1 (HIGH)</td><td>0</td><td>0</td><td>Code Access</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Read Memory</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Write Memory</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Passive</td></tr>
</table></td></tr>
<tr><td>$\overline{RQ}/\overline{GT}_0$, $\overline{RQ}/\overline{GT}_1$</td><td>30, 31</td><td>I/O</td><td>REQUEST/GRANT: pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional with $\overline{RQ}/\overline{GT}_0$ having higher priority than $\overline{RQ}/\overline{GT}_1$. $\overline{RQ}/\overline{GT}$ has an internal pull-up resistor so may be left unconnected. The request/grant sequence is as follows (see timing diagram):
1. A pulse of 1 CLK wide from another local bus master indicates a local bus request ("hold") to the 80C86AL (pulse 1).
2. During a T_4 or T_1 clock cycle, a pulse 1 CLK wide from the 80C86AL to the requesting master (pulse 2), indicates that the 80C86AL has allowed the local bus to float and that it will enter the "hold acknowledge" state at the next CLK. The CPU's bus interface unit is disconnected logically from the local bus during "hold acknowledge."
3. A pulse 1 CLK wide from the requesting master indicates to the 80C86AL (pulse 3) that the "hold" request is about to end and that 80C86AL can reclaim the local bus at the next CLK.
Each master-master exchange of the local bus is a sequence of 3 pulses. There must be one dead CLK cycle after each bus exchange. Pulses are active LOW.
If the request is made while the CPU is performing a memory cycle, it will release the local bus during T_4 of the cycle when all the following conditions are met:
1. Request occurs on or before T_2.
2. Current cycle is not the low byte of a word (on an odd address).
3. Current cycle is not the first acknowledge of an interrupt acknowledge sequence.
4. A locked instruction is not currently executing.</td></tr>
</table>

Table 1. Pin Description (Continued)

Symbol	P-DIP Config. Pin No.	Type	Name and Function
			If the local bus is idle when the request is made the two possible events will follow: 1. Local bus will be released during the next clock. 2. A memory cycle will start within 3 clocks. Now the four rules for a currently active memory cycle apply with condition number 1 already satisfied.
$\overline{LOCK}$	29	O	**$\overline{LOCK}$:** output indicates that other system bus masters are not to gain control of the system bus while $\overline{LOCK}$ is active LOW. The $\overline{LOCK}$ signal is activated by the "LOCK" prefix instruction and remains active until the completion of the next instruction. This signal is active LOW, and floats to 3-state OFF(1) in "hold acknowledge."
QS_1, QS_0	24, 25	O	**QUEUE STATUS:** The queue status is valid during the CLK cycle after which the queue operation is performed. QS_1 and QS_0 provide status to allow external tracking of the internal 80C86AL instruction queue.

QS_1	QS_0	Characteristics
0 (LOW)	0	No Operation
0	1	First Byte of Op Code from Queue
1 (HIGH)	0	Empty the Queue
1	1	Subsequent Byte from Queue

The following pin function descriptions are for the 80C86AL in minimum mode (i.e., MN/$\overline{MX}$ = V_{CC}). Only the pin functions which are unique to minimum mode are described; all other pin functions are described above.

M/$\overline{IO}$	28	O	**STATUS LINE:** logically equivalent to S_2 in the maximum mode. It is used to distinguish a memory access from an I/O access. M/$\overline{IO}$ becomes valid in the T_4 preceding a bus cycle and remains valid until the final T_4 of the cycle (M = HIGH, IO = LOW). M/$\overline{IO}$ floats to 3-state OFF(1) in local bus "hold acknowledge."
$\overline{WR}$	29	O	**WRITE:** indicates that the processor is performing a write memory or write I/O cycle, depending on the state of the M/$\overline{IO}$ signal. $\overline{WR}$ is active for T_2, T_3 and T_W of any write cycle. It is active LOW, and floats to 3-state OFF(1) in local bus "hold acknowledge."
$\overline{INTA}$	24	O	$\overline{INTA}$ is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T_2, T_3 and T_W of each interrupt acknowledge cycle.
ALE	25	O	**ADDRESS LATCH ENABLE:** provided by the processor to latch the address into an address latch. It is a HIGH pulse active during T_1 of any bus cycle. Note that ALE is never floated.
DT/$\overline{R}$	27	O	**DATA TRANSMIT/RECEIVE:** needed in minimum system that desires to use a data bus transceiver. It is used to control the direction of data flow through the transceiver. Logically DT/$\overline{R}$ is equivalent to $\overline{S_1}$ in the maximum mode, and its timing is the same as for M/$\overline{IO}$. (T = HIGH, R = LOW.) This signal floats to 3-state OFF(1) in local bus "hold acknowledge."

Table 1. Pin Description (Continued)

Symbol	P-DIP Config. Pin No.	Type	Name and Function
$\overline{DEN}$	26	O	**DATA ENABLE:** provided as an output enable for the transceiver in a minimum system which uses the transceiver. $\overline{DEN}$ is active LOW during each memory and I/O access and for INTA cycles. For a read or $\overline{INTA}$ cycle it is active from the middle of T_2 until the middle of T_4, while for a write cycle it is active from the beginning of T_2 until the middle of T_4. $\overline{DEN}$ floats to 3-state OFF(1) in local bus "hold acknowledge."
HOLD, HLDA	31, 30	I/O	**HOLD:** indicates that another master is requesting a local bus "hold." To be acknowledged, HOLD must be active HIGH. The processor receiving the "hold" request will issue HLDA (HIGH) as an acknowledgement in the middle of a T_4 or T_i clock cycle. Simultaneous with the issuance of HLDA the processor will float the local bus and control lines. After HOLD is detected as being LOW, the processor will LOWer the HLDA, and when the processor needs to run another cycle, it will again drive the local bus and control lines. The same rules as for $\overline{RQ}/\overline{GT}$ apply regarding when the local bus will be released. HOLD is not an asynchronous input. External synchronization should be provided if the system cannot otherwise guarantee the setup time.

NOTE:
1. See the section on Bus Hold Circuitry.

FUNCTIONAL DESCRIPTION

STATIC OPERATION

All 80C86AL circuitry is of static design. Internal registers, counters and latches are static and require no refresh as with dynamic circuit design. This eliminates the minimum operating frequency restriction placed on other microprocessors. The CMOS 80C86AL can operate from DC to the appropriate upper frequency limit. The processor clock may be stopped in either state (high/low) and held there indefinitely. This type of operation is especially useful for system debug or power critical applications.

The 80C86AL can be single stepped using only the CPU clock. This state can be maintained as long as is necessary. Single step clock operation allows simple interface circuitry to provide critical information for bringing up your system.

Static design also allows very low frequency operation. In a power critical situation, this can provide extremely low power operation since 80C86AL power dissipation is directly related to operating frequency. As the system frequency is reduced, so is the operating power until, ultimately, at a DC input frequency, the 80C86AL power requirement is the standby current.

INTERNAL ARCHITECTURE

The internal functions of the 80C86AL processor are partitioned logically into two processing units. The first is the Bus Interface Unit (BIU) and the second is the Execution Unit (EU) as shown in the block diagram of Figure 1.

These units can interact directly but for the most part perform as separate asynchronous operational processors. The bus interface unit provides the functions related to instruction fetching and queuing, operand fetch and store, and address relocation. This unit also provides the basic bus control. The overlap of instruction pre-fetching provided by this unit serves to increase processor performance through improved bus bandwidth utilization. Up to 6 bytes of the instruction stream can be queued while waiting for decoding and execution.

The instruction stream queuing mechanism allows the BIU to keep the memory utilized very efficiently. Whenever there is space for at least 2 bytes in the queue, the BIU will attempt a word fetch memory cycle. This greatly reduces "dead time" on the memory bus. The queue acts as a First-In-First Out (FIFO) buffer, from which the EU extracts instruction bytes as required. If the queue is empty (following a branch instruction, for example), the first byte into the queue immediately becomes available to the EU.

Memory Reference Need	Segment Register Used	Segment Selection Rule
Instructions	CODE (CS)	Automatic with all instruction prefetch.
Stack	STACK (SS)	All stack pushes and pops. Memory references relative to BP base register except data references.
Local Data	DATA (DS)	Data references when: relative to stack, destination of string operation, or explicitly overridden.
External (Global) Data	EXTRA (ES)	Destination of string operations: Explicitly selected using a segment override.

The execution units receives pre-fetched instructions from the BIU queue and provides un-relocated operand addresses to the BIU. Memory operands are passed through the BIU for processing by the EU, which passes results to the BIU for storage. See the Instruction Set description for further register set and architectural descriptions.

MEMORY ORGANIZATION

The processor provides a 20-bit address to memory which locates the byte being referenced. The memory is organized as a linear array of up to 1 million bytes, addressed as 00000(H) to FFFFF(H). The memory is logically divided into code, data, extra data, and stack segments of up to 64k bytes each, with each segment falling on 16-byte boundaries. (See Figure 3a.)

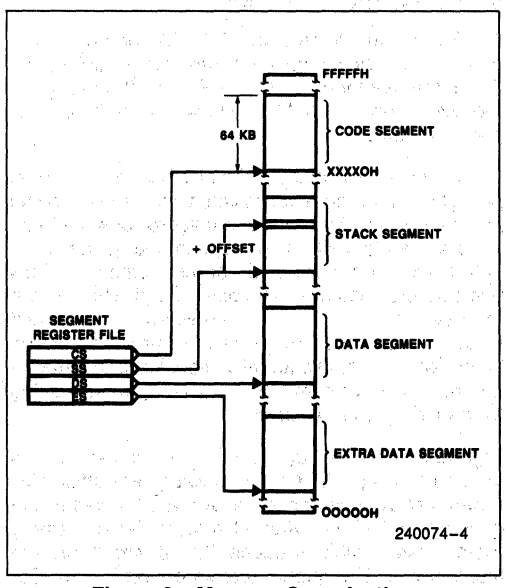


Figure 3a. Memory Organization

All memory references are made relative to base addresses contained in high speed segment registers. The segment types were chosen based on the addressing needs of programs. The segment register to be selected is automatically chosen according to the rules of the following table. All information in one segment type share the same logical attributes (e.g. code or data). By structuring memory into relocatable areas of similar characteristics and by automatically selecting segment registers, programs are shorter, faster, and more structured.

Word (16-bit) operands can be located on even or odd address boundaries and are thus not constrained to even boundaries as is the case in many 16-bit computers. For address and data operands, the least significant byte of the word is stored in the lower valued address location and the most significant byte in the next higher address location. The BIU automatically performs the proper number of memory accesses, one if the word operand is on an even byte boundary and two if it is on an odd byte boundary. Except for the performance penalty, this double access is transparent to the software. This performance penalty does not occur for instruction fetches, only word operands.

Physically, the memory is organized as a high bank (D_{15}–D_8) and a low bank (D_7–D_0) of 512k 8-bit bytes addressed in parallel by the processor's address lines.

A_{19}–A_1. Byte data with even addresses is transferred on the D_7-D_0 bus lines while odd addressed byte data (A_0 HIGH) is transferred on the D_{15}–D_8 bus lines. The processor provides two enable signals, $\overline{BHE}$ and A_0, to selectively allow reading from or writing into either an odd byte location, even byte location, or both. The instruction stream is fetched from memory as words and is addressed internally by the processor to the byte level as necessary.

In referencing word data the BIU requires one or two memory cycles depending on whether or not the starting byte of the word is on an even or odd address, respectively. Consequently, in referencing

word operands performance can be optimized by locating data on even address boundaries. This is an especially useful technique for using the stack, since odd address references to the stack may adversely affect the context switching time for interrupt processing or task multiplexing.

Certain locations in memory are reserved for specific CPU operations (see Figure 3b.) Locations from address FFFF0H through FFFFFH are reserved for operations including a jump to the initial program loading routine. Following RESET, the CPU will always begin execution at location FFFF0H where the jump must be. Locations 00000H through 003FFH are reserved for interrupt operations. Each of the 256 possible interrupt types has its service routine pointed to by a 4-byte pointer element consisting of a 16-bit segment address and a 16-bit offset address. The pointer elements are assumed to have been stored at the respective places in reserved memory prior to occurrence of interrupts.

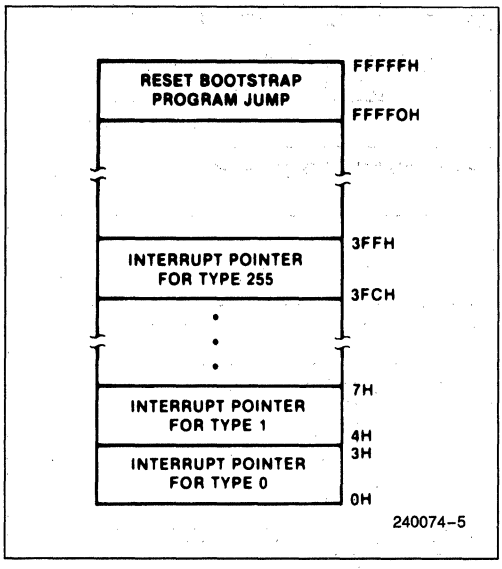


Figure 3b. Reserved Memory Locations

MINIMUM AND MAXIMUM MODES

The requirements for supporting minimum and maximum 80C86AL systems are sufficiently different that they cannot be done efficiently with 40 uniquely defined pins. Consequently, the 80C86AL is equipped with a strap pin (MN/$\overline{MX}$) which defines the system configuration. The definition of a certain subset of the pins changes dependent on the condition of the strap pin. When MN/$\overline{MX}$ pin is strapped to GND, the 80C86AL treats pins 24 through 31 in maximum mode. An 82C88 bus controller interprets status information coded into $\overline{S}_0$, $\overline{S}_1$, $\overline{S}_2$ to generate bus timing and control signals compatible with the MULTIBUS® architecture. When the MN/$\overline{MX}$ pin is strapped to V_{CC}, the 80C86AL generates bus control signals itself on pins 24 through 31, as shown in parentheses in Figure 2. Examples of minimum mode and maximum mode systems are shown in Figure 4.

BUS OPERATION

The 80C86AL has a combined address and data bus commonly referred to as a time multiplexed bus. This technique provides the most efficient use of pins on the processor. This "local bus" can be buffered directly and used throughout the system with address latching provided on memory and I/O modules. In addition, the bus can also be demultiplexed at the processor with a single set of address latches if a standard non-multiplexed bus is desired for the system.

Each processor bus cycle consists of at least four CLK cycles. These are referred to as T_1, T_2, T_3 and T_4 (see Figure 5). The address is emitted from the processor during T_1 and data transfer occurs on the bus during T_3 and T_4. T_2 is used primarily for changing the direction of the bus during read operations. In the event that a "NOT READY" indication is given by the addressed device, "Wait" states (T_W) are inserted between T_3 and T_4. Each inserted "Wait" state is of the same duration as a CLK cycle. Periods can occur between 80C86AL bus cycles. These are referred to as "Idle" states (T_1) or inactive CLK cycles. The processor uses these cycles for internal housekeeping.

During T_1 of any bus cycle the ALE (Address Latch Enable) signal is emitted (by either the processor or the 82C88 bus controller, depending on the MN/$\overline{MX}$ strap). At the trailing edge of this pulse, a valid address and certain status information for the cycle may be latched.

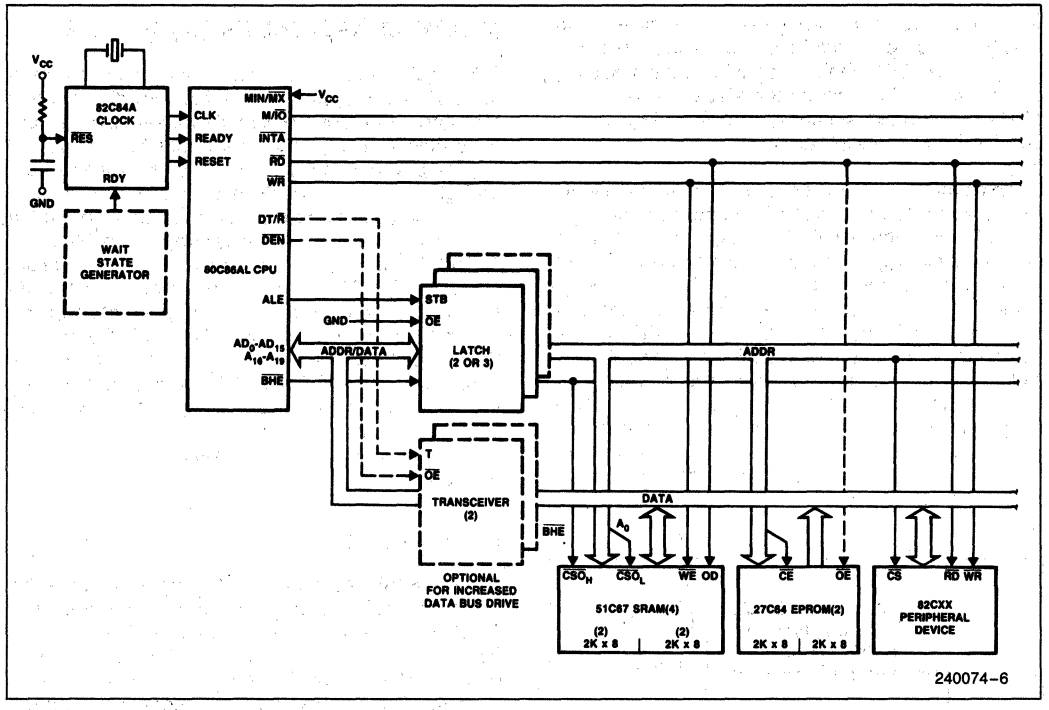


Figure 4a. Minimum Mode iAPX 80C86AL Typical Configuration

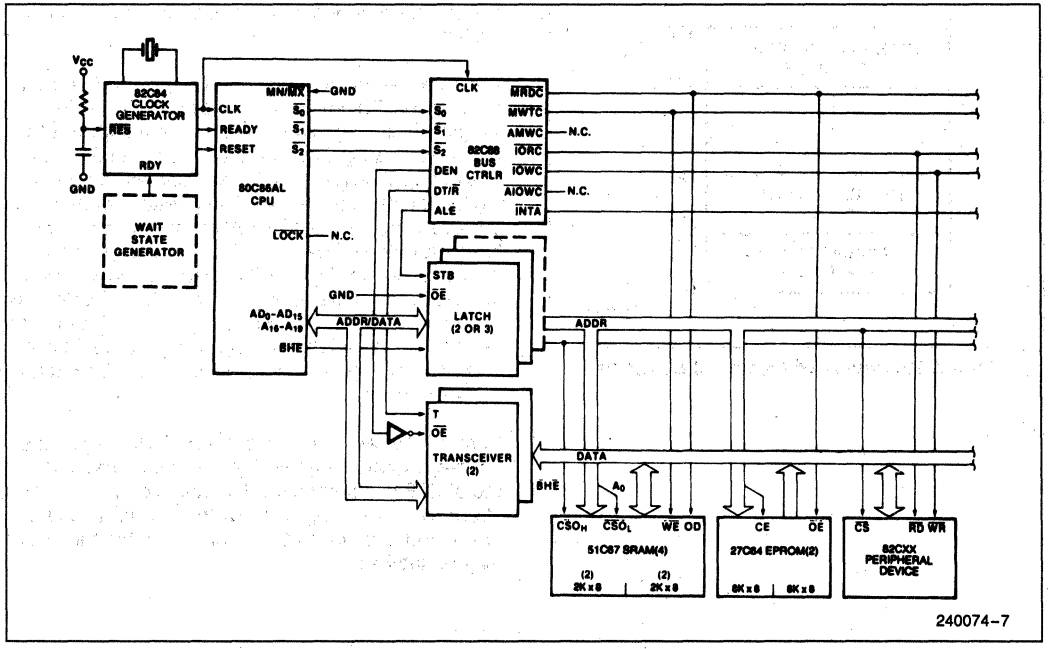


Figure 4b. Maximum Mode 80C86AL Typical Configuration

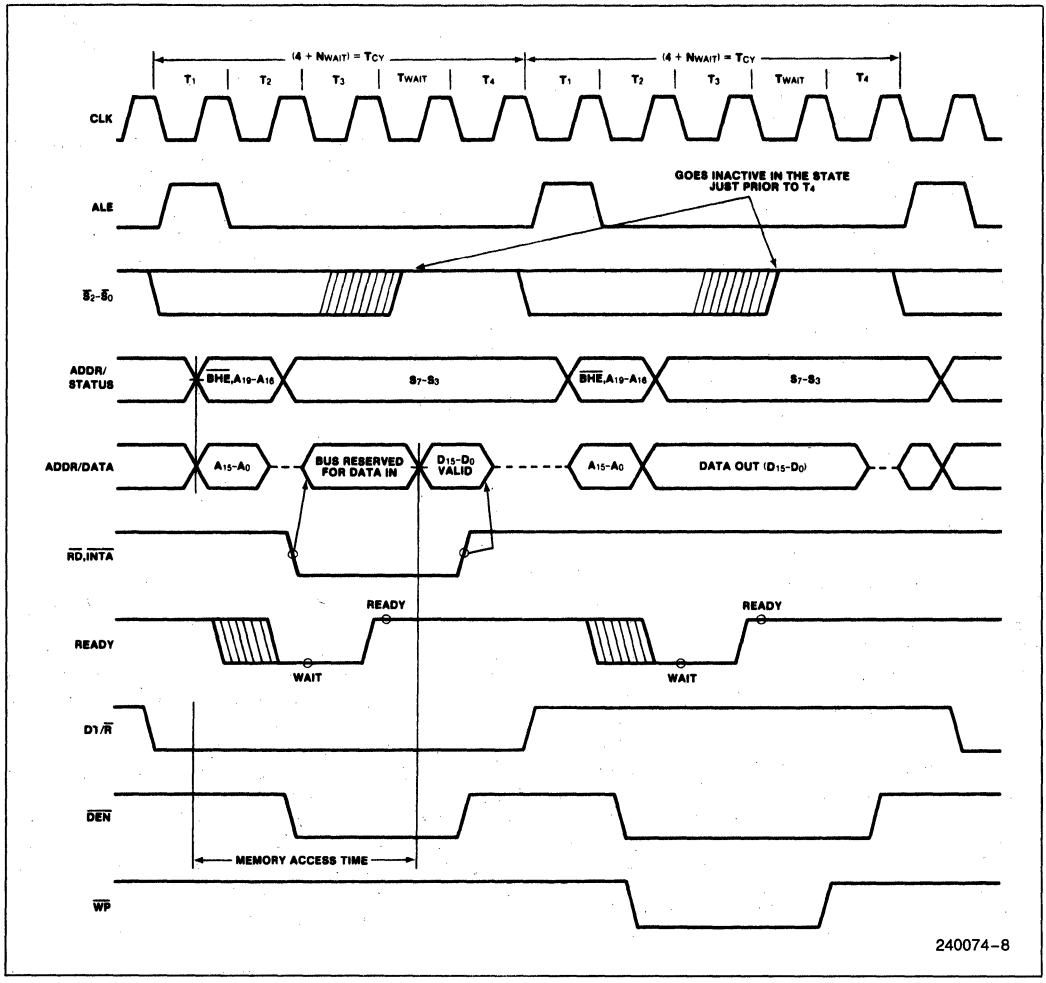


Figure 5. Basic System Timing

Status bits $\overline{S}_0$, $\overline{S}_1$, and $\overline{S}_2$ are used, in maximum mode, by the bus controller to identify the type of bus transaction according to the following table:

$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	Characteristics
0 (LOW)	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1 (HIGH)	0	0	Instruction Fetch
1	0	1	Read Data from Memory
1	1	0	Write Data to Memory
1	1	1	Passive (no bus cycle)

Status bits S_3 through S_7 are multiplexed with high-order address bits and the $\overline{BHE}$ signal, and are therefore valid during T_2 through T_4. S_3 and S_4 indicate which segment register (see Instruction Set description) was used for this bus cycle in forming the address, according to the following table:

S_4	S_3	Characteristics
0 (LOW)	0	Alternate Data (extra segment)
0	1	Stack
1 (HIGH)	0	Code or None
1	1	Data

S_5 is a reflection of the PSW interrupt enable bit. $S_6 = 0$ and S_7 is a spare status pin.

I/O ADDRESSING

In the 80C86AL, I/O operations can address up to a maximum of 64k I/O byte registers or 32k I/O word registers. The I/O address appears in the same format as the memory address on bus lines A_{15}–A_0. The address lines A_{19}–A_{16} are zero in I/O operations. The variable I/O instructions which use register DX as a pointer have full address capability while the direct I/O instructions directly address one or two of the 256 I/O byte locations in page 0 of the I/O address space.

I/O ports are addressed in the same manner as memory locations. Even addressed bytes are transferred on the D_7–D_0 bus lines and odd addressed bytes on D_{15}–D_8. Care must be taken to assure that each register within an 8-bit peripheral located on the lower portion of the bus be addressed as even.

EXTERNAL INTERFACE

PROCESSOR RESET AND INITIALIZATION

Processor initialization or start up is accomplished with activation (HIGH) of the RESET pin. The 80C86AL RESET is required to be HIGH for four or more CLK cycles. The 80C86AL will terminate operations on the high-going edge of RESET and will remain dormant as long as RESET is HIGH. The low-going transition of RESET triggers an internal reset sequence for approximately 7 CLK cycles. After this interval the 80C86AL operates normally beginning with the instruction in absolute location FFFF0H (see Figure 3b). The details of this operation are specified in the Instruction Set description of the MCS®-86 Family User's Manual. The RESET input is internally synchronized to the processor clock. At initialization the HIGH-to-LOW transition of RESET must occur no sooner than 50 μs after power-up, to allow complete initialization of the 80C86AL.

NMI asserted prior to the 2nd clock after the end of RESET will not be honored. If NMI is asserted after that point and during the internal reset sequence, the processor may execute one instruction before responding to the interrupt. A hold request active immediately after RESET will be honored before the first instruction fetch.

All 3-state outputs float to 3-state OFF(1) during RESET. Status is active in the idle state for the first clock after RESET becomes active and then floats to 3-state OFF(1). ALE and HLDA are driven low.

NOTE:
1. See the section on Bus Hold Circuitry.

BUS HOLD CIRCUITRY

To avoid high current conditions caused by floating inputs to CMOS devices and eliminate the need for pull-up/down resistors, "bus-hold" circuitry has been used on the 80C86AL pins 2–16, 26–32, and 34–39 (Figures 6a, 6b). These circuits will maintain the last valid logic state if no driving source is present (i.e. an unconnected pin or a driving source which goes to a high impedance state). To overdrive the "bus hold" circuits, an external driver must be capable of supplying 350 μA minimum sink or source current at valid input voltage levels. Since this "bus hold" circuitry is active and not a "resistive" type element, the associated power supply current is negligible and power dissipation is significantly reduced when compared to the use of passive pull-up resistors.

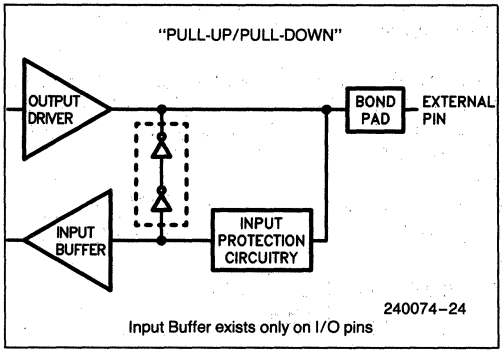


Figure 6a. Bus hold circuitry pin 2-16, 34-39 for P-DIP package.

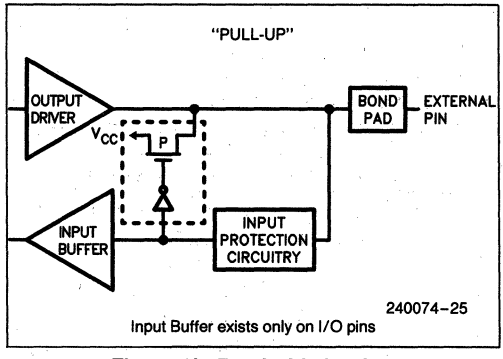


Figure 6b. Bus hold circuitry pin 26-32 for P-DIP package.

INTERRUPT OPERATIONS

Interrupt operations fall into two classes; software or hardware initiated. The software initiated interrupts and software aspects of hardware interrupts are specified in the Instruction Set description. Hardware interrupts can be classified as non-maskable or maskable.

Interrupts result in a transfer of control to a new program location. A 256-element table containing address pointers to the interrupt service program locations resides in absolute locations 0 through 3FFH (see Figure 3b), which are reserved for this purpose. Each element in the table is 4 bytes in size and corresponds to an interrupt "type". An interrupting device supplies an 8-bit type number, during the interrupt acknowledge sequence, which is used to "vector" through the appropriate element to the new interrupt service program location.

NON-MASKABLE INTERRUPT (NMI)

The processor provides a single non-maskable interrupt pin (NMI) which has higher priority than the maskable interrupt request pin (INTR). A typical use would be to activate a power failure routine. The NMI is edge-triggered on a LOW-to-HIGH transition. The activation of this pin causes a type 2 interrupt. (See Instruction Set description.) NMI is required to have a duration in the HIGH state of greater than two CLK cycles, but is not required to be synchronized to the clock. Any high-going transition of NMI is latched on-chip and will be serviced at the end of the current instruction or between whole moves of a block-type instruction. Worst case response to NMI would be for multiply, divide and variable shift instructions. There is no specification on the occurrence of the low-going edge; it may occur before, during, or after the servicing of NMI. Another high-going edge triggers another response if it occurs after the start of the NMI procedure. The signal must be free of logical spikes in general and be free of bounces on the low-going edge to avoid triggering extraneous responses.

MASKABLE INTERRUPT (INTR)

The 80C86AL provides a single interrupt request input (INTR) which can be masked internally by software with the resetting of the interrupt enable FLAG status bit. The interrupt request signal is level triggered. It is internally synchronized during each clock cycle on the high-going edge of CLK. To be responded to, INTR must be present (HIGH) during the clock period preceding the end of the current instruction or the end of a whole move for a block-type instruction. During the interrupt response sequence further interrupts are disabled. The enable bit is reset as part of the response to any interrupt (INTR, NMI, software interrupt or single-step), although the FLAGS register which is automatically pushed onto the stack reflects the state of the processor prior to the interrupt. Until the old FLAGS register is restored the enable bit will be zero unless specifically set by an instruction.

During the response sequence (Figure 7) the processor executes two successive (back-to-back) interrupt acknowledge cycles. The 80C86AL emits the LOCK signal from T_2 of the first bus cycle until T_2 of the second. A local bus "hold" request will not be honored until the end of the second bus cycle. In the second bus cycle a byte is fetched from the external interrupt system (e.g., 82C59 PIC) which identifies the source (type) of the interrupt. This byte is multiplied by four and used as a pointer into the interrupt vector lookup table. An INTR signal left HIGH will be continually responded to within the limitations of the enable bit and sample period. The INTERRUPT RETURN instruction includes a FLAGS pop which returns the status of the original interrupt enable bit when it restores the FLAGS.

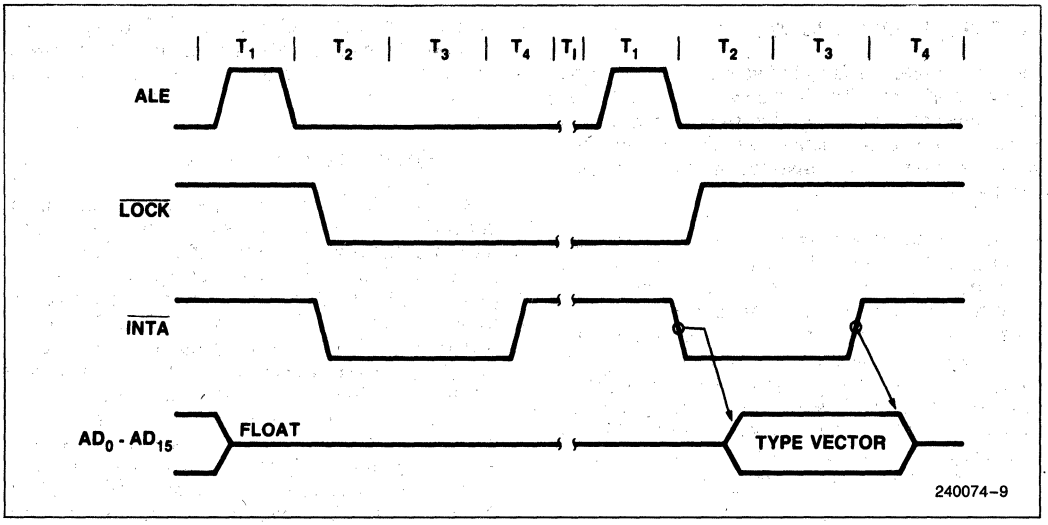


Figure 7. Interrupt Acknowledge Sequence

HALT

When a software "HALT" instruction is executed the processor indicates that it is entering the "HALT" state in one of two ways depending upon which mode is strapped. In minimum mode, the processor issues one ALE with no qualifying bus control signals. In Maximum Mode, the processor issues appropriate HALT status on $\overline{S}_2$, $\overline{S}_1$ and $\overline{S}_0$ and the 82C88 bus controller issues one ALE. The 80C86AL will not leave the "HALT" state when a local bus "hold" is entered while in "HALT". In this case, the processor reissues the HALT indicator. An interrupt request or RESET will force the 80C86AL out of the "HALT" state.

READ/MODIFY/WRITE (SEMAPHORE) OPERATIONS VIA LOCK

The $\overline{LOCK}$ status information is provided by the processor when directly consecutive bus cycles are required during the execution of an instruction. This provides the processor with the capability of performing read/modify/write operations on memory (via the Exchange Register With Memory instruction, for example) without the possibility of another system bus master receiving intervening memory cycles. This is useful in mutliprocessor system configurations to accomplish "test and set lock" operations. The $\overline{LOCK}$ signal is activated (forced LOW) in the clock cycle following the one in which the software "LOCK" prefix instruction is decoded by the EU. It is deactivated at the end of the last bus cycle of the instruction following the "LOCK" prefix instruction. While $\overline{LOCK}$ is active a request on a RQ/GT pin will be recorded and then honored at the end of the LOCK.

EXTERNAL SYNCHRONIZATION VIA TEST

As an alternative to the interrupts and general I/O capabilities, the 80C86AL provides a single software-testable input known as the $\overline{TEST}$ signal. At any time the program may execute a WAIT instruction. If at that time the $\overline{TEST}$ signal is inactive (HIGH), program execution becomes suspended while the processor waits for $\overline{TEST}$ to become active. It must remain active for at least 5 CLK cycles. The WAIT instruction is re-executed repeatedly until that time. This activity does not consume bus cycles. The processor remains in an idle state while waiting. All 80C86AL drivers go to 3-state OFF if bus "Hold" is entered. If interrupts are enabled, they may occur while the processor is waiting. When this occurs the processor fetches the WAIT instruction one extra time, processes the interrupt, and then re-fetches and re-executes the WAIT instruction upon returning from the interrupt.

BASIC SYSTEM TIMING

Typical system configurations for the processor operating in minimum mode and in maximum mode are shown in Figures 4a and 4b, respectively. In minimum mode, the MN/$\overline{MX}$ pin is strapped to V_{CC} and the processor emits bus control signals in a manner similar to the 8085. In maximum mode, the MN/$\overline{MX}$ pin is strapped to V_{SS} and the processor emits coded status information which the 82C88 bus controller uses to generate MULTIBUS compatible bus control signals. Figure 5 illustrates the signal timing relationships.

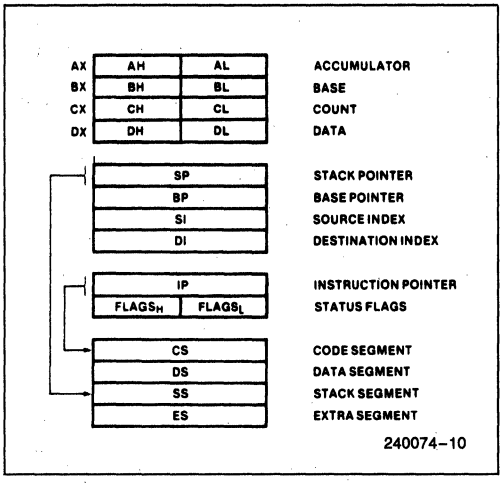


Figure 8. iAPX 80C86AL Register Model

SYSTEM TIMING—MINIMUM SYSTEM

The read cycle begins in T_1 with the assertion of the Address Latch Enable (ALE) signal. The trailing (low-going) edge of this signal is used to latch the address information, which is valid on the local bus at this time, into a latch. The $\overline{BHE}$ and A_0 signals address the low, high, or both bytes. From T_1 to T_4 the M/$\overline{IO}$ signal indicates a memory or I/O operation. At T_2 the address is removed from the local bus and the bus goes to a high impedance state. The read control signal is also asserted at T_2. The read ($\overline{RD}$) signal causes the addressed device to enable its data bus drivers to the local bus. Some time later valid data will be available on the bus and the addressed device will drive the READY line HIGH. When the processor returns the read signal to a HIGH level, the addressed device will again 3-state its bus drivers. If a transceiver is required to buffer the 80C86AL local bus, signals DT/$\overline{R}$ and $\overline{DEN}$ are provided by the 80C86AL.

A write cycle also begins with the assertion of ALE and the emission of the address. The M/$\overline{IO}$ signal is again asserted to indicate a memory or I/O write operation. In the T_2 immediately following the address emission the processor emits the data to be written into the addressed location. This data remains valid until the middle of T_4. During T_2, T_3, and T_W the processor asserts the write control signal. The write ($\overline{WR}$) signal becomes active at the beginning of T_2 as opposed to the read which is delayed somewhat into T_2 to provide time for the bus to float.

The $\overline{BHE}$ and A_0 signals are used to select the proper byte(s) of the memory/IO word to be read or written according to the following table:

$\overline{BHE}$	A0	Characteristics
0	0	Whole word
0	1	Upper byte from/ to odd address
1	0	Lower byte from/ to even address
1	1	None

I/O ports are addressed in the same manner as memory location. Even addressed bytes are transferred on the D_7–D_0 bus lines and odd addressed bytes on D_{15}–D_8.

The basic difference between the interrupt acknowledge cycle and a read cycle is that the interrupt acknowledge signal ($\overline{INTA}$) is asserted in place of the read ($\overline{RD}$) signal and the address bus is floated. (See Figure 7.) In the second of two successive INTA cycles, a byte of information is read from bus lines D_7–D_0 as supplied by the interrupt system logic (i.e., 82C59A Priority Interrupt Controller). This byte identifies the source (type) of the interrupt. It is multiplied by four and used as a pointer into an interrupt vector lookup table, as described earlier.

BUS TIMING—MEDIUM SIZE SYSTEMS

For medium size systems the MN/$\overline{MX}$ pin is connected to V_{SS} and the 82C88 Bus Controller is added to the system as well as a latch for latching the system address, and a transceiver to allow for bus loading greater than the 80C86AL is capable of handling. Signals ALE, DEN, and DT/$\overline{R}$ are generated by the 82C88 instead of the processor in this configuration although their timing remains relatively the same. The 80C86AL status outputs ($\overline{S}_2$, $\overline{S}_1$, and $\overline{S}_0$) provide type-of-cycle information and become 82C88 inputs. This bus cycle information specifies read (code, data, or I/O), write (data or I/O), interrupt acknowledge, or software halt. The 82C88 thus issues control signals specifying memory read or write, I/O read or write, or interrupt acknowledge. The 82C88 provides two types of write strobes, normal and advanced, to be applied as required. The normal write strobes have data valid at the leading edge of write. The advanced write strobes have the same timing as read strobes, and hence data isn't valid at the leading edge of write. The transceiver receives the usual T and OE inputs from the 82C88 DT/$\overline{R}$ and DEN.

The pointer into the interrupt vector table, which is passed during the second INTA cycle, can derive from an 82C59A located on either the local bus or the system bus. If the master 82C59A Priority Interrupt Controller is positioned on the local bus, a TTL gate is required to disable the transceiver when reading from the master 82C59A during the interrupt acknowledge sequence and software "poll".

ABSOLUTE MAXIMUM RATINGS*

Supply Voltage
(With respect to ground) ..... −0.5 to 8.0V
Input Voltage Applied
(w.r.t. ground) ..... −2.0 to V_{CC} + 0.5V
Output Voltage Applied
(w.r.t. ground) ..... −0.5 to V_{CC} + 0.5V
Power Dissipation 1.0W
Storage Temperature ..... −65°C to 150°C
Ambient Temperature Under Bias ..... 0°C to 70°C
Case Temperature (PDIP) ..... 0°C to 80°C
Case Temperature (PLCC) ..... 0°C to 85°C

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

(T_A = 0°C to 70°C, T_{CASE} (Plastic) = 0°C to 80°C, T_{CASE} (PLCC) = 0°C to 85°C)
(V_{CC} = 5V ±10% for 80C86AL, V_{CC} = 5V ±5% for 80C86AL-2)

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage		+0.8	V	(Note 4)
V_{IH}	Input High Voltage (All inputs except clock and MN/$\overline{MX}$)	2.0		V	(Note 5)
V_{CH}	Clock and MN/$\overline{MX}$ Input High Voltage	V_{CC}−0.8		V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2.5 mA
V_{OH}	Output High Voltage	3.0 V_{CC}−0.4		V	I_{OH} = −2.5 mA I_{OH} = −100 μA
I_{CC}	Power Supply Current		10 mA/MHz		V_{IL} = GND, V_{IH} = V_{CC}
I_{CCS}	Standby Supply Current		500	μA	V_{IN} = V_{CC} or GND Outputs Unloaded CLK = GND or V_{CC}
I_{LI}	Input Leakage Current		±1.0	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{BHL}	Input Leakage Current (Bus Hold Low)	50	300	μA	V_{IN} = 0.8V
I_{BHH}	Input Leakage Current (Bus Hold High)	−50	−300	μA	V_{IN} = 3.0V
I_{BHLO}	Bus Hold Low Overdrive		400	μA	(Note 2)
I_{BHHO}	Bus Hold High Overdrive		−400	μA	(Note 3)
I_{LO}	Output Leakage Current		±10	μA	V_{OUT} = GND or V_{CC}
C_{IN}	Capacitance of Input Buffer (All inputs except AD_0–AD_{15}, $\overline{RQ}/\overline{GT}$)		5	pF	(Note 1)
C_{IO}	Capacitance of I/O Buffer (AD_0–AD_{15}, $\overline{RQ}/\overline{GT}$)		20	pF	(Note 1)
C_{OUT}	Output Capacitance		15	pF	(Note 1)

NOTES:
1. Characterization conditions are a) Frequency = 1 MHz; b) Unmeasured pins at GND; c) V_{IN} at +5.0V or GND.
2. An external driver must source at least I_{BHLO} to switch this node from LOW to HIGH.
3. An external driver must sink at least I_{BHHO} to switch this node from HIGH to LOW.
4. V_{IL} for all input pins (except MN/$\overline{MX}$ pin) tested with MN/$\overline{MX}$ pin = GND.
5. V_{IH} tested with MN/$\overline{MX}$ pin = V_{CC}.

A.C. CHARACTERISTICS

(T_A = 0°C to 70°C, T_{CASE} (Plastic) = 0°C to 80°C, T_{CASE} (PLCC) = 0°C to 85°C)
(V_{CC} = 5V ±10% for 80C86AL, V_{CC} = 5V ±5% for 80C86AL-2)

MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

Symbol	Parameter	80C86AL		80C86AL-2		Units	Test Conditions
		Min	Max	Min	Max		
TCLCL	CLK Cycle Period	200	D.C.	125	D.C.	ns	
TCLCH	CLK Low Time	118		68		ns	
TCHCL	CLK High Time	69		44		ns	
TCH1CH2	CLK Rise Time		10		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10		10	ns	From 3.5V to 1.0V
TDVCL	Data in Setup Time	30		20		ns	
TCLDX	Data in Hold Time	10		10		ns	
TR1VCL	RDY Setup Time into 82C84A (Notes 1, 2)	35		35		ns	
TCLR1X	RDY Hold Time into 82C84A (Notes 1, 2)	0		0		ns	
TRYHCH	READY Setup Time into 80C86AL	118		68		ns	
TCHRYX	READY Hold Time into 80C86AL	30		20		ns	
TRYLCL	READY Inactive to CLK (Note 3)	−8		−8		ns	
THVCH	HOLD Setup Time	35		20		ns	
TINVCH	INTR, NMI, $\overline{\text{TEST}}$ Setup Time (Note 2)	30		15		ns	
TILIH	Input Rise Time (Except CLK)		15		15	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK)		15		15	ns	From 2.0V to 0.8V

A.C. CHARACTERISTICS (Continued)

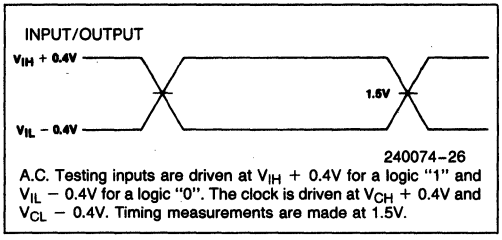
(T_A = 0°C to 70°C, T_{CASE} (Plastic) = 0°C to 80°C, T_{CASE} (PLCC) = 0°C to 85°C)
(V_{CC} = 5V ±10% for 80C86AL, V_{CC} = 5V ±5% for 80C86AL-2)

Timing Responses

Symbol	Parameter	80C86AL		80C86AL-2		Units	Test Conditions
		Min	Max	Min	Max		
TCLAV	Address Valid Delay	10	110	10	60	ns	
TCLAX	Address Hold Time	10		10		ns	
TCLAZ	Address Float Delay	TCLAX	80	TCLAX	50	ns	
TLHLL	ALE Width	TCLCH−20		TCLCH−10		ns	
TCLLH	ALE Active Delay		80		50	ns	
TCHLL	ALE Inactive Delay		85		55	ns	
TLLAX	Address Hold Time to ALE Inactive	TCHCL−10		TCHCL−10		ns	
TCLDV	Data Valid Delay	10	110	10	60	ns	
TCHDX	Data Hold Time	10		10		ns	
TWHDX	Data Hold Time After WR	TCLCH−30		TCLCH−30		ns	
TCVCTV	Control Active Delay 1	10	110	10	70	ns	
TCHCTV	Control Active Delay 2	10	110	10	60	ns	
TCVCTX	Control Inactive Delay	10	110	10	70	ns	
TAZRL	Address Float to READ Active	0		0		ns	
TCLRL	$\overline{RD}$ Active Delay	10	165	10	100	ns	
TCLRH	$\overline{RD}$ Inactive Delay	10	150	10	80	ns	
TRHAV	$\overline{RD}$ Inactive to Next Address Active	TCLCL−45		TCLCL−40		ns	
TCLHAV	HLDA Valid Delay	10	160	10	100	ns	
TRLRH	$\overline{RD}$ Width	2TCLCL−75		2TCLCL−50		ns	
TWLWH	$\overline{WR}$ Width	2TCLCL−60		2TCLCL−40		ns	
TAVAL	Address Valid to ALE Low	TCLCH−60		TCLCH−40		ns	
TOLOH	Output Rise Time (Note 4)		15		15	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time (Note 4)		15		15	ns	From 2.0V to 0.8V

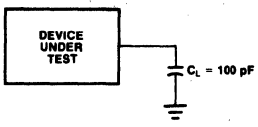
NOTES:
1. Signal at 82C84A shown for reference only. See 82C84A data sheet for the most recent specifications.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T2 state. (8 ns into T3).
4. These parameters are characterized and not 100% tested.

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. Testing inputs are driven at V_{IH} + 0.4V for a logic "1" and V_{IL} − 0.4V for a logic "0". The clock is driven at V_{CH} + 0.4V and V_{CL} − 0.4V. Timing measurements are made at 1.5V.

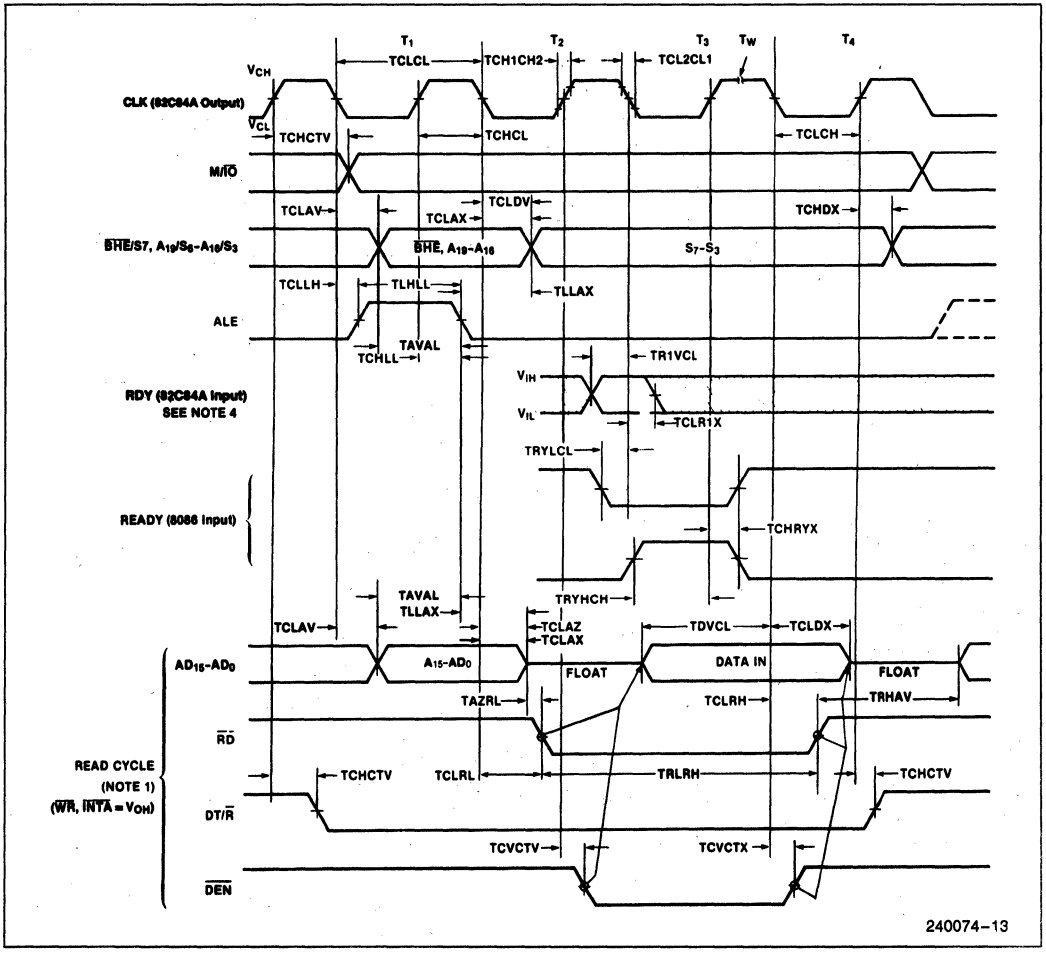
A.C. TESTING LOAD CIRCUIT



C_L Includes Jig Capacitance

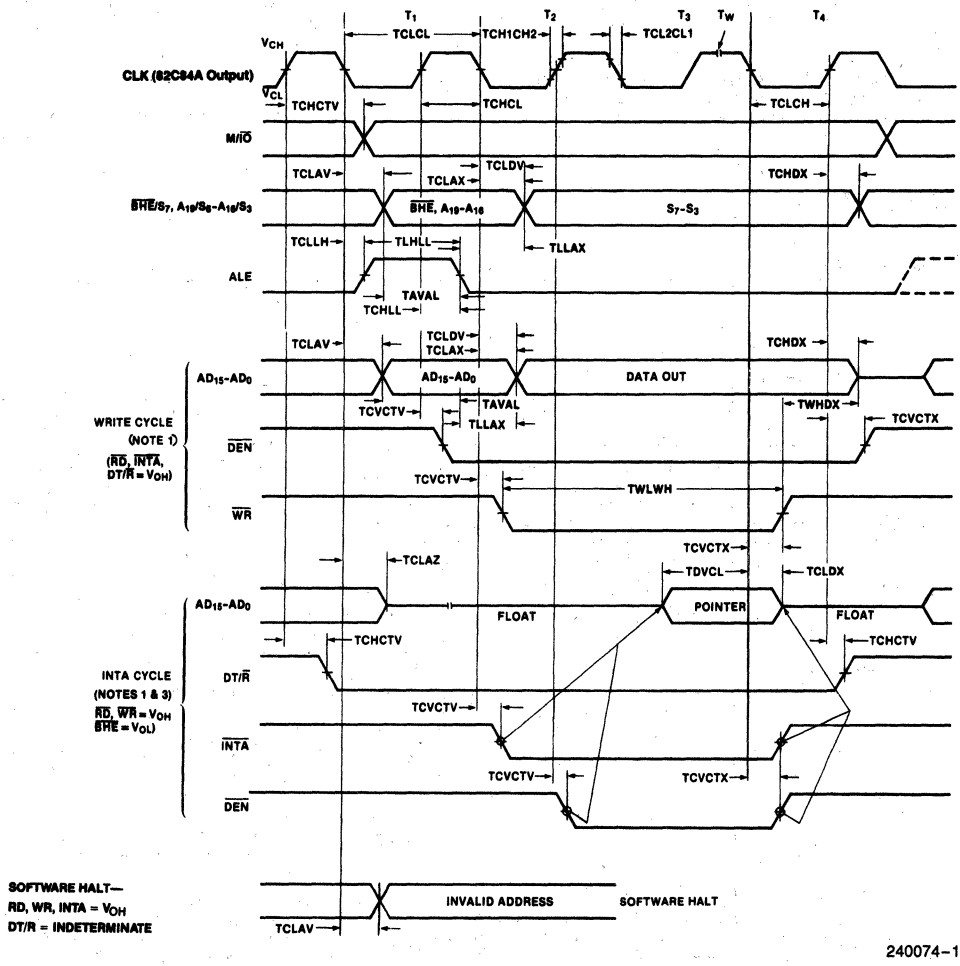
WAVEFORMS

MINIMUM MODE



WAVEFORMS (Continued)

MINIMUM MODE (Continued)



NOTES:
1. All output timing measurements are made at 1.5V unless otherwise noted.
2. RDY is sampled near the end of T_2, T_3, T_W to determine if T_W machines states are to be inserted.
3. Two INTA cycles run back-to-back. The 80C86AL local ADDR/DATA BUS is floating during both INTA cycles. Control signals shown for second INTA cycle.
4. Signals at 82C84A are shown for reference only.

A.C. CHARACTERISTICS

MAX MODE SYSTEM (USING 82C88 BUS CONTROLLER) TIMING REQUIREMENTS

Symbol	Parameter	80C86AL		80C86AL-2		Units	Test Conditions
		Min	Max	Min	Max		
TCLCL	CLK Cycle Period	200	D.C.	125	D.C.	ns	
TCLCH	CLK Low Time	118		68		ns	
TCHCL	CLK High Time	69		44		ns	
TCH1CH2	CLK Rise Time		10		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10		10	ns	From 3.5V to 1.0V
TDVCL	Data in Setup Time	30		20		ns	
TCLDX	Data in Hold Time	10		10		ns	
TR1VCL	RDY Setup Time into 82C84A (Notes 1, 2)	35		35		ns	
TCLR1X	RDY Hold Time into 82C84A (Notes 1, 2)	0		0		ns	
TRYHCH	READY Setup Time into 80C86AL	118		68		ns	
TCHRYX	READY Hold Time into 80C86AL	30		20		ns	
TRYLCL	READY Inactive to CLK (Note 4)	−8		−8		ns	
TINVCH	Setup Time for Recognition (INTR, NMI, $\overline{TEST}$) (Note 2)	30		15		ns	
TGVCH	$\overline{RQ}/\overline{GT}$ Setup Time	30		15		ns	
TCHGX	$\overline{RQ}$ Hold Time into 80C86AL	40		30		ns	
TILIH	Input Rise Time (Except CLK) (Note 5)		15		15	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK) (Note 5)		15		15	ns	From 2.0V to 0.8V

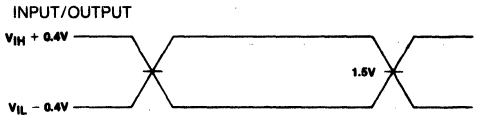
A.C. CHARACTERISTICS (Continued)

TIMING RESPONSES

Symbol	Parameter	80C86AL		80C86AL-2		Units	Test Conditions
		Min	Max	Min	Max		
TCLML	Command Active Delay (Note 1)	5	45	5	35	ns	
TCLMH	Command Inactive Delay (Note 1)	5	45	5	35	ns	
TRYHSH	READY Active to Status Passive (Note 3)		110		65	ns	
TCHSV	Status Active Delay	10	110	10	60	ns	
TCLSH	Status Inactive Delay	10	130	10	70	ns	
TCLAV	Address Valid Delay	10	110	10	60	ns	
TCLAX	Address Hold Time	10		10		ns	
TCLAZ	Address Float Delay	TCLAX	80	TCLAX	50	ns	
TSVLH	Status Valid to ALE High (Note 1)		35		20	ns	
TSVMCH	Status Valid to MCE High (Note 1)		35		30	ns	
TCLLH	CLK Low to ALE Valid (Note 1)		35		20	ns	
TCLMCH	CLK Low to MCE High (Note 1)		35		25	ns	
TCHLL	ALE Inactive Delay (Note 1)	4	35	4	25	ns	
TCLDV	Data Valid Delay	10	110	10	60	ns	
TCHDX	Data Hold Time	10		10		ns	
TCVNV	Control Active Delay (Note 1)	5	45	5	45	ns	
TCVNX	Control Inactive Delay (Note 1)	5	45	10	45	ns	
TAZRL	Address Float to Read Active	0		0		ns	
TCLRL	RD Active Delay	10	165	10	100	ns	
TCLRH	RD Inactive Delay	10	150	10	80	ns	
TRHAV	RD Inactive to Next Address Active	TCLCL−45		TCLCL−40		ns	
TCHDTL	Direction Control Active Delay (Note 1)		50		50	ns	
TCHDTH	Direction Control Inactive Delay (Note 1)		35		30	ns	
TCLGL	GT Active Delay	0	85	0	50	ns	
TCLGH	GT Inactive Delay	0	85	0	50	ns	
TRLRH	RD Width	2TCLCL−75		2TCLCL−50		ns	
TOLOH	Output Rise Time		15		15	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time		15		15	ns	From 2.0V to 0.8V

NOTES:
1. Signal at 82C84A or 82C88 shown for reference only. See 82C84A and 82C88 for the most recent specifications.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T3 and wait states.
4. Applies only to T2 state (8 ns into T3).
5. These parameters are characterized and not 100% tested.

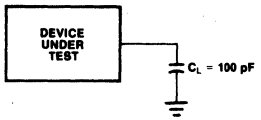
A.C. TESTING INPUT, OUTPUT WAVEFORM



240074-11

A.C. Testing inputs are driven at V_{IH} + 0.4V for a logic "1" and V_{IL} − 0.4V for a logic "0". The clock is driven at V_{CH} + 0.4V and V_{CL} − 0.4V. Timing measurements are made at 1.5V.

A.C. TESTING LOAD CIRCUIT

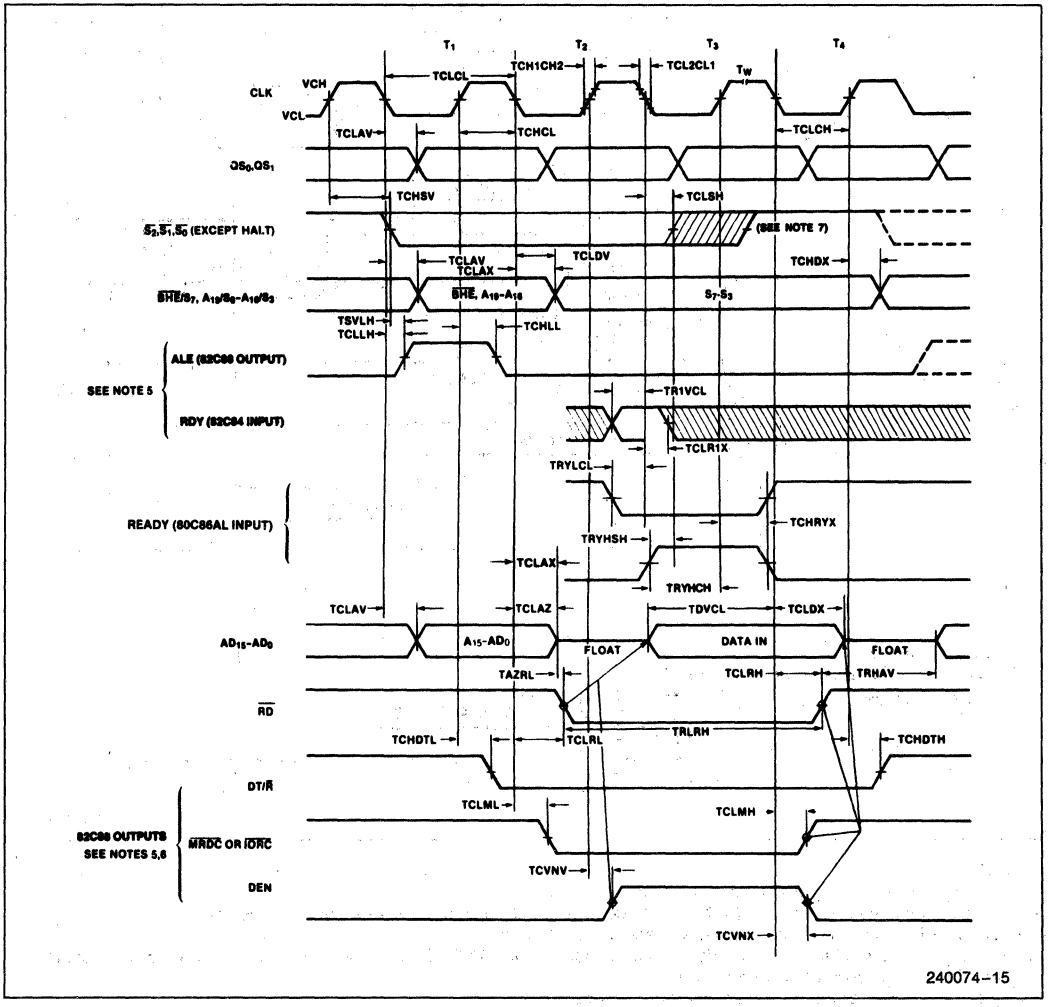


240074-12

C_L Includes Jig Capacitance

WAVEFORMS

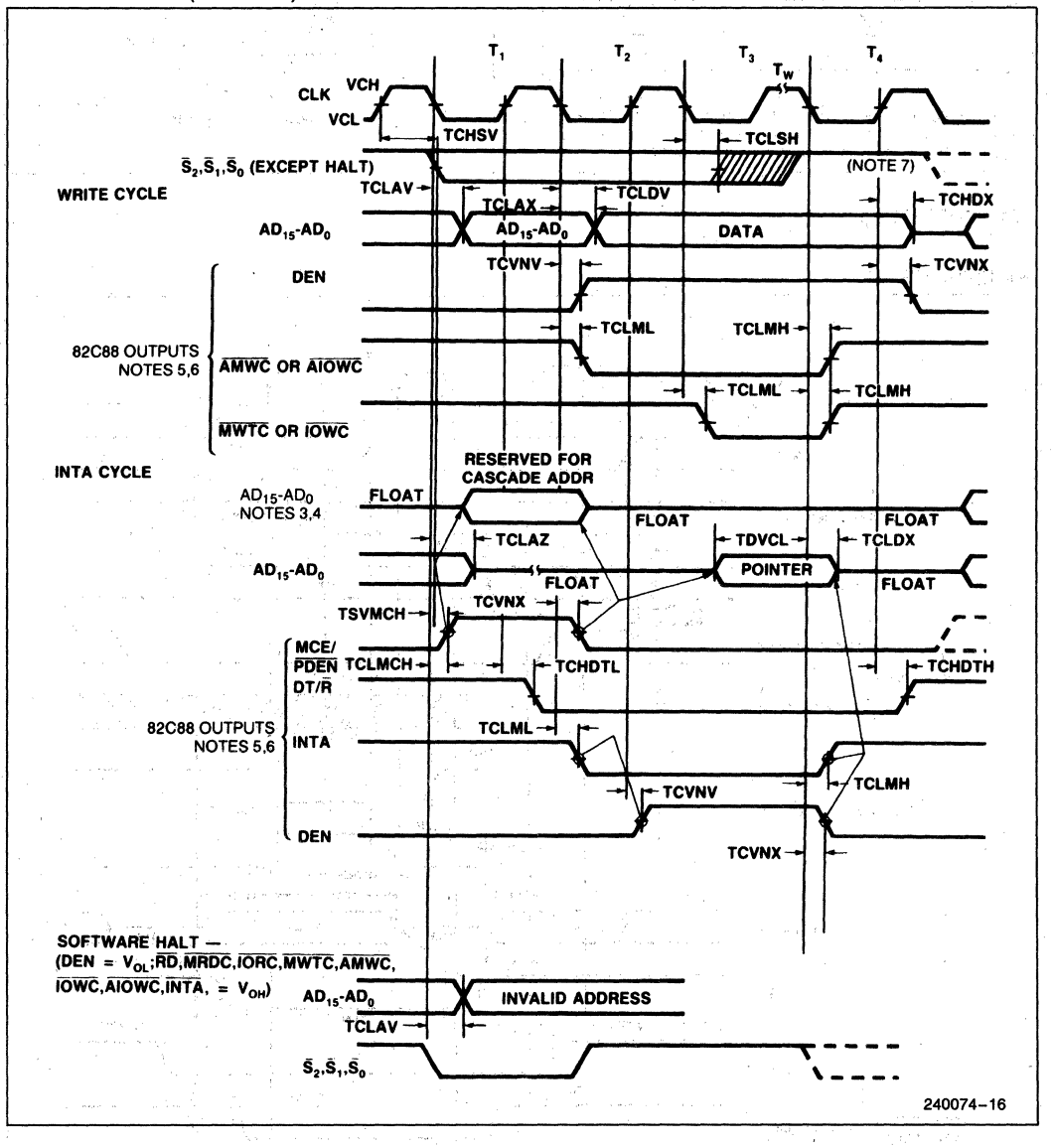
MAXIMUM MODE



240074-15

WAVEFORMS (Continued)

MAXIMUM MODE (Continued)

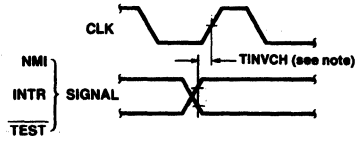


240074-16

NOTES:
1. All timing measurements are made at 1.5V unless otherwise noted.
2. RDY is sampled near the end of T_2, T_3, T_W to determine if T_W machines states are to be inserted.
3. Cascade address is valid between first and second INTA cycle.
4. Two INTA cycles run back-to-back. The 80C86AL local ADDR/DATA BUS is floating during both INTA cycles. Control for pointer address is shown for second INTA cycle.
5. Signals at 82C84A or 82C88 are shown for reference only.
6. The issuance of the 82C88 command and control signals ($\overline{MRDC}$, $\overline{MWTC}$, $\overline{AMWC}$, $\overline{IORC}$, $\overline{IOWC}$, $\overline{AIOWC}$, $\overline{INTA}$ and DEN) lags the active high 82C88 CEN.
7. Status inactive in state just prior to T_4.

WAVEFORMS (Continued)

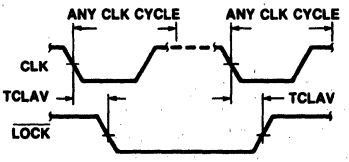
ASYNCHRONOUS SIGNAL RECOGNITION



240074-17

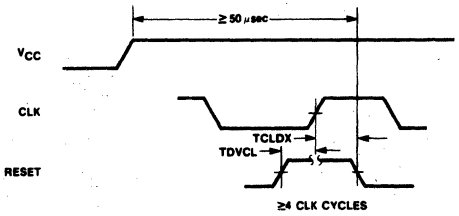
NOTE: Setup requirements for asynchronous signals only to guarantee recognition at next CLK.

BUS LOCK SIGNAL TIMING (MAXIMUM MODE ONLY)



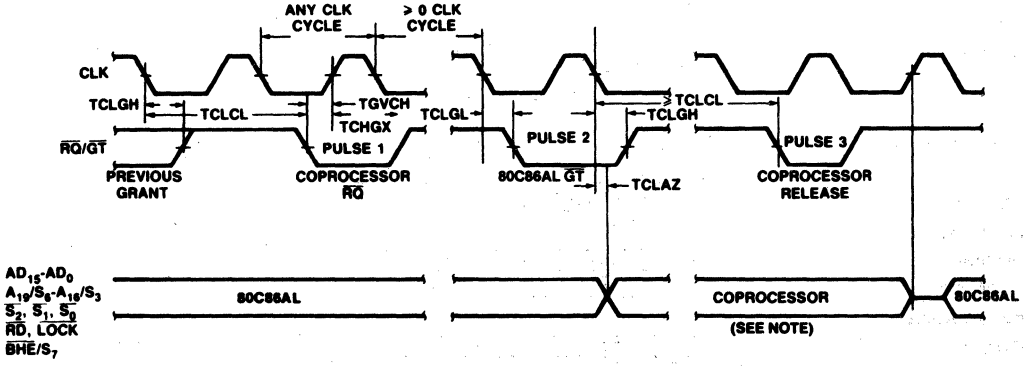
240074-18

RESET TIMING



240074-19

REQUEST/GRANT SEQUENCE TIMING (MAXIMUM MODE ONLY)

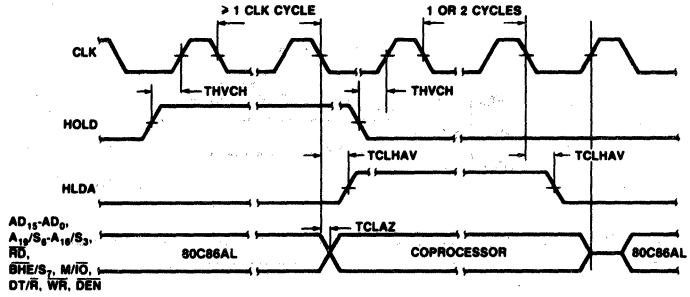


240074-20

NOTE: The coprocessor may not drive the buses outside the region shown without risking contention.

WAVEFORMS (Continued)

HOLD/HOLD ACKNOWLEDGE TIMING (MINIMUM MODE ONLY)



240074-21

Table 2. Instruction Set Summary

Mnemonic and Description	Instruction Code			
DATA TRANSFER				
MOV = Move:	76543210	76543210	76543210	76543210
Register/Memory to/from Register**	100010dw	mod reg r/m		
Immediate to Register/Memory	1100011w	mod 000 r/m	data	data if w = 1
Immediate to Register	1011w reg	data	data if w = 1	
Memory to Accumulator	1010000w	addr-low	addr-high	
Accumulator to Memory	1010001w	addr-low	addr-high	
Register/Memory to Segment Register**	10001110	mod 0 reg r/m		
Segment Register to Register/Memory	10001100	mod 0 reg r/m		
PUSH = Push:				
Register/Memory	11111111	mod 110 r/m		
Register	01010 reg			
Segment Register	000 reg 110			
POP = Pop:				
Register/Memory	10001111	mod 000 r/m		
Register	01011 reg			
Segment Register	000 reg 111			
XCHG = Exchange:				
Register/Memory with Register	1000011w	mod reg r/m		
Register with Accumulator	10010 reg			
IN = Input from:				
Fixed Port	1110010w	port		
Variable Port	1110110w			
OUT = Output to:				
Fixed Port	1110011w	port		
Variable Port	1110111w			
XLAT = Translate Byte to AL	11010111			
LEA = Load EA to Register	10001101	mod reg r/m		
LDS = Load Pointer to DS	11000101	mod reg r/m		
LES = Load Pointer to ES	11000100	mod reg r/m		
LAHF = Load AH with Flags	10011111			
SAHF = Store AH into Flags	10011110			
PUSHF = Push Flags	10011100			
POPF = Pop Flags	10011101			

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
ARITHMETIC	76543210	76543210	76543210	76543210
ADD = Add:				
Reg./Memory with Register to Either	000000dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod 000 r/m	data	data if s w = 01
Immediate to Accumulator	0000010w	data	data if w = 1	
ADC = Add with Carry:				
Reg./Memory with Register to Either	000100dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod 010 r/m	data	data if s w = 01
Immediate to Accumulator	0001010w	data	data if w = 1	
INC = Increment:				
Register/Memory	1111111w	mod 000 r/m		
Register	01000reg			
AAA = ASCII Adjust for Add	00110111			
DAA = Decimal Adjust for Add	00100111			
SUB = Subtract:				
Reg./Memory and Register to Either	001010dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod 101 r/m	data	data if s w = 01
Immediate from Accumulator	0010110w	data	data if w = 1	
SBB = Subtract with Borrow				
Reg./Memory and Register to Either	000110dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod 011 r/m	data	data if s w = 01
Immediate from Accumulator	0001110w	data	data if w = 1	
DEC = Decrement:				
Register/Memory	1111111w	mod 001 r/m		
Register	01001 reg			
NEG = Change Sign	1111011w	mod 011 r/m		
CMP = Compare:				
Register/Memory and Register	001110dw	mod reg r/m		
Immediate with Register/Memory	100000sw	mod 111 r/m	data	data if s w = 01
Immediate with Accumulator	0011110w	data	data if w = 1	
AAS = ASCII Adjust for Subtract	00111111			
DAS = Decimal Adjust for Subtract	00101111			
MUL = Multiply (Unsigned)	1111011w	mod 100 r/m		
IMUL = Integer Multiply (Signed)	1111011w	mod 101 r/m		
AAM = ASCII Adjust for Multiply	11010100	00001010		
DIV = Divide (Unsigned)	1111011w	mod 110 r/m		
IDIV = Integer Divide (Signed)	1111011w	mod 111 r/m		
AAD = ASCII Adjust for Divide	11010101	00001010		
CBW = Convert Byte to Word	10011000			
CWD = Convert Word to Double Word	10011001			

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
LOGIC	76543210	76543210	76543210	76543210
NOT = Invert	1111011w	mod 010 r/m		
SHL/SAL = Shift Logical/Arithmetic Left	110100vw	mod 100 r/m		
SHR = Shift Logical Right	110100vw	mod 101 r/m		
SAR = Shift Arithmetic Right	110100vw	mod 111 r/m		
ROL = Rotate Left	110100vw	mod 000 r/m		
ROR = Rotate Right	110100vw	mod 001 r/m		
RCL = Rotate Through Carry Flag Left	110100vw	mod 010 r/m		
RCR = Rotate Through Carry Right	110100vw	mod 011 r/m		
AND = **And:**				
Reg./Memory and Register to Either	001000dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 100 r/m	data	data if w = 1
Immediate to Accumulator	0010010w	data	data if w = 1	
TEST = **And Function to Flags, No Result:**				
Register/Memory and Register	1000010w	mod reg r/m		
Immediate Data and Register/Memory	1111011w	mod 000 r/m	data	data if w = 1
Immediate Data and Accumulator	1010100w	data	data if w = 1	
OR = **Or:**				
Reg./Memory and Register to Either	000010dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 001 r/m	data	data if w = 1
Immediate to Accumulator	0000110w	data	data if w = 1	
XOR = **Exclusive OR:**				
Reg./Memory and Register to Either	001100dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 110 r/m	data	data if w = 1
Immediate to Accumulator	0011010w	data	data if w = 1	
STRING MANIPULATION				
REP = Repeat	1111001z			
MOVS = Move Byte/Word	1010010w			
CMPS = Compare Byte/Word	1010011w			
SCAS = Scan Byte/Word	1010111w			
LODS = Load Byte/Wd to AL/AX	1010110w			
STOS = Stor Byte/Wd from AL/A	1010101w			
CONTROL TRANSFER				
CALL = **Call:**				
Direct Within Segment	11101000	disp-low	disp-high	
Indirect Within Segment	11111111	mod 010 r/m		
Direct Intersegment	10011010	offset-low	offset-high	
		seg-low	seg-high	
Indirect Intersegment	11111111	mod 011 r/m		

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code		
CONTROL TRANSFER (Continued) **JMP = Unconditional Jump:**	76543210	76543210	76543210
Direct Within Segment	11101001	disp-low	disp-high
Direct Within Segment-Short	11101011	disp	
Indirect Within Segment	11111111	mod 100 r/m	
Direct Intersegment	11101010	offset-low	offset-high
		seg-low	seg-high
Indirect Intersegment	11111111	mod 101 r/m	
RET = Return from CALL:			
Within Segment	11000011		
Within Seg. Adding Immed to SP	11000010	data-low	data-high
Intersegment	11001011		
Intersegment Adding Immediate to SP	11001010	data-low	data-high
JE/JZ = Jump on Equal/Zero	01110100	disp	
JL/JNGE = Jump on Less/Not Greater or Equal	01111100	disp	
JLE/JNG = Jump on Less or Equal/ Not Greater	01111110	disp	
JB/JNAE = Jump on Below/Not Above or Equal	01110010	disp	
JBE/JNA = Jump on Below or Equal/ Not Above	01110110	disp	
JP/JPE = Jump on Parity/Parity Even	01111010	disp	
JO = Jump on Overflow	01110000	disp	
JS = Jump on Sign	01111000	disp	
JNE/JNZ = Jump on Not Equal/Not Zero	01110101	disp	
JNL/JGE = Jump on Not Less/Greater or Equal	01111101	disp	
JNLE/JG = Jump on Not Less or Equal/ Greater	01111111	disp	
JNB/JAE = Jump on Not Below/Above or Equal	01110011	disp	
JNBE/JA = Jump on Not Below or Equal/Above	01110111	disp	
JNP/JPO = Jump on Not Par/Par Odd	01111011	disp	
JNO = Jump on Not Overflow	01110001	disp	
JNS = Jump on Not Sign	01111001	disp	
LOOP = Loop CX Times	11100010	disp	
LOOPZ/LOOPE = Loop While Zero/Equal	11100001	disp	
LOOPNZ/LOOPNE = Loop While Not Zero/Equal	11100000	disp	
JCXZ = Jump on CX Zero	11100011	disp	
INT = Interrupt			
Type Specified	11001101	type	
Type 3	11001100		
INTO = Interrupt on Overflow	11001110		
IRET = Interrupt Return	11001111		

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code	
PROCESSOR CONTROL	76543210	76543210
CLC = Clear Carry	11111000	
CMC = Complement Carry	11110101	
STC = Set Carry	11111001	
CLD = Clear Direction	11111100	
STD = Set Direction	11111101	
CLI = Clear Interrupt	11111010	
STI = Set Interrupt	11111011	
HLT = Halt	11110100	
WAIT = Wait	10011011	
ESC = Escape (to External Device)	11011xxx	mod x x x r/m
LOCK = Bus Lock Prefix	11110000	

NOTES:
AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value.
Greater = more positive:
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
if w = 1 then word instruction; if w = 0 then byte instruction
if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16 bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low
if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)
*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.
**MOV CS, REG/MEMORY not allowed.
if s w = 01 then 16 bits of immediate data form the operand
if s w = 11 then an immediate data byte is sign extended to form the 16-bit operand
if v = 0 then "count" = 1; if v = 1 then "count" in (CL) register
x = don't care
z is used for string primitives for comparison with ZF FLAG

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:
FLAGS =
X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

DATA SHEET REVISION REVIEW

The following list represents key differences between this and the -001 data sheet. Please review this summary carefully.

1. In the Pin Description Table (Table 1), the description of the HLDA signal being issued has been corrected. HLDA will be issued in the middle of either the T_4 or T_i state.

8088
8-BIT HMOS MICROPROCESSOR
8088/8088-2

- 8-Bit Data Bus Interface
- 16-Bit Internal Architecture
- Direct Addressing Capability to 1 Mbyte of Memory
- Direct Software Compatibility with 8086 CPU
- 14-Word by 16-Bit Register Set with Symmetrical Operations
- 24 Operand Addressing Modes
- Byte, Word, and Block Operations
- 8-Bit and 16-Bit Signed and Unsigned Arithmetic in Binary or Decimal, Including Multiply and Divide
- Two Clock Rates:
  - 5 MHz for 8088
  - 8 MHz for 8088-2
- Available in EXPRESS
  - Standard Temperature Range
  - Extended Temperature Range

The Intel® 8088 is a high performance microprocessor implemented in N-channel, depletion load, silicon gate technology (HMOS), and packaged in a 40-pin CERDIP package. The processor has attributes of both 8- and 16-bit microprocessors. It is directly compatible with 8086 software and 8080/8085 hardware and peripherals.

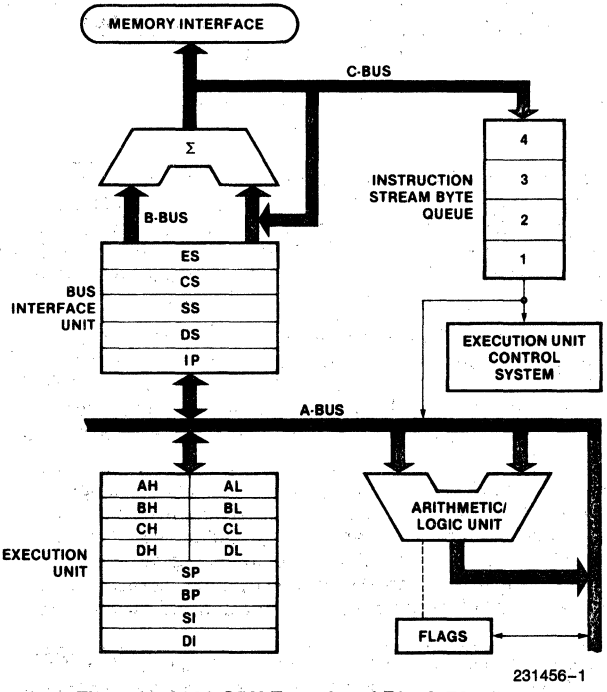


Figure 1. 8088 CPU Functional Block Diagram

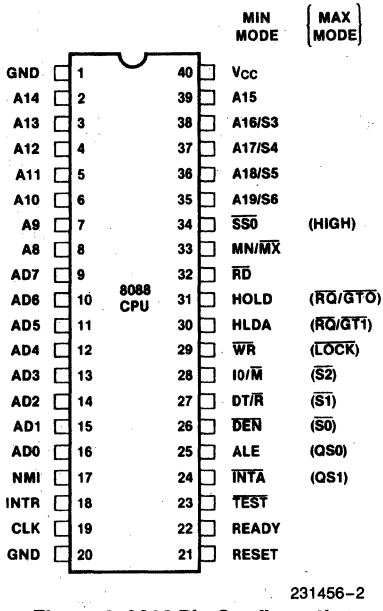


Figure 2. 8088 Pin Configuration

October 1988
Order Number: 231456-004

Table 1. Pin Description

The following pin function descriptions are for 8088 systems in either minimum or maximum mode. The "local bus" in these descriptions is the direct multiplexed bus interface connection to the 8088 (without regard to additional bus buffers).

<table>
<tr><th>Symbol</th><th>Pin No.</th><th>Type</th><th>Name and Function</th></tr>
<tr><td>AD7–AD0</td><td>9–16</td><td>I/O</td><td>ADDRESS DATA BUS: These lines constitute the time multiplexed memory/IO address (T1) and data (T2, T3, Tw, T4) bus. These lines are active HIGH and float to 3-state OFF during interrupt acknowledge and local bus "hold acknowledge".</td></tr>
<tr><td>A15–A8</td><td>2–8, 39</td><td>O</td><td>ADDRESS BUS: These lines provide address bits 8 through 15 for the entire bus cycle (T1–T4). These lines do not have to be latched by ALE to remain valid. A15–A8 are active HIGH and float to 3-state OFF during interrupt acknowledge and local bus "hold acknowledge".</td></tr>
<tr><td>A19/S6, A18/S5, A17/S4, A16/S3</td><td>35–38</td><td>O</td><td>ADDRESS/STATUS: During T1, these are the four most significant address lines for memory operations. During I/O operations, these lines are LOW. During memory and I/O operations, status information is available on these lines during T2, T3, Tw, and T4. S6 is always low. The status of the interrupt enable flag bit (S5) is updated at the beginning of each clock cycle. S4 and S3 are encoded as shown.
This information indicates which segment register is presently being used for data accessing.
These lines float to 3-state OFF during local bus "hold acknowledge".
<table>
<tr><th>S4</th><th>S3</th><th>Characteristics</th></tr>
<tr><td>0 (LOW)</td><td>0</td><td>Alternate Data</td></tr>
<tr><td>0</td><td>1</td><td>Stack</td></tr>
<tr><td>1 (HIGH)</td><td>0</td><td>Code or None</td></tr>
<tr><td>1</td><td>1</td><td>Data</td></tr>
<tr><td>S6 is 0 (LOW)</td><td></td><td></td></tr>
</table></td></tr>
<tr><td>$\overline{RD}$</td><td>32</td><td>O</td><td>READ: Read strobe indicates that the processor is performing a memory or I/O read cycle, depending on the state of the IO/$\overline{M}$ pin or S2. This signal is used to read devices which reside on the 8088 local bus. $\overline{RD}$ is active LOW during T2, T3 and Tw of any read cycle, and is guaranteed to remain HIGH in T2 until the 8088 local bus has floated.
This signal floats to 3-state OFF in "hold acknowledge".</td></tr>
<tr><td>READY</td><td>22</td><td>I</td><td>READY: is the acknowledgement from the addressed memory or I/O device that it will complete the data transfer. The RDY signal from memory or I/O is synchronized by the 8284 clock generator to form READY. This signal is active HIGH. The 8088 READY input is not synchronized. Correct operation is not guaranteed if the set up and hold times are not met.</td></tr>
<tr><td>INTR</td><td>18</td><td>I</td><td>INTERRUPT REQUEST: is a level triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored to via an interrupt vector lookup table located in system memory. It can be internally masked by software resetting the interrupt enable bit. INTR is internally synchronized. This signal is active HIGH.</td></tr>
<tr><td>$\overline{TEST}$</td><td>23</td><td>I</td><td>$\overline{TEST}$: input is examined by the "wait for test" instruction. If the $\overline{TEST}$ input is LOW, execution continues, otherwise the processor waits in an "idle" state. This input is synchronized internally during each clock cycle on the leading edge of CLK.</td></tr>
</table>

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
NMI	17	I	**NON-MASKABLE INTERRUPT:** is an edge triggered input which causes a type 2 interrupt. A subroutine is vectored to via an interrupt vector lookup table located in system memory. NMI is not maskable internally by software. A transition from a LOW to HIGH initiates the interrupt at the end of the current instruction. This input is internally synchronized.
RESET	21	I	**RESET:** causes the processor to immediately terminate its present activity. The signal must be active HIGH for at least four clock cycles. It restarts execution, as described in the instruction set description, when RESET returns LOW. RESET is internally synchronized.
CLK	19	I	**CLOCK:** provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing.
V_{CC}	40		**V_{CC}:** is the +5V ±10% power supply pin.
GND	1, 20		**GND:** are the ground pins.
MN/$\overline{MX}$	33	I	**MINIMUM/MAXIMUM:** indicates what mode the processor is to operate in. The two modes are discussed in the following sections.

The following pin function descriptions are for the 8088 minimum mode (i.e., MN/$\overline{MX}$ = V_{CC}). Only the pin functions which are unique to minimum mode are described; all other pin functions are as described above.

Symbol	Pin No.	Type	Name and Function
IO/$\overline{M}$	28	O	**STATUS LINE:** is an inverted maximum mode $\overline{S2}$. It is used to distinguish a memory access from an I/O access. IO/$\overline{M}$ becomes valid in the T4 preceding a bus cycle and remains valid until the final T4 of the cycle (I/O = HIGH, M = LOW). IO/$\overline{M}$ floats to 3-state OFF in local bus "hold acknowledge".
$\overline{WR}$	29	O	**WRITE:** strobe indicates that the processor is performing a write memory or write I/O cycle, depending on the state of the IO/$\overline{M}$ signal. WR is active for T2, T3, and Tw of any write cycle. It is active LOW, and floats to 3-state OFF in local bus "hold acknowledge".
$\overline{INTA}$	24	O	**INTA:** is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T2, T3, and Tw of each interrupt acknowledge cycle.
ALE	25	O	**ADDRESS LATCH ENABLE:** is provided by the processor to latch the address into an address latch. It is a HIGH pulse active during clock low of T1 of any bus cycle. Note that ALE is never floated.
DT/$\overline{R}$	27	O	**DATA TRANSMIT/RECEIVE:** is needed in a minimum system that desires to use a data bus transceiver. It is used to control the direction of data flow through the transceiver. Logically, DT/$\overline{R}$ is equivalent to $\overline{S1}$ in the maximum mode, and its timing is the same as for IO/$\overline{M}$ (T = HIGH, R = LOW). This signal floats to 3-state OFF in local "hold acknowledge".
$\overline{DEN}$	26	O	**DATA ENABLE:** is provided as an output enable for the data bus transceiver in a minimum system which uses the transceiver. $\overline{DEN}$ is active LOW during each memory and I/O access, and for $\overline{INTA}$ cycles. For a read or $\overline{INTA}$ cycle, it is active from the middle of T2 until the middle of T4, while for a write cycle, it is active from the beginning of T2 until the middle of T4. $\overline{DEN}$ floats to 3-state OFF during local bus "hold acknowledge".

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
HOLD, HLDA	31, 30	I, O	**HOLD:** indicates that another master is requesting a local bus "hold". To be acknowledged, HOLD must be active HIGH. The processor receiving the "hold" request will issue HLDA (HIGH) as an acknowledgement, in the middle of a T4 or Ti clock cycle. Simultaneous with the issuance of HLDA the processor will float the local bus and control lines. After HOLD is detected as being LOW, the processor lowers HLDA, and when the processor needs to run another cycle, it will again drive the local bus and control lines. Hold is not an asynchronous input. External synchronization should be provided if the system cannot otherwise guarantee the set up time.
$\overline{SSO}$	34	O	**STATUS LINE:** is logically equivalent to $\overline{S0}$ in the maximum mode. The combination of $\overline{SSO}$, IO/$\overline{M}$ and DT/$\overline{R}$ allows the system to completely decode the current bus cycle status.

IO/$\overline{M}$	DT/$\overline{R}$	$\overline{SSO}$	Characteristics
1(HIGH)	0	0	Interrupt Acknowledge
1	0	1	Read I/O Port
1	1	0	Write I/O Port
1	1	1	Halt
0(LOW)	0	0	Code Access
0	0	1	Read Memory
0	1	0	Write Memory
0	1	1	Passive

The following pin function descriptions are for the 8088/8288 system in maximum mode (i.e., MN/$\overline{MX}$ = GND). Only the pin functions which are unique to maximum mode are described; all other pin functions are as described above.

Symbol	Pin No.	Type	Name and Function
$\overline{S2}$, $\overline{S1}$, $\overline{S0}$	26–28	O	**STATUS:** is active during clock high of T4, T1, and T2, and is returned to the passive state (1,1,1) during T3 or during Tw when READY is HIGH. This status is used by the 8288 bus controller to generate all memory and I/O access control signals. Any change by $\overline{S2}$, $\overline{S1}$, or $\overline{S0}$ during T4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T3 and Tw is used to indicate the end of a bus cycle. These signals float to 3-state OFF during "hold acknowledge". During the first clock cycle after RESET becomes active, these signals are active HIGH. After this first clock, they float to 3-state OFF.

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Characteristics
0(LOW)	0	0	Interrupt Acknowledge
0	0	1	Read I/O Port
0	1	0	Write I/O Port
0	1	1	Halt
1(HIGH)	0	0	Code Access
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Passive

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
$\overline{RQ}/\overline{GT0}$, $\overline{RQ}/\overline{GT1}$	30, 31	I/O	**REQUEST/GRANT:** pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional with $\overline{RQ}/\overline{GT0}$ having higher priority than $\overline{RQ}/\overline{GT1}$. $\overline{RQ}/\overline{GT}$ has an internal pull-up resistor, so may be left unconnected. The request/grant sequence is as follows (See Figure 8): 1. A pulse of one CLK wide from another local bus master indicates a local bus request ("hold") to the 8088 (pulse 1). 2. During a T4 or TI clock cycle, a pulse one clock wide from the 8088 to the requesting master (pulse 2), indicates that the 8088 has allowed the local bus to float and that it will enter the "hold acknowledge" state at the next CLK. The CPU's bus interface unit is disconnected logically from the local bus during "hold acknowledge". The same rules as for HOLD/HOLDA apply as for when the bus is released. 3. A pulse one CLK wide from the requesting master indicates to the 8088 (pulse 3) that the "hold" request is about to end and that the 8088 can reclaim the local bus at the next CLK. The CPU then enters T4. Each master-master exchange of the local bus is a sequence of three pulses. There must be one idle CLK cycle after each bus exchange. Pulses are active LOW. If the request is made while the CPU is performing a memory cycle, it will release the local bus during T4 of the cycle when all the following conditions are met: 1. Request occurs on or before T2. 2. Current cycle is not the low bit of a word. 3. Current cycle is not the first acknowledge of an interrupt acknowledge sequence. 4. A locked instruction is not currently executing. If the local bus is idle when the request is made the two possible events will follow: 1. Local bus will be released during the next clock. 2. A memory cycle will start within 3 clocks. Now the four rules for a currently active memory cycle apply with condition number 1 already satisfied.
$\overline{LOCK}$	29	O	**$\overline{LOCK}$:** indicates that other system bus masters are not to gain control of the system bus while $\overline{LOCK}$ is active (LOW). The $\overline{LOCK}$ signal is activated by the "LOCK" prefix instruction and remains active until the completion of the next instruction. This signal is active LOW, and floats to 3-state off in "hold acknowledge".
QS1, QS0	24, 25	O	**QUEUE STATUS:** provide status to allow external tracking of the internal 8088 instruction queue. The queue status is valid during the CLK cycle after which the queue operation is performed.

QS1	QS0	Characteristics
0(LOW)	0	No Operation
0	1	First Byte of Opcode from Queue
1(HIGH)	0	Empty the Queue
1	1	Subsequent Byte from Queue

—	34	O	Pin 34 is always high in the maximum mode.

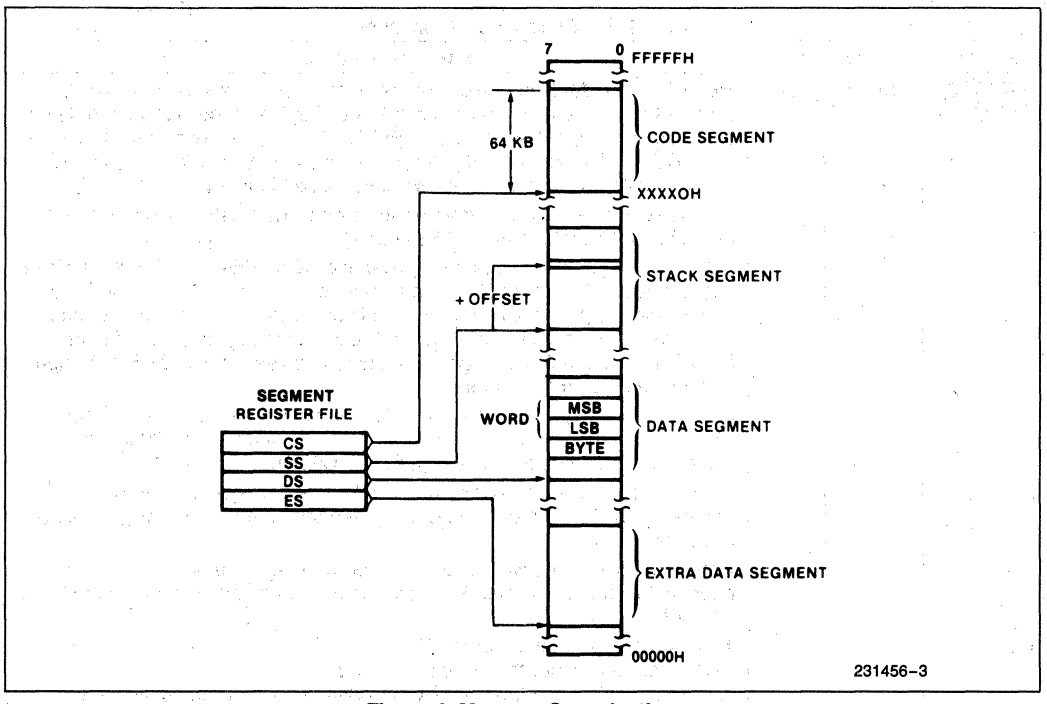


Figure 3. Memory Organization

FUNCTIONAL DESCRIPTION

Memory Organization

The processor provides a 20-bit address to memory which locates the byte being referenced. The memory is organized as a linear array of up to 1 million bytes, addressed as 00000(H) to FFFFF(H). The memory is logically divided into code, data, extra data, and stack segments of up to 64K bytes each, with each segment falling on 16-byte boundaries (See Figure 3).

All memory references are made relative to base addresses contained in high speed segment registers. The segment types were chosen based on the addressing needs of programs. The segment register to be selected is automatically chosen according to the rules of the following table. All information in one segment type share the same logical attributes (e.g. code or data). By structuring memory into relocatable areas of similar characteristics and by automatically selecting segment registers, programs are shorter, faster, and more structured.

Word (16-bit) operands can be located on even or odd address boundaries. For address and data operands, the least significant byte of the word is stored in the lower valued address location and the most significant byte in the next higher address location. The BIU will automatically execute two fetch or write cycles for 16-bit operands.

Memory Reference Used	Segment Register Used	Segment Selection Rule
Instructions	CODE (CS)	Automatic with all instruction prefetch.
Stack	STACK (SS)	All stack pushes and pops. Memory references relative to BP base register except data references.
Local Data	DATA (DS)	Data references when: relative to stack, destination of string operation, or explicity overridden.
External (Global) Data	EXTRA (ES)	Destination of string operations: Explicitly selected using a segment override.

Certain locations in memory are reserved for specific CPU operations (See Figure 4). Locations from addresses FFFF0H through FFFFFH are reserved for operations including a jump to the initial system initialization routine. Following RESET, the CPU will always begin execution at location FFFF0H where the jump must be located. Locations 00000H through 003FFH are reserved for interrupt operations. Four-byte pointers consisting of a 16-bit segment address and a 16-bit offset address direct program flow to one of the 256 possible interrupt service routines. The pointer elements are assumed to have been stored at their respective places in reserved memory prior to the occurrence of interrupts.

Minimum and Maximum Modes

The requirements for supporting minimum and maximum 8088 systems are sufficiently different that they cannot be done efficiently with 40 uniquely defined pins. Consequently, the 8088 is equipped with a strap pin (MN/$\overline{MX}$) which defines the system configuration. The definition of a certain subset of the pins changes, dependent on the condition of the strap pin. When the MN/$\overline{MX}$ pin is strapped to GND, the 8088 defines pins 24 through 31 and 34 in maximum mode. When the MN/$\overline{MX}$ pin is strapped to V_{CC}, the 8088 generates bus control signals itself on pins 24 through 31 and 34.

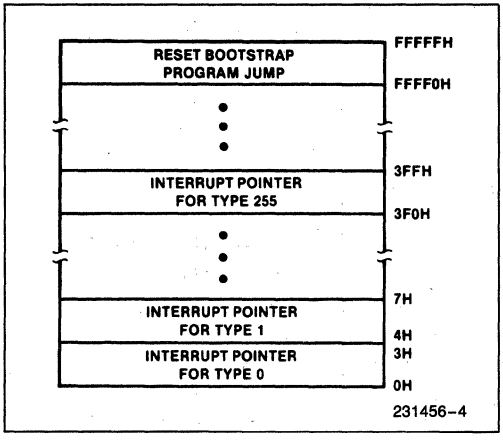


Figure 4. Reserved Memory Locations

The minimum mode 8088 can be used with either a multiplexed or demultiplexed bus. The multiplexed bus configuration is compatible with the MCS-85™ multiplexed bus peripherals. This configuration (See Figure 5) provides the user with a minimum chip count system. This architecture provides the 8088 processing power in a highly integrated form.

The demultiplexed mode requires one latch (for 64K addressability) or two latches (for a full megabyte of addressing). A third latch can be used for buffering if the address bus loading requires it. A transceiver can also be used if data bus buffering is required (See Figure 6). The 8088 provides $\overline{DEN}$ and DT/$\overline{R}$ to control the transceiver, and ALE to latch the addresses. This configuration of the minimum mode provides the standard demultiplexed bus structure with heavy bus buffering and relaxed bus timing requirements.

The maximum mode employs the 8288 bus controller (See Figure 7). The 8288 decodes status lines $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$, and provides the system with all bus control signals. Moving the bus control to the 8288 provides better source and sink current capability to the control lines, and frees the 8088 pins for extended large system features. Hardware lock, queue status, and two request/grant interfaces are provided by the 8088 in maximum mode. These features allow co-processors in local bus and remote bus configurations.

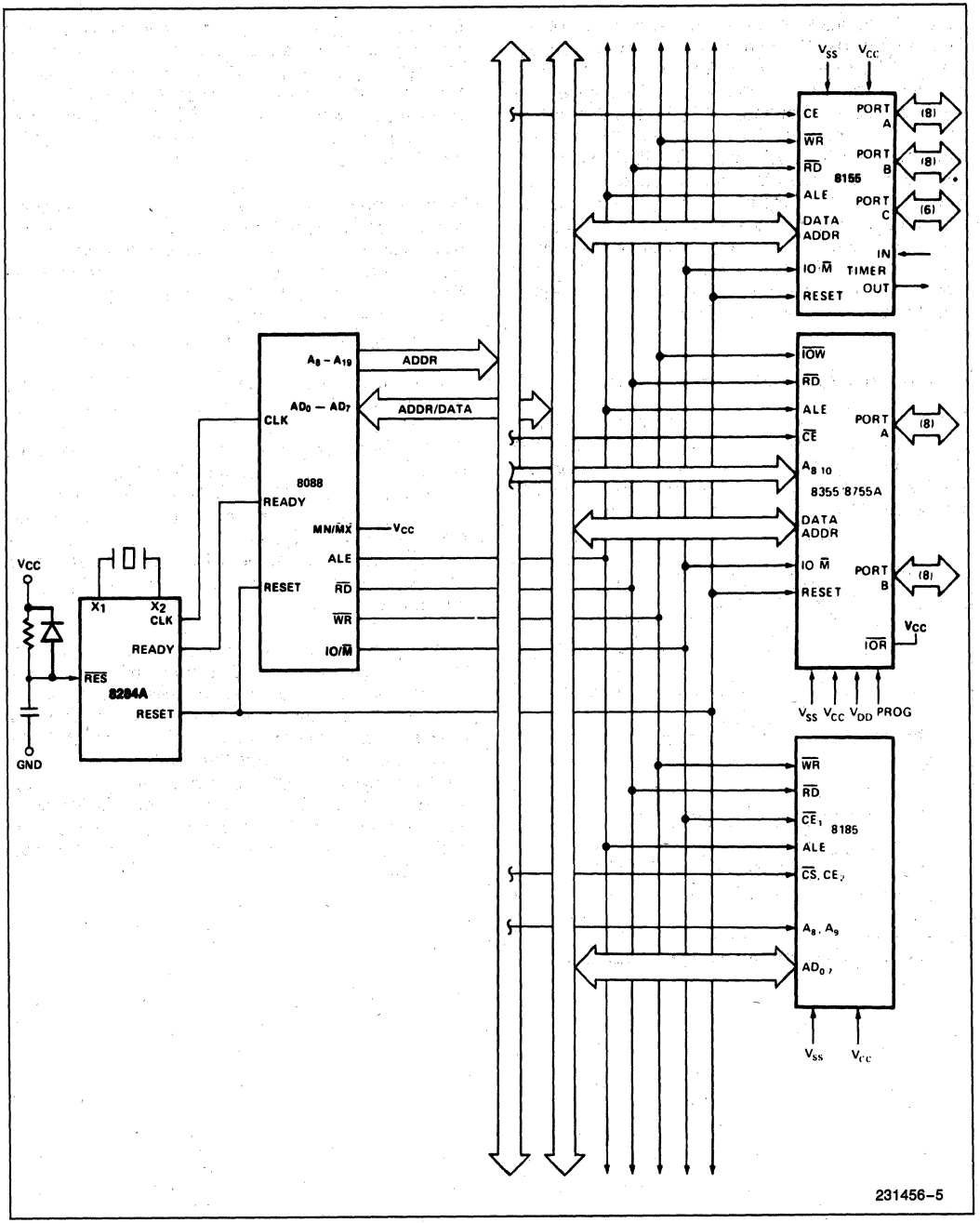


Figure 5. Multiplexed Bus Configuration

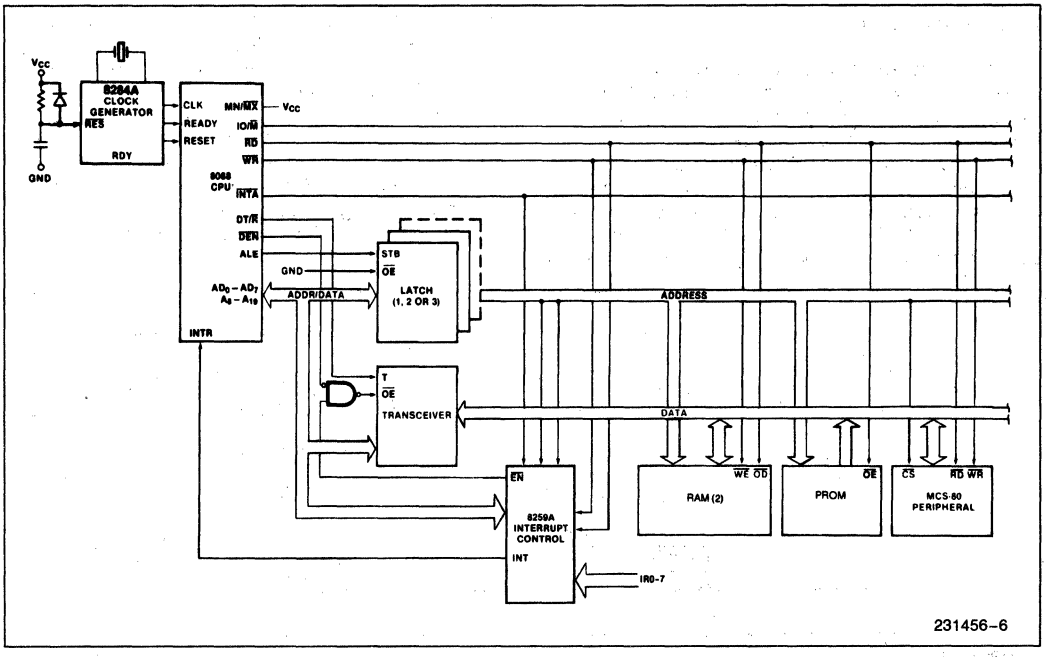


Figure 6. Demultiplexed Bus Configuration

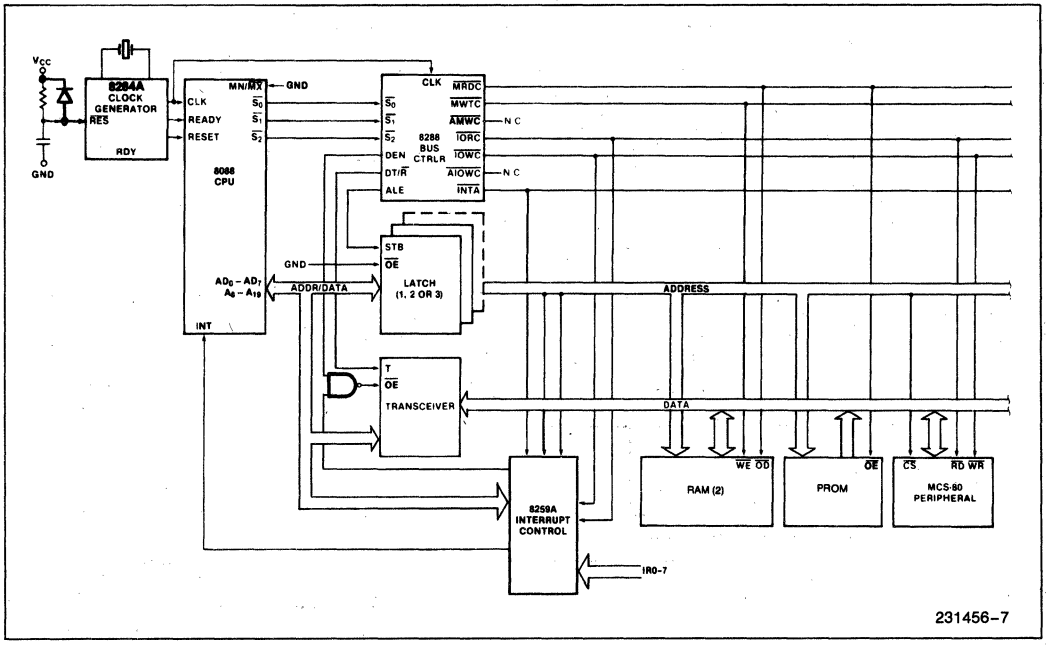


Figure 7. Fully Buffered System Using Bus Controller

Bus Operation

The 8088 address/data bus is broken into three parts—the lower eight address/data bits (AD0–AD7), the middle eight address bits (A8–A15), and the upper four address bits (A16–A19). The address/data bits and the highest four address bits are time multiplexed. This technique provides the most efficient use of pins on the processor, permitting the use of a standard 40 lead package. The middle eight address bits are not multiplexed, i.e. they remain valid throughout each bus cycle. In addition, the bus can be demultiplexed at the processor with a single address latch if a standard, non-multiplexed bus is desired for the system.

Each processor bus cycle consists of at least four CLK cycles. These are referred to as T1, T2, T3, and T4 (See Figure 8). The address is emitted from the processor during T1 and data transfer occurs on the bus during T3 and T4. T2 is used primarily for chang-

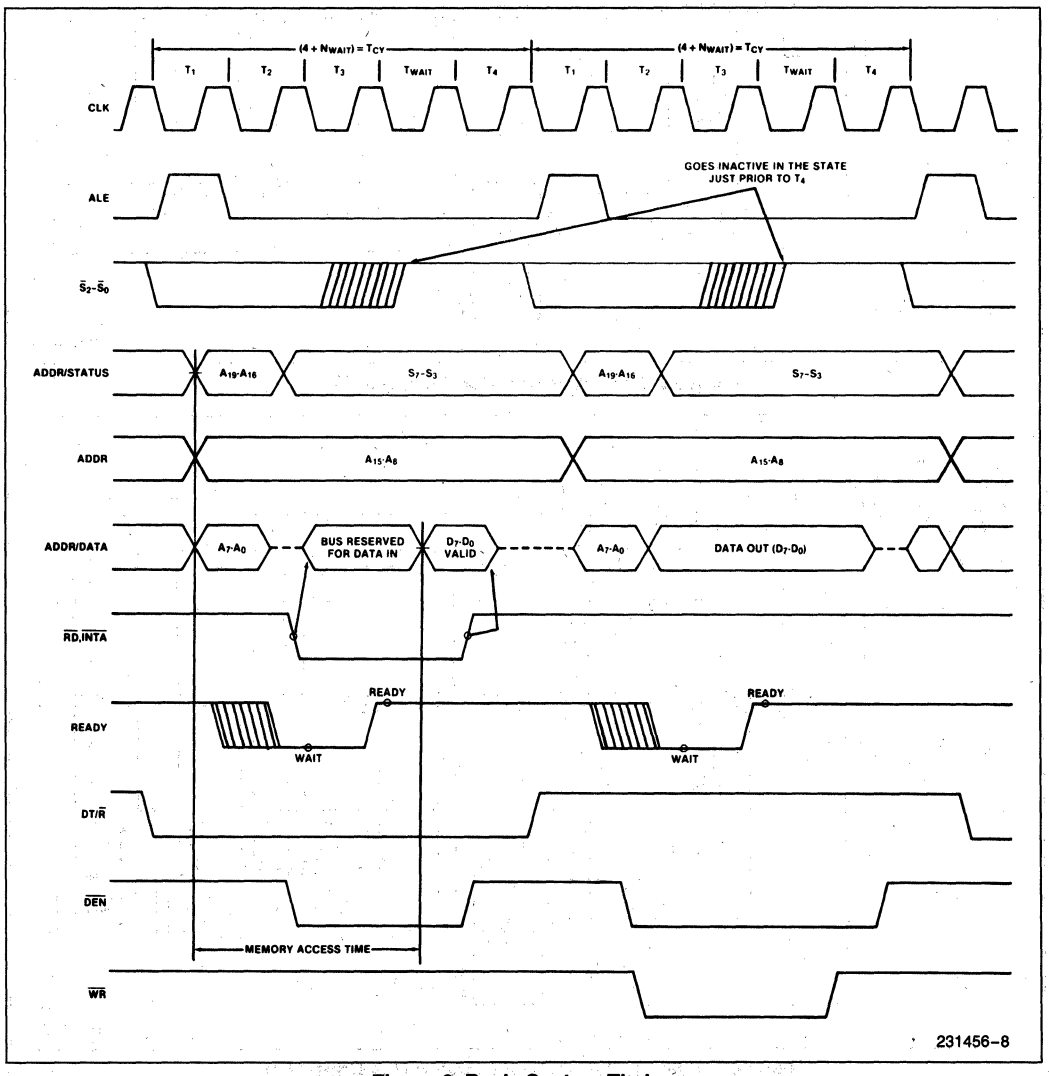


Figure 8. Basic System Timing

ing the direction of the bus during read operations. In the event that a "NOT READY" indication is given by the addressed device, "wait" states (Tw) are inserted between T3 and T4. Each inserted "wait" state is of the same duration as a CLK cycle. Periods can occur between 8088 driven bus cycles. These are referred to as "idle" states (Ti), or inactive CLK cycles. The processor uses these cycles for internal housekeeping.

During T1 of any bus cycle, the ALE (address latch enable) signal is emitted (by either the processor or the 8288 bus controller, depending on the MN/$\overline{MX}$ strap). At the trailing edge of this pulse, a valid address and certain status information for the cycle may be latched.

Status bits $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$ are used by the bus controller, in maximum mode, to identify the type of bus transaction according to the following table:

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Characteristics
0(LOW)	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1(HIGH)	0	0	Instruction Fetch
1	0	1	Read Data from Memory
1	1	0	Write Data to Memory
1	1	1	Passive (No Bus Cycle)

Status bits S3 through S6 are multiplexed with high order address bits and are therefore valid during T2 through T4. S3 and S4 indicate which segment register was used for this bus cycle in forming the address according to the following table:

S_4	S_3	Characteristics
0(LOW)	0	Alternate Data (Extra Segment)
0	1	Stack
1(HIGH)	0	Code or None
1	1	Data

S5 is a reflection of the PSW interrupt enable bit. S6 is always equal to 0.

I/O Addressing

In the 8088, I/O operations can address up to a maximum of 64K I/O registers. The I/O address appears in the same format as the memory address on bus lines A15–A0. The address lines A19–A16 are zero in I/O operations. The variable I/O instructions, which use register DX as a pointer, have full address capability, while the direct I/O instructions directly address one or two of the 256 I/O byte locations in page 0 of the I/O address space. I/O ports are addressed in the same manner as memory locations.

Designers familiar with the 8085 or upgrading an 8085 design should note that the 8085 addresses I/O with an 8-bit address on both halves of the 16-bit address bus. The 8088 uses a full 16-bit address on its lower 16 address lines.

EXTERNAL INTERFACE

Processor Reset and Initialization

Processor initialization or start up is accomplished with activation (HIGH) of the RESET pin. The 8088 RESET is required to be HIGH for greater than four clock cycles. The 8088 will terminate operations on the high-going edge of RESET and will remain dormant as long as RESET is HIGH. The low-going transition of RESET triggers an internal reset sequence for approximately 7 clock cycles. After this interval the 8088 operates normally, beginning with the instruction in absolute locations FFFF0H (See Figure 4). The RESET input is internally synchronized to the processor clock. At initialization, the HIGH to LOW transition of RESET must occur no sooner than 50 μs after power up, to allow complete initialization of the 8088.

NMI asserted prior to the 2nd clock after the end of RESET will not be honored. If NMI is asserted after that point and during the internal reset sequence, the processor may execute one instruction before responding to the interrupt. A hold request active immediately after RESET will be honored before the first instruction fetch.

All 3-state outputs float to 3-state OFF during RESET. Status is active in the idle state for the first clock after RESET becomes active and then floats to 3-state OFF. ALE and HLDA are driven low.

Interrupt Operations

Interrupt operations fall into two classes: software or hardware initiated. The software initiated interrupts and software aspects of hardware interrupts are specified in the instruction set description in the iAPX 88 book or the iAPX 86,88 User's Manual. Hardware interrupts can be classified as nonmaskable or maskable.

Interrupts result in a transfer of control to a new program location. A 256 element table containing address pointers to the interrupt service program locations resides in absolute locations 0 through 3FFH (See Figure 4), which are reserved for this purpose. Each element in the table is 4 bytes in size and corresponds to an interrupt "type." An interrupting device supplies an 8-bit type number, during the interrupt acknowledge sequence, which is used to vector through the appropriate element to the new interrupt service program location.

Non-Maskable Interrupt (NMI)

The processor provides a single non-maskable interrupt (NMI) pin which has higher priority than the maskable interrupt request (INTR) pin. A typical use would be to activate a power failure routine. The NMI is edge-triggered on a LOW to HIGH transition. The activation of this pin causes a type 2 interrupt.

NMI is required to have a duration in the HIGH state of greater than two clock cycles, but is not required to be synchronized to the clock. Any higher going transition of NMI is latched on-chip and will be serviced at the end of the current instruction or between whole moves (2 bytes in the case of word moves) of a block type instruction. Worst case response to NMI would be for multiply, divide, and variable shift instructions. There is no specification on the occurrence of the low-going edge; it may occur before, during, or after the servicing of NMI. Another high-going edge triggers another response if it occurs after the start of the NMI procedure. The signal must be free of logical spikes in general and be free of bounces on the low-going edge to avoid triggering extraneous responses.

Maskable Interrupt (INTR)

The 8088 provides a single interrupt request input (INTR) which can be masked internally by software with the resetting of the interrupt enable (IF) flag bit. The interrupt request signal is level triggered. It is internally synchronized during each clock cycle on the high-going edge of CLK. To be responded to, INTR must be present (HIGH) during the clock period preceding the end of the current instruction or the end of a whole move for a block type instruction. During interrupt response sequence, further interrupts are disabled. The enable bit is reset as part of the response to any interrupt (INTR, NMI, software interrupt, or single step), although the FLAGS register which is automatically pushed onto the stack reflects the state of the processor prior to the interrupt. Until the old FLAGS register is restored, the enable bit will be zero unless specifically set by an instruction.

During the response sequence (See Figure 9), the processor executes two successive (back to back) interrupt acknowledge cycles. The 8088 emits the LOCK signal (maximum mode only) from T2 of the first bus cycle until T2 of the second. A local bus "hold" request will not be honored until the end of the second bus cycle. In the second bus cycle, a byte is fetched from the external interrupt system (e.g., 8259A PIC) which identifies the source (type) of the interrupt. This byte is multiplied by four and used as a pointer into the interrupt vector lookup table. An INTR signal left HIGH will be continually responded to within the limitations of the enable bit and sample period. The interrupt return instruction includes a flags pop which returns the status of the original interrupt enable bit when it restores the flags.

HALT

When a software HALT instruction is executed, the processor indicates that it is entering the HALT state in one of two ways, depending upon which mode is strapped. In minimum mode, the processor issues ALE, delayed by one clock cycle, to allow the system to latch the halt status. Halt status is available on IO/$\overline{M}$, DT/$\overline{R}$, and $\overline{SSO}$. In maximum mode, the processor issues appropriate HALT status on $\overline{S2}$, $\overline{S1}$, and $\overline{S0}$, and the 8288 bus controller issues one ALE. The 8088 will not leave the HALT state when a local bus hold is entered while in HALT. In this case, the processor reissues the HALT indicator at the end of the local bus hold. An interrupt request or RESET will force the 8088 out of the HALT state.

Read/Modify/Write (Semaphore) Operations via LOCK

The LOCK status information is provided by the processor when consecutive bus cycles are required during the execution of an instruction. This allows the processor to perform read/modify/write operations on memory (via the "exchange register with memory" instruction), without another system bus master receiving intervening memory cycles. This is useful in multiprocessor system configurations to accomplish "test and set lock" operations. The $\overline{LOCK}$ signal is activated (LOW) in the clock cycle following decoding of the LOCK prefix instruction. It is deactivated at the end of the last bus cycle of the instruction following the LOCK prefix. While $\overline{LOCK}$ is active, a request on a $\overline{RQ}$/$\overline{GT}$ pin will be recorded, and then honored at the end of the LOCK.

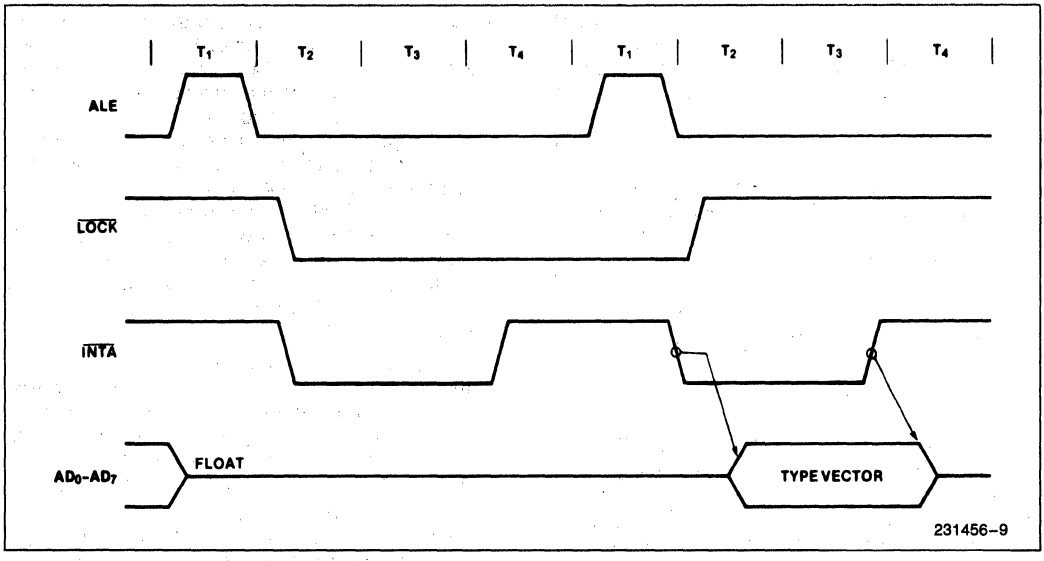


Figure 9. Interrupt Acknowledge Sequence

External Synchronization via TEST

As an alternative to interrupts, the 8088 provides a single software-testable input pin (TEST). This input is utilized by executing a WAIT instruction. The single WAIT instruction is repeatedly executed until the TEST input goes active (LOW). The execution of WAIT does not consume bus cycles once the queue is full.

If a local bus request occurs during WAIT execution, the 8088 3-states all output drivers. If interrupts are enabled, the 8088 will recognize interrupts and process them. The WAIT instruction is then refetched, and reexecuted.

Basic System Timing

In minimum mode, the MN/MX pin is strapped to V_{CC} and the processor emits bus control signals compatible with the 8085 bus structure. In maximum mode, the MN/MX pin is strapped to GND and the processor emits coded status information which the 8288 bus controller uses to generate MULTIBUS compatible bus control signals.

System Timing—Minimum System

(See Figure 8)

The read cycle begins in T1 with the assertion of the address latch enable (ALE) signal. The trailing (low going) edge of this signal is used to latch the address information, which is valid on the address/data bus (AD0–AD7) at this time, into the 8282/8283 latch. Address lines A8 through A15 do not need to be latched because they remain valid throughout the bus cycle. From T1 to T4 the IO/M signal indicates a memory or I/O operation. At T2 the address is removed from the address/data bus and the bus goes to a high impedance state. The read control signal is also asserted at T2. The read (RD) signal causes the addressed device to enable its data bus drivers to the local bus. Some time later, valid data will be available on the bus and the addressed device will drive the READY line HIGH. When the processor returns the read signal to a HIGH level, the addressed device will again 3-state its bus drivers. If a transceiver is required to buffer the 8088 local bus, signals DT/R and DEN are provided by the 8088.

A write cycle also begins with the assertion of ALE and the emission of the address. The IO/M signal is again asserted to indicate a memory or I/O write operation. In T2, immediately following the address emission, the processor emits the data to be written into the addressed location. This data remains valid until at least the middle of T4. During T2, T3, and Tw, the processor asserts the write control signal. The write (WR) signal becomes active at the beginning of T2, as opposed to the read, which is delayed somewhat into T2 to provide time for the bus to float.

The basic difference between the interrupt acknowledge cycle and a read cycle is that the interrupt acknowledge ($\overline{INTA}$) signal is asserted in place of the read ($\overline{RD}$) signal and the address bus is floated. (See Figure 9) In the second of two successive $\overline{INTA}$ cycles, a byte of information is read from the data bus, as supplied by the interrupt system logic (i.e. 8259A priority interrupt controller). This byte identifies the source (type) of the interrupt. It is multiplied by four and used as a pointer into the interrupt vector lookup table, as described earlier.

Bus Timing—Medium Complexity Systems

(See Figure 10)

For medium complexity systems, the MN/$\overline{MX}$ pin is connected to GND and the 8288 bus controller is added to the system, as well as a latch for latching the system address, and a transceiver to allow for bus loading greater than the 8088 is capable of handling. Signals ALE, $\overline{DEN}$, and DT/$\overline{R}$ are generated by the 8288 instead of the processor in this configuration, although their timing remains relatively the same. The 8088 status outputs ($\overline{S2}$, $\overline{S1}$, and $\overline{S0}$) provide type of cycle information and become 8288 inputs. This bus cycle information specifies read (code, data, or I/O), write (data or I/O), interrupt acknowledge, or software halt. The 8288 thus issues control signals specifying memory read or write, I/O read or write, or interrupt acknowledge. The 8288 provides two types of write strobes, normal and advanced, to be applied as required. The normal write strobes have data valid at the leading edge of write. The advanced write strobes have the same timing as read strobes, and hence, data is not valid at the leading edge of write. The transceiver receives the usual T and $\overline{OE}$ inputs from the 8288's DT/$\overline{R}$ and $\overline{DEN}$ outputs.

The pointer into the interrupt vector table, which is passed during the second $\overline{INTA}$ cycle, can derive from an 8259A located on either the local bus or the system bus. If the master 8289A priority interrupt controller is positioned on the local bus, a TTL gate is required to disable the transceiver when reading from the master 8259A during the interrupt acknowledge sequence and software "poll".

The 8088 Compared to the 8086

The 8088 CPU is an 8-bit processor designed around the 8086 internal structure. Most internal functions of the 8088 are identical to the equivalent 8086 functions. The 8088 handles the external bus the same way the 8086 does with the distinction of handling only 8 bits at a time. Sixteen-bit operands are fetched or written in two consecutive bus cycles. Both processors will appear identical to the software engineer, with the exception of execution time. The internal register structure is identical and all instructions have the same end result. The differences between the 8088 and 8086 are outlined below. The engineer who is unfamiliar with the 8086 is referred to the iAPX 86, 88 User's Manual, Chapters 2 and 4, for function description and instruction set information. Internally, there are three differences between the 8088 and the 8086. All changes are related to the 8-bit bus interface.

- The queue length is 4 bytes in the 8088, whereas the 8086 queue contains 6 bytes, or three words. The queue was shortened to prevent overuse of the bus by the BIU when prefetching instructions. This was required because of the additional time necessary to fetch instructions 8 bits at a time.
- To further optimize the queue, the prefetching algorithm was changed. The 8088 BIU will fetch a new instruction to load into the queue each time there is a 1 byte hole (space available) in the queue. The 8086 waits until a 2-byte space is available.
- The internal execution time of the instruction set is affected by the 8-bit interface. All 16-bit fetches and writes from/to memory take an additional four clock cycles. The CPU is also limited by the speed of instruction fetches. This latter problem only occurs when a series of simple operations occur. When the more sophisticated instructions of the 8088 are being used, the queue has time to fill and the execution proceeds as fast as the execution unit will allow.

The 8088 and 8086 are completely software compatible by virtue of their identical execution units. Software that is system dependent may not be completely transferable, but software that is not system dependent will operate equally as well on an 8088 and an 8086.

The hardware interface of the 8088 contains the major differences between the two CPUs. The pin assignments are nearly identical, however, with the following functional changes:

- A8–A15—These pins are only address outputs on the 8088. These address lines are latched internally and remain valid throughout a bus cycle in a manner similar to the 8085 upper address lines.
- $\overline{BHE}$ has no meaning on the 8088 and has been eliminated.

- $\overline{SSO}$ provides the $\overline{SO}$ status information in the minimum mode. This output occurs on pin 34 in minimum mode only. DT/$\overline{R}$, IO/$\overline{M}$, and $\overline{SSO}$ provide the complete bus status in minimum mode.
- IO/$\overline{M}$ has been inverted to be compatible with the MCS-85 bus structure.
- ALE is delayed by one clock cycle in the minimum mode when entering HALT, to allow the status to be latched with ALE.

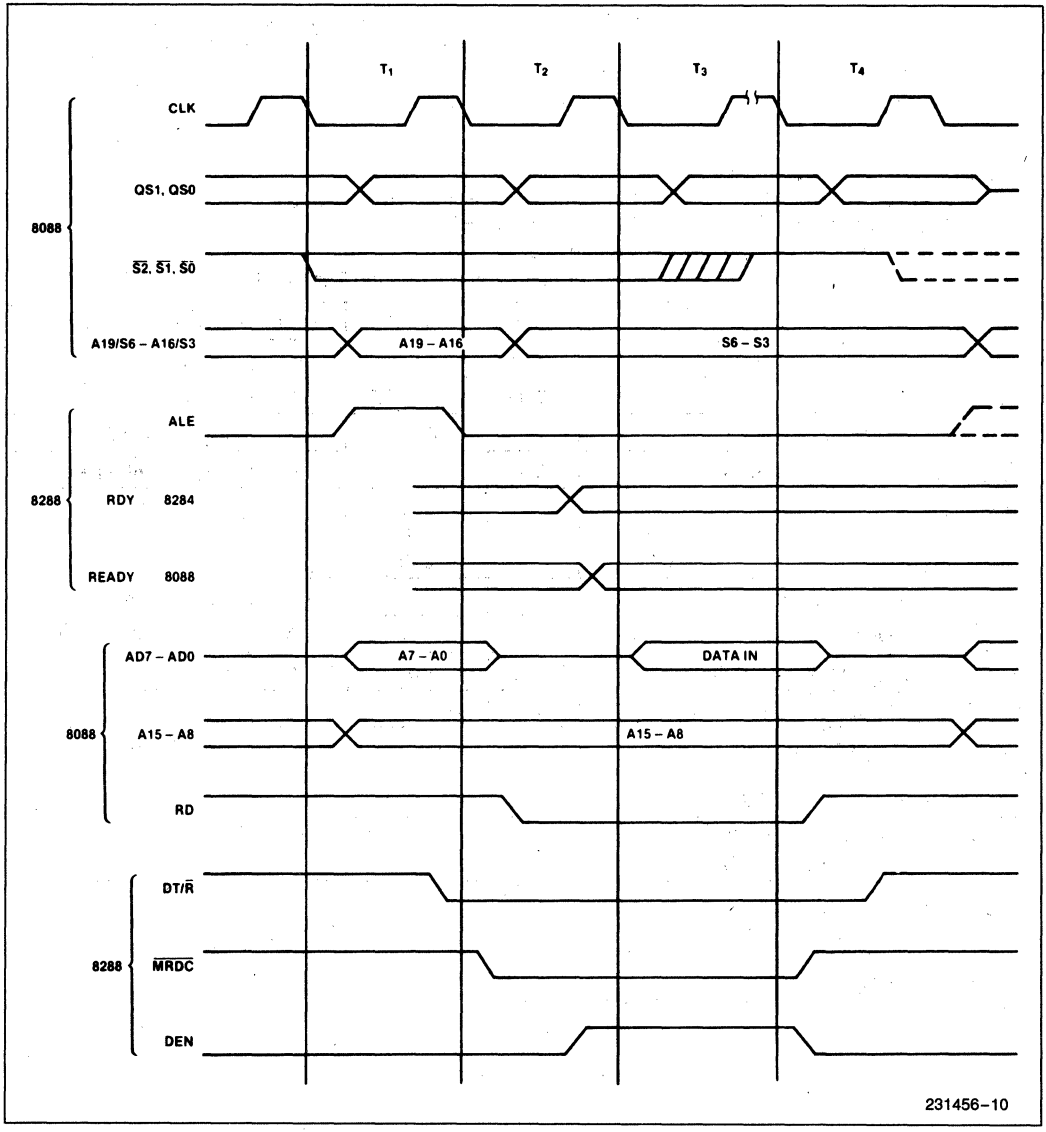


Figure 10. Medium Complexity System Timing

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ....0°C to +70°C
Case Temperature (Plastic) .....0°C to +95°C
Case Temperature (CERDIP) .....0°C to +75°C
Storage Temperature .....−65°C to +150°C
Voltage on Any Pin with
Respect to Ground .....−1.0 to +7V
Power Dissipation .....2.5 Watt

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

(T_A = 0°C to 70°C, T_{CASE} (Plastic) = 0°C to 95°C, T_{CASE} (CERDIP) = 0°C to 75°C,
T_A = 0°C to 55°C and T_{CASE} = 0°C to 75°C for P8088-2 only
T_A is guaranteed as long as T_{CASE} is not exceeded)

(V_{CC} = 5V ±10% for 8088, V_{CC} = 5V ±5% for 8088-2 and Extended Temperature EXPRESS)

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	+0.8	V	(Note 1)
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.5	V	(Notes 1, 2)
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −400 μA
I_{CC}	Power Supply Current: 8088 8088-2 P8088		340 350 250	mA	T_A = 25°C
I_{LI}	Input Leakage Current		±10	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{LO}	Output and I/O Leakage Current		±10	μA	0.45V ≤ V_{OUT} ≤ V_{CC}
V_{CL}	Clock Input Low Voltage	−0.5	+0.6	V	
V_{CH}	Clock Input High Voltage	3.9	V_{CC} + 1.0	V	
C_{IN}	Capacitance If Input Buffer (All Input Except AD_0–AD_7, RQ/GT)		15	pF	fc = 1 MHz
C_{IO}	Capacitance of I/O Buffer AD_0–AD_7, RQ/GT)		15	pF	fc = 1 MHz

NOTES:
1. V_{IL} tested with MN/$\overline{MX}$ Pin = 0V
   V_{IH} tested with MN/$\overline{MX}$ Pin = 5V
   MN/$\overline{MX}$ Pin is a strap Pin
2. Not applicable to $\overline{RQ}$/$\overline{GT0}$ and $\overline{RQ}$/$\overline{GT1}$ Pins (Pin 30 and 31)

A.C. CHARACTERISTICS

(T_A = 0°C to 70°C, T_{CASE} (Plastic) = 0°C to 95°C, T_{CASE} (CERDIP) = 0°C to 75°C,
T_A = 0°C to 55°C and T_{CASE} = 0°C to 80°C for P8088-2 only
T_A is guaranteed as long as T_{CASE} is not exceeded)

(V_{CC} = 5V ±10% for 8088, V_{CC} = 5V ±5% for 8088-2 and Extended Temperature EXPRESS)

MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

Symbol	Parameter	8088		8088-2		Units	Test Conditions
		Min	Max	Min	Max		
TCLCL	CLK Cycle Period	200	500	125	500	ns	
TCLCH	CLK Low Time	118		68		ns	
TCHCL	CLK High Time	69		44		ns	
TCH1CH2	CLK Rise Time		10		10	ns	From 1.0V to 3.5V
TCL2CL2	CLK Fall Time		10		10	ns	From 3.5V to 1.0V
TDVCL	Data in Setup Time	30		20		ns	
TCLDX	Data in Hold Time	10		10		ns	
TR1VCL	RDY Setup Time into 8284 (Notes 1, 2)	35		35		ns	
TCLR1X	RDY Hold Time into 8284 (Notes 1, 2)	0		0		ns	
TRYHCH	READY Setup Time into 8088	118		68		ns	
TCHRYX	READY Hold Time into 8088	30		20		ns	
TRYLCL	READY Inactive to CLK (Note 3)	−8		−8		ns	
THVCH	HOLD Setup Time	35		20		ns	
TINVCH	INTR, NMI, $\overline{TEST}$ Setup Time (Note 2)	30		15		ns	
TILIH	Input Rise Time (Except CLK)		20		20	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK)		12		12	ns	From 2.0V to 0.8V

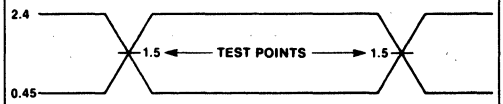
A.C. CHARACTERISTICS (Continued)

TIMING RESPONSES

Symbol	Parameter	8088		8088-2		Units	Test Conditions
		Min	Max	Min	Max		
TCLAV	Address Valid Delay	10	110	10	60	ns	
TCLAX	Address Hold Time	10		10		ns	
TCLAZ	Address Float Delay	TCLAX	80	TCLAX	50	ns	
TLHLL	ALE Width	TCLCH−20		TCLCH−10		ns	
TCLLH	ALE Active Delay		80		50	ns	
TCHLL	ALE Inactive Delay		85		55	ns	
TLLAX	Address Hold Time to ALE Inactive	TCHCL−10		TCHCL−10		ns	
TCLDV	Data Valid Delay	10	110	10	60	ns	
TCHDX	Data Hold Time	10		10		ns	
TWHDX	Data Hold Time after $\overline{WR}$	TCLCH−30		TCLCH−30		ns	
TCVCTV	Control Active Delay 1	10	110	10	70	ns	
TCHCTV	Control Active Delay 2	10	110	10	60	ns	
TCVCTX	Control Inactive Delay	10	110	10	70	ns	
TAZRL	Address Float to READ Active	0		0		ns	
TCLRL	$\overline{RD}$ Active Delay	10	165	10	100	ns	
TCLRH	$\overline{RD}$ Inactive Delay	10	150	10	80	ns	
TRHAV	$\overline{RD}$ Inactive to Next Address Active	TCLCL−45		TCLCL−40		ns	
TCLHAV	HLDA Valid Delay	10	160	10	100	ns	
TRLRH	$\overline{RD}$ Width	2TCLCL−75		2TCLCL−50		ns	
TWLWH	$\overline{WR}$ Width	2TCLCL−60		2TCLCL−40		ns	
TAVAL	Address Valid to ALE Low	TCLCH−60		TCLCH−40		ns	
TOLOH	Output Rise Time		20		20	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time		12		12	ns	From 2.0V to 0.8V

NOTES:
1. Signal at 8284A shown for reference only. See 8284A data sheet for the most recent specifications.
2. Set up requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T2 state (8 ns into T3 state).

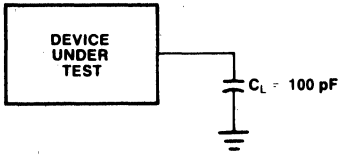
A.C. TESTING INPUT, OUTPUT WAVEFORM



231456-11

A.C. Testing; Inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0". Timing measurements are made at 2.0V for a logic "1" and 0.8V for a logic "0".

A.C. TESTING LOAD CIRCUIT

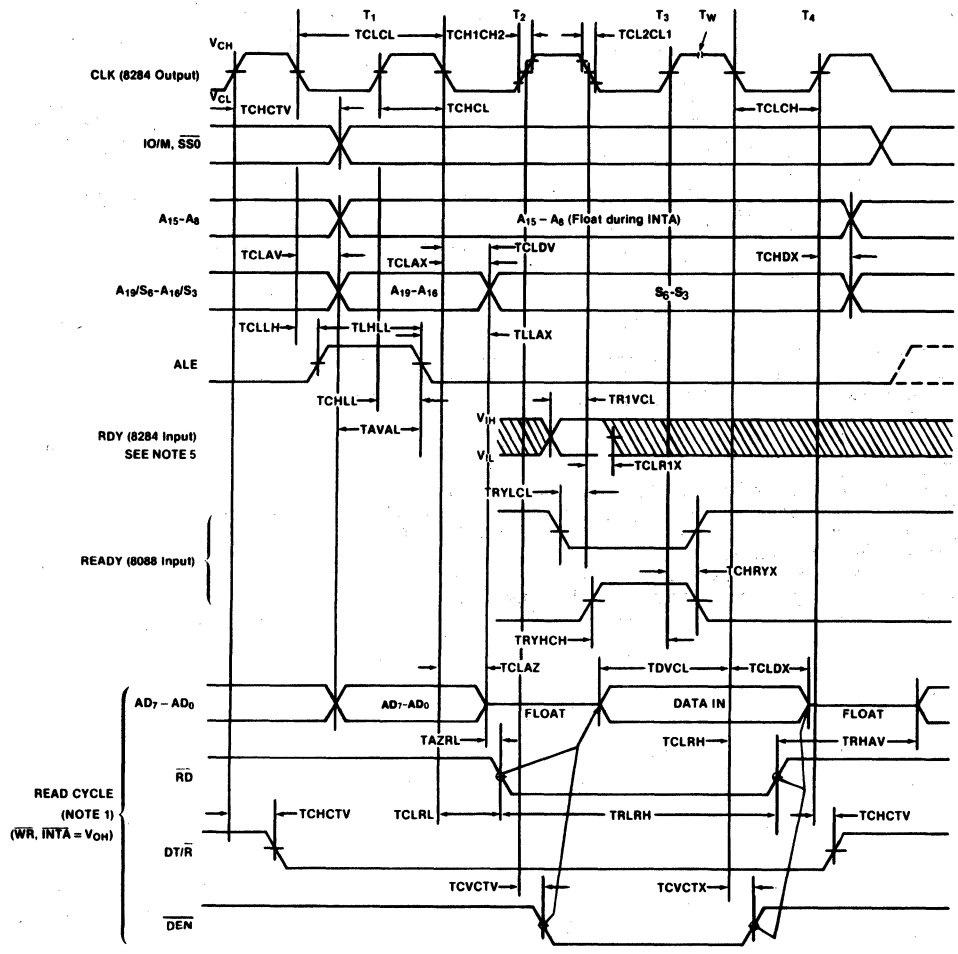


231456-12

C_L Includes Jig Capacitance

WAVEFORMS

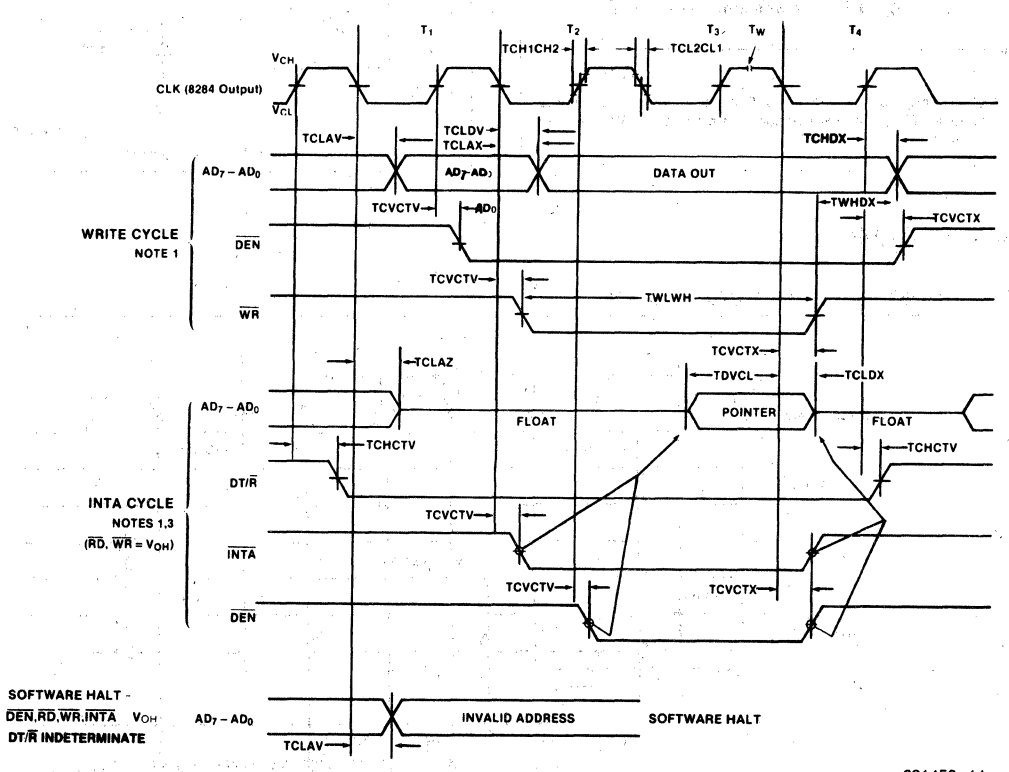
BUS TIMING—MINIMUM MODE SYSTEM



231456-13

WAVEFORMS (Continued)

BUS TIMING—MINIMUM MODE SYSTEM (Continued)



231456-14

NOTES:
1. All signals switch between V_{OH} and V_{OL} unless otherwise specified.
2. RDY is sampled near the end of T_2, T_3, T_w to determine if T_w machines states are to be inserted.
3. Two INTA cycles run back-to-back. The 8088 local ADDR/DATA bus is floating during both INTA cycles. Control signals are shown for the second INTA cycle.
4. Signals at 8284 are shown for reference only.
5. All timing measurements are made at 1.5V unless otherwise noted.

A.C. CHARACTERISTICS

MAX MODE SYSTEM (USING 8288 BUS CONTROLLER)

TIMING REQUIREMENTS

Symbol	Parameter	8088		8088-2		Units	Test Conditions
		Min	Max	Min	Max		
TCLCL	CLK Cycle Period	200	500	125	500	ns	
TCLCH	CLK Low Time	118		68		ns	
TCHCL	CLK High Time	69		44		ns	
TCH1CH2	CLK Rise Time		10		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10		10	ns	From 3.5V to 1.0V
TDVCL	Data in Setup Time	30		20		ns	
TCLDX	Data in Hold Time	10		10		ns	
TR1VCL	RDY Setup Time into 8284 (Notes 1, 2)	35		35		ns	
TCLR1X	RDY Hold Time into 8284 (Notes 1, 2)	0		0		ns	
TRYHCH	READY Setup Time into 8088	118		68		ns	
TCHRYX	READY Hold Time into 8088	30		20		ns	
TRYLCL	READY Inactive to CLK (Note 4)	−8		−8		ns	
TINVCH	Setup Time for Recognition (INTR, NMI, TEST) (Note 2)	30		15		ns	
TGVCH	RQ/GT Setup Time	30		15		ns	
TCHGX	RQ Hold Time into 8088	40		30		ns	
TILIH	Input Rise Time (Except CLK)		20		20	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK)		12		12	ns	From 2.0V to 0.8V

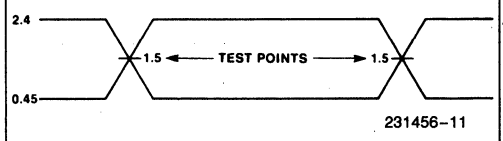
A.C. CHARACTERISTICS (Continued)

TIMING RESPONSES

Symbol	Parameter	8088		8088-2		Units	Test Conditions
		Min	Max	Min	Max		
TCLML	Command Active Delay (Note 1)	10	35	10	35	ns	
TCLMH	Command Inactive Delay (Note 1)	10	35	10	35	ns	
TRYHSH	READY Active to Status Passive (Note 3)		110		65	ns	
TCHSV	Status Active Delay	10	110	10	60	ns	
TCLSH	Status Inactive Delay	10	130	10	70	ns	
TCLAV	Address Valid Delay	10	110	10	60	ns	
TCLAX	Address Hold Time	10		10		ns	
TCLAZ	Address Float Delay	TCLAX	80	TCLAX	50	ns	
TSVLH	Status Valid to ALE High (Note 1)		15		15	ns	
TSVMCH	Status Valid to MCE High (Note 1)		15		15	ns	
TCLLH	CLK Low to ALE Valid (Note 1)		15		15	ns	
TCLMCH	CLK Low to MCE (Note 1)		15		15	ns	
TCHLL	ALE Inactive Delay (Note 1)		15		15	ns	
TCLMCL	MCE Inactive Delay (Note 1)		15		15	ns	
TCLDV	Data Valid Delay	10	110	10	60	ns	
TCHDX	Data Hold Time	10		10		ns	
TCVNV	Control Active Delay (Note 1)	5	45	5	45	ns	C_L = 20–100 pF for All 8088 Outputs in Addition to Internal Loads
TCVNX	Control Inactive Delay (Note 1)	10	45	10	45	ns	
TAZRL	Address Float to Read Active	0		0		ns	
TCLRL	$\overline{RD}$ Active Delay	10	165	10	100	ns	
TCLRH	$\overline{RD}$ Inactive Delay	10	150	10	80	ns	
TRHAV	$\overline{RD}$ Inactive to Next Address Active	TCLCL−45		TCLCL−40		ns	
TCHDTL	Direction Control Active Delay (Note 1)		50		50	ns	
TCHDTH	Direction Control Inactive Delay (Note 1)		30		30	ns	
TCLGL	GT Active Delay		85		50	ns	
TCLGH	GT Inactive Delay		85		50	ns	
TRLRH	$\overline{RD}$ Width	2TCLCL−75		2TCLCL−50		ns	
TOLOH	Output Rise Time		20		20	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time		12		12	ns	From 2.0V to 0.8V

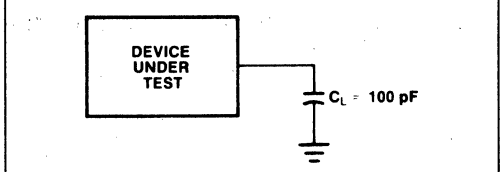
NOTES:
1. Signal at 8284 or 8288 shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T3 and wait states.
4. Applies only to T2 state (8 ns into T3 state).

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. Testing; Inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0". Timing measurements are made at 2.0V for a logic "1" and 0.8V for a logic "0".

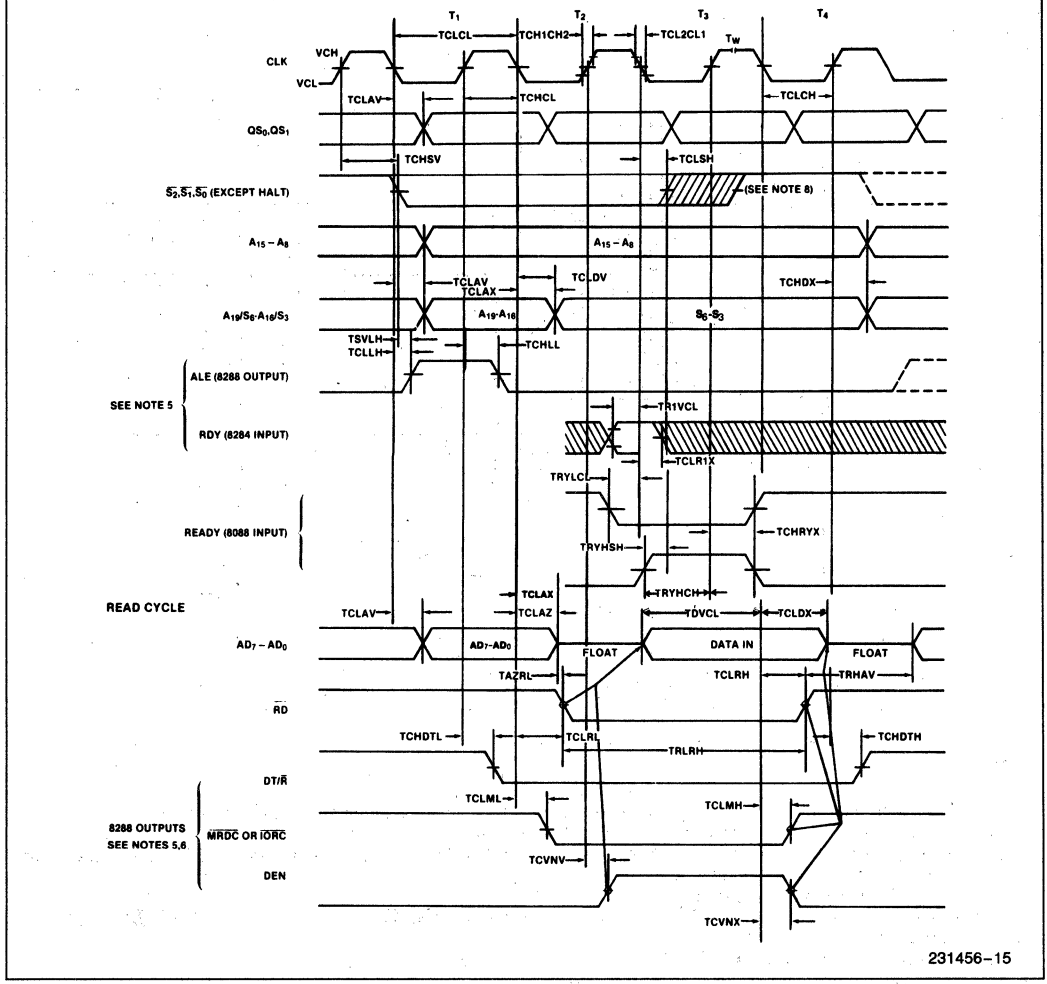
A.C. TESTING LOAD CIRCUIT



C_L Includes Jig Capacitance

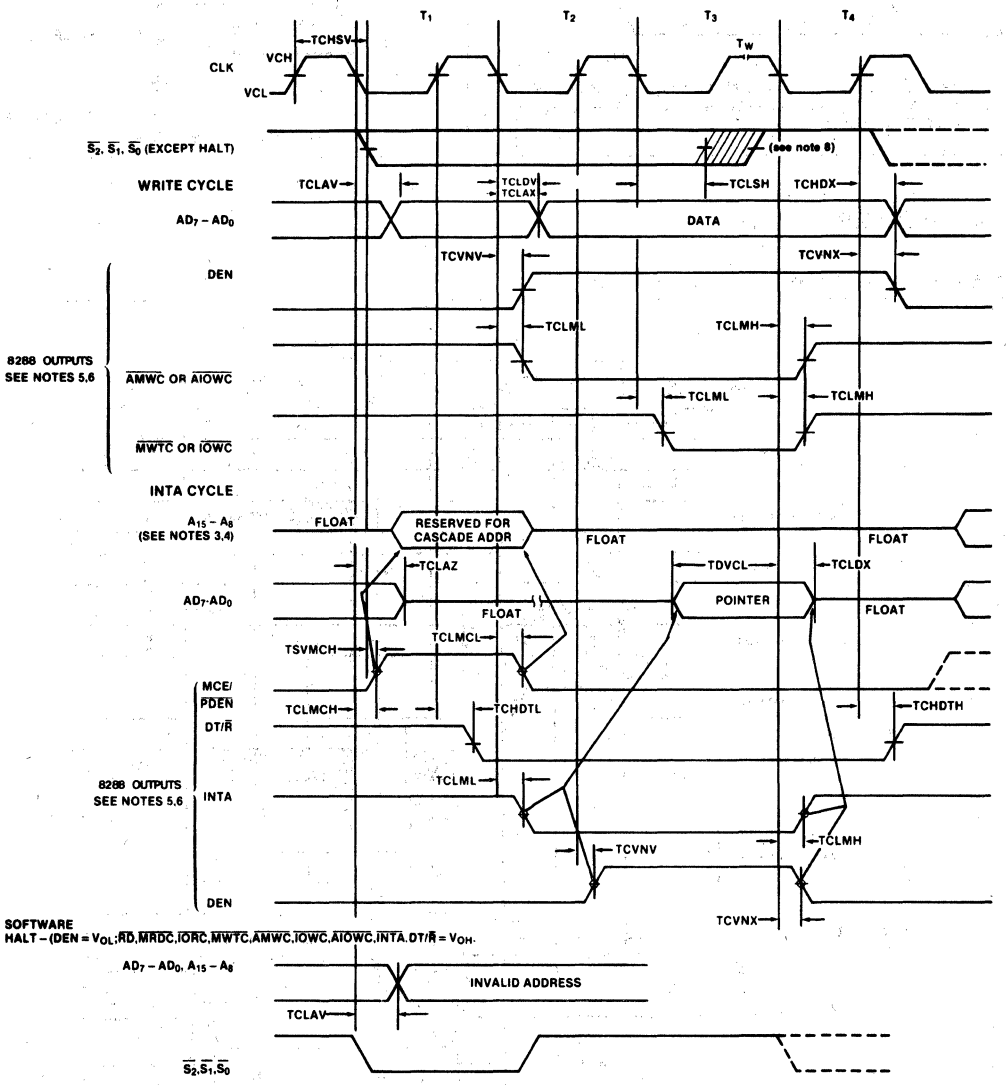
WAVEFORMS (Continued)

BUS TIMING—MAXIMUM MODE SYSTEM



WAVEFORMS (Continued)

BUS TIMING—MAXIMUM MODE SYSTEM (USING 8288)



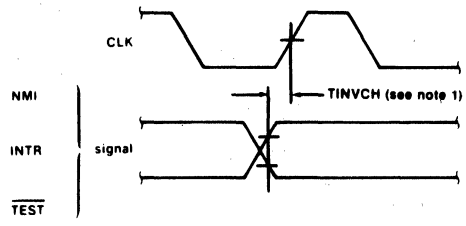
231456-16

NOTES:

1. All signals switch between V_{OH} and V_{OL} unless otherwise specified.
2. RDY is sampled near the end of T_2, T_3, T_W to determine if T_W machines states are to be inserted.
3. Cascade address is valid between first and second INTA cycles.
4. Two INTA cycles run back-to-back. The 8088 local ADDR/DATA bus is floating during both INTA cycles. Control for pointer address is shown for second INTA cycle.
5. Signals at 8284 or 8288 are shown for reference only.
6. The issuance of the 8288 command and control signals ($\overline{MRDC}$, $\overline{MWTC}$, $\overline{AMWC}$, $\overline{IORC}$, $\overline{IOWC}$, $\overline{AIOWC}$, $\overline{INTA}$ and DEN) lags the active high 8288 CEN.
7. All timing measurements are made at 1.5V unless otherwise noted.
8. Status inactive in state just prior to T_4.

WAVEFORMS (Continued)

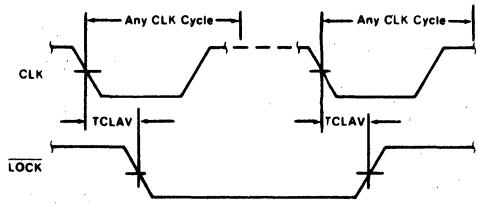
ASYNCHRONOUS SIGNAL RECOGNITION



231456-17

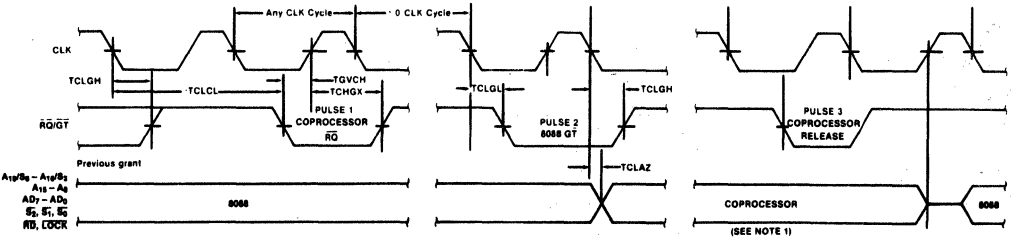
NOTE:
1. Setup requirements for asynchronous signals only to guarantee recognition at next CLK.

BUS LOCK SIGNAL TIMING (MAXIMUM MODE ONLY)



231456-18

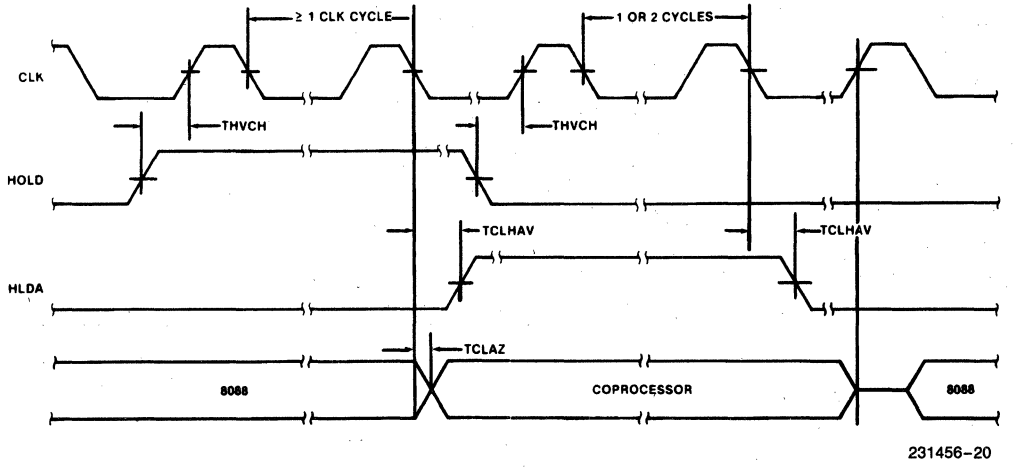
REQUEST/GRANT SEQUENCE TIMING (MAXIMUM MODE ONLY)



231456-19

NOTE:
1. The coprocessor may not drive the busses outside the region shown without risking contention.

HOLD/HOLD ACKNOWLEDGE TIMING (MINIMUM MODE ONLY)



231456-20

8086/8088 Instruction Set Summary

Mnemonic and Description	Instruction Code			
DATA TRANSFER				
MOV = Move:	76543210	76543210	76543210	76543210
Register/Memory to/from Register	100010dw	mod reg r/m		
Immediate to Register/Memory	1100011w	mod 000 r/m	data	data if w = 1
Immediate to Register	1011 w reg	data	data if w = 1	
Memory to Accumulator	1010000w	addr-low	addr-high	
Accumulator to Memory	1010001w	addr-low	addr-high	
Register/Memory to Segment Register	10001110	mod 0 reg r/m		
Segment Register to Register/Memory	10001100	mod 0 reg r/m		
PUSH = Push:				
Register/Memory	11111111	mod 1 1 0 r/m		
Register	01010 reg			
Segment Register	000 reg 110			
POP = Pop:				
Register/Memory	10001111	mod 000 r/m		
Register	01011 reg			
Segment Register	000 reg 111			
XCHG = Exchange:				
Register/Memory with Register	1000011w	mod reg r/m		
Register with Accumulator	10010 reg			
IN = Input from:				
Fixed Port	1110010w	port		
Variable Port	1110110w			
OUT = Output to:				
Fixed Port	1110011w	port		
Variable Port	1110111w			
XLAT = Translate Byte to AL	11010111			
LEA = Load EA to Register	10001101	mod reg r/m		
LDS = Load Pointer to DS	11000101	mod reg r/m		
LES = Load Pointer to ES	11000100	mod reg r/m		
LAHF = Load AH with Flags	10011111			
SAHF = Store AH into Flags	10011110			
PUSHF = Push Flags	10011100			
POPF = Pop Flags	10011101			

8086/8088 Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
ARITHMETIC	76543210	76543210	76543210	76543210
ADD = Add:				
Reg./Memory with Register to Either	000000dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod 000 r/m	data	data if s:w = 01
Immediate to Accumulator	0000010w	data	data if w = 1	
ADC = Add with Carry:				
Reg./Memory with Register to Either	000100dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod 010 r/m	data	data if s:w = 01
Immediate to Accumulator	0001010w	data	data if w = 1	
INC = Increment:				
Register/Memory	1111111w	mod 000 r/m		
Register	01000reg			
AAA = ASCII Adjust for Add	00110111			
BAA = Decimal Adjust for Add	00100111			
SUB = Subtract:				
Reg./Memory and Register to Either	001010dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod 101 r/m	data	data if s:w = 01
Immediate from Accumulator	0010110w	data	data if w = 1	
SSB = Subtract with Borrow				
Reg./Memory and Register to Either	000110dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod 011 r/m	data	data if s:w = 01
Immediate from Accumulator	0001111w	data	data if w = 1	
DEC = Decrement:				
Register/memory	1111111w	mod 001 r/m		
Register	01001 reg			
NEG = Change sign	1111011w	mod 011 r/m		
CMP = Compare:				
Register/Memory and Register	001110dw	mod reg r/m		
Immediate with Register/Memory	100000sw	mod 111 r/m	data	data if s:w = 01
Immediate with Accumulator	0011110w	data	data if w = 1	
AAS = ASCII Adjust for Subtract	00111111			
DAS = Decimal Adjust for Subtract	00101111			
MUL = Multiply (Unsigned)	1111011w	mod 100 r/m		
IMUL = Integer Multiply (Signed)	1111011w	mod 101 r/m		
AAM = ASCII Adjust for Multiply	11010100	00001010		
DIV = Divide (Unsigned)	1111011w	mod 110 r/m		
IDIV = Integer Divide (Signed)	1111011w	mod 111 r/m		
AAD = ASCII Adjust for Divide	11010101	00001010		
CBW = Convert Byte to Word	10011000			
CWD = Convert Word to Double Word	10011001			

8086/8088 Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
LOGIC	76543210	76543210	76543210	76543210
NOT = Invert	1111011w	mod 010 r/m		
SHL/SAL = Shift Logical/Arithmetic Left	110100vw	mod 100 r/m		
SHR = Shift Logical Right	110100vw	mod 101 r/m		
SAR = Shift Arithmetic Right	110100vw	mod 111 r/m		
ROL = Rotate Left	110100vw	mod 000 r/m		
ROR = Rotate Right	110100vw	mod 001 r/m		
RCL = Rotate Through Carry Flag Left	110100vw	mod 010 r/m		
RCR = Rotate Through Carry Right	110100vw	mod 011 r/m		
AND = **And:**				
Reg./Memory and Register to Either	001000dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 100 r/m	data	data if w = 1
Immediate to Accumulator	0010010w	data	data if w = 1	
TEST = **And Function to Flags. No Result:**				
Register/Memory and Register	1000010w	mod reg r/m		
Immediate Data and Register/Memory	1111011w	mod 000 r/m	data	data if w = 1
Immediate Data and Accumulator	1010100w	data	data if w = 1	
OR = **Or:**				
Reg./Memory and Register to Either	000010dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 001 r/m	data	data if w = 1
Immediate to Accumulator	0000110w	data	data if w = 1	
XOR = **Exclusive or:**				
Reg./Memory and Register to Either	001100dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 110 r/m	data	data if w = 1
Immediate to Accumulator	0011010w	data	data if w = 1	
STRING MANIPULATION				
REP = Repeat	1111001z			
MOVS = Move Byte/Word	1010010w			
CMPS = Compare Byte/Word	1010011w			
SCAS = Scan Byte/Word	1010111w			
LODS = Load Byte/Wd to AL/AX	1010110w			
STOS = Stor Byte/Wd from AL/A	1010101w			
CONTROL TRANSFER				
CALL = **Call:**				
Direct Within Segment	11101000	disp-low	disp-high	
Indirect Within Segment	11111111	mod 010 r/m		
Direct Intersegment	10011010	offset-low	offset-high	
		seg-low	seg-high	
Indirect Intersegment	11111111	mod 011 r/m		

8086/8088 Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code		
JMP = Unconditional Jump:	76543210	76543210	76543210
Direct Within Segment	11101001	disp-low	disp-high
Direct Within Segment-Short	11101011	disp	
Indirect Within Segment	11111111	mod 100 r/m	
Direct Intersegment	11101010	offset-low	offset-high
		seg-low	seg-high
Indirect Intersegment	11111111	mod 101 r/m	
RET = Return from CALL:			
Within Segment	11000011		
Within Seg Adding Immed to SP	11000010	data-low	data-high
Intersegment	11001011		
Intersegment Adding Immediate to SP	11001010	data-low	data-high
JE/JZ = Jump on Equal/Zero	01110100	disp	
JL/JNGE = Jump on Less/Not Greater or Equal	01111100	disp	
JLE/JNG = Jump on Less or Equal/ Not Greater	01111110	disp	
JB/JNAE = Jump on Below/Not Above or Equal	01110010	disp	
JBE/JNA = Jump on Below or Equal/ Not Above	01110110	disp	
JP/JPE = Jump on Parity/Parity Even	01111010	disp	
JO = Jump on Overflow	01110000	disp	
JS = Jump on Sign	01111000	disp	
JNE/JNZ = Jump on Not Equal/Not Zero	01110101	disp	
JNL/JGE = Jump on Not Less/Greater or Equal	01111101	disp	
JNLE/JG = Jump on Not Less or Equal/ Greater	01111111	disp	
JNB/JAE = Jump on Not Below/Above or Equal	01110011	disp	
JNBE/JA = Jump on Not Below or Equal/Above	01110111	disp	
JNP/JPO = Jump on Not Par/Par Odd	01111011	disp	
JNO = Jump on Not Overflow	01110001	disp	
JNS = Jump on Not Sign	01111001	disp	
LOOP = Loop CX Times	11100010	disp	
LOOPZ/LOOPE = Loop While Zero/Equal	11100001	disp	
LOOPNZ/LOOPNE = Loop While Not Zero/Equal	11100000	disp	
JCXZ = Jump on CX Zero	11100011	disp	
INT = Interrupt			
Type Specified	11001101	type	
Type 3	11001100		
INTO = Interrupt on Overflow	11001110		
IRET = Interrupt Return	11001111		

8086/8088 Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code	
	76543210	76543210
PROCESSOR CONTROL		
CLC = Clear Carry	11111000	
CMC = Complement Carry	11110101	
STC = Set Carry	11111001	
CLD = Clear Direction	11111100	
STD = Set Direction	11111101	
CLI = Clear Interrupt	11111010	
STI = Set Interrupt	11111011	
HLT = Halt	11110100	
WAIT = Wait	10011011	
ESC = Escape (to External Device)	11011xxx	mod x x x r/m
LOCK = Bus Lock Prefix	11110000	

NOTES:
AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value
Greater = more positive:
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
if w = 1 then word instruction; if w = 0 then byte instruction
if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16 bits, disp-high is absent
if mod = 10 then DISP = disp-high; disp-low
if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)
*except if mod = 00 and r/m = then EA = disp-high: disp-low.
if s:w = 01 then 16 bits of immediate data form the operand
if s:w = 11 then an immediate data byte is sign extended to form the 16-bit operand
if v = 0 then "count" = 1; if v = 1 then "count" in (CL) register
x = don't care
z is used for string primitives for comparison with ZF FLAG
SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:
FLAGS =
X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

DATA SHEET REVISION REVIEW

The following list represents key differences between this and the -003 data sheet. Please review this summary carefully.

1. In the Pin Description Table (Table 1), the description of the HLDA signal being issued has been corrected. HLDA will be issued in the middle of either the T4 or Ti state.

80C88A
8-BIT CHMOS MICROPROCESSOR

- **Pin-for-Pin and Functionally Compatible to Industry Standard HMOS 8088**
- **Direct Software Compatibility with 80C86, 8086, 8088**
- **Fully Static Design with Frequency Range from D.C. to:**
  — 8 MHz for 80C88A-2
- **Low Power Operation**
  — Operating I_{CC} = 10 mA/MHz
  — Standby I_{CCS} = 500 μA max
- **Bus-Hold Circuitry Eliminates Pull-Up Resistors**
- **Direct Addressing Capability of 1 MByte of Memory**
- **Architecture Designed for Powerful Assembly Language and Efficient High Level Languages**
- **24 Operand Addressing Modes**
- **Byte, Word and Block Operations**
- **8 and 16-Bit Signed and Unsigned Arithmetic**
  — Binary or Decimal
  — Multiply and Divide
- **Available in 40-Lead Plastic DIP**
  (See Packaging Spec., Order #231369)

The Intel 80C88A is a high performance, CHMOS version of the industry standard HMOS 8088 8-bit CPU. The processor has attributes of both 8 and 16-bit microprocessors. The 80C88A, available in 8 MHz clock rate, offers two modes of operation: MINimum for small systems and MAXimum for larger applications such as multi-processing. It is available in 40-pin DIP.

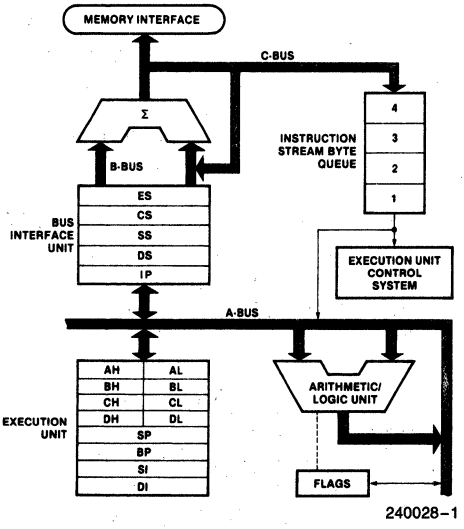


Figure 1. 80C88A CPU Functional Block Diagram

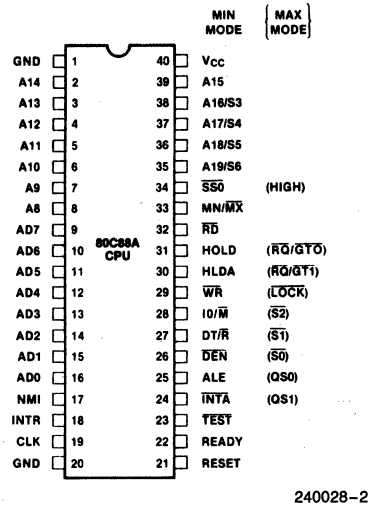


Figure 2. 80C88A 40-Lead DIP Configuration

September 1988
Order Number: 240028-002

Table 1. Pin Description

The following pin function descriptions are for 80C88A systems in either minimum or maximum mode. The "local bus" in these descriptions is the direct multiplexed bus interface connection to the 80C88A (without regard to additional bus buffers).

Symbol	Pin No.	Type	Name and Function
AD7–AD0	9–16	I/O	**ADDRESS DATA BUS:** These lines constitute the time multiplexed memory/IO address (T1) and data (T2, T3, Tw, and T4) bus. These lines are active HIGH and float to 3-state OFF(1) during interrupt acknowledge and local bus "hold acknowledge".
A15–A8	2–8, 39	O	**ADDRESS BUS:** These lines provide address bits 8 through 15 for the entire bus cycle (T1–T4). These lines do not have to be latched by ALE to remain valid. A15–A8 are active HIGH and float to 3-state OFF(1) during interrupt acknowledge and local bus "hold acknowledge".
A19/S6, A18/S5, A17/S4, A16/S3	35–38	O	**ADDRESS/STATUS:** During T1, these are the four most significant address lines for memory operations. During I/O operations, these lines are LOW. During memory and I/O operations, status information is available on these lines during T2, T3, Tw, and T4. S6 is always low. The status of the interrupt enable flag bit (S5) is updated at the beginning of each clock cycle. S4 and S3 are encoded as shown. This information indicates which segment register is presently being used for data accessing. These lines float to 3-state OFF(1) during local bus "hold acknowledge".

S4	S3	CHARACTERISTICS
0(LOW)	0	Alternate Data
0	1	Stack
1(HIGH)	0	Code or None
1	1	Data
S6 is 0(LOW)		

$\overline{RD}$	32	O	**READ:** Read strobe indicates that the processor is performing a memory or I/O read cycle, depending on the state of the IO/$\overline{M}$ pin or S2. This signal is used to read devices which reside on the 80C88A local bus. $\overline{RD}$ is active LOW during T2, T3 and Tw of any read cycle, and is guaranteed to remain HIGH in T2 until the 80C88A local bus has floated. This signal floats to 3-state OFF(1) in "hold acknowledge".
READY	22	I	**READY:** is the acknowledgement from the addressed memory or I/O device that it will complete the data transfer. The RDY signal from memory or I/O is synchronized by the 82C84A clock generator to form READY. This signal is active HIGH. The 80C88A READY input is not synchronized. Correct operation is not guaranteed if the set up and hold times are not met.

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
INTR	18	I	**INTERRUPT REQUEST:** is a level triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored to via an interrupt vector lookup table located in system memory. It can be internally masked by software resetting the interrupt enable bit. INTR is internally synchronized. This signal is active HIGH.
$\overline{TEST}$	23	I	**$\overline{TEST}$:** input is examined by the "wait for test" instruction. If the $\overline{TEST}$ input is LOW, execution continues, otherwise the processor waits in an "idle" state. This input is synchronized internally during each clock cycle on the leading edge of CLK.
NMI	17	I	**NON-MASKABLE INTERRUPT:** is an edge triggered input which causes a type 2 interrupt. A subroutine is vectored to via an interrupt vector lookup table located in system memory. NMI is not maskable internally by software. A transition from a LOW to HIGH initiates the interrupt at the end of the current instruction. This input is internally synchronized.
RESET	21	I	**RESET:** causes the processor to immediately terminate its present activity. The signal must be active HIGH for at least four clock cycles. It restarts execution, as described in the instruction set description, when RESET returns LOW. RESET is internally synchronized.
CLK	19	I	**CLOCK:** provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing.
V_{CC}	40		**V_{CC}:** is the +5V ±10% power supply pin.
GND	1, 20		**GND:** are the ground pins. Both must be connected.
MN/$\overline{MX}$	33	I	**MINIMUM/MAXIMUM:** indicates what mode the processor is to operate in. The two modes are discussed in the following sections.

The following pin function descriptions are for the 80C88A minimum mode (i.e., MN/$\overline{MX}$ = V_{CC}). Only the pin functions which are unique to minimum mode are described; all other pin functions are as described above.

IO/$\overline{M}$	28	O	**STATUS LINE:** is an inverted maximum mode $\overline{S2}$. It is used to distinguish a memory access from an I/O access. IO/$\overline{M}$ becomes valid in the T4 preceding a bus cycle and remains valid until the final T4 of the cycle (I/O = HIGH, M = LOW). IO/$\overline{M}$ floats to 3-state OFF(1) in local bus "hold acknowledge".
$\overline{WR}$	29	O	**WRITE:** strobe indicates that the processor is performing a write memory or write I/O cycle, depending on the state of the IO/$\overline{M}$ signal. WR is active for T2, T3, and Tw of any write cycle. It is active LOW, and floats to 3-state OFF(1) in local bus "hold acknowledge".
$\overline{INTA}$	24	O	**INTA:** is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T2, T3, and Tw of each interrupt acknowledge cycle.

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
ALE	25	O	**ADDRESS LATCH ENABLE:** is provided by the processor to latch the address into an address latch. It is a HIGH pulse active during clock low of T1 of any bus cycle. Note that ALE is never floated.
DT/$\overline{R}$	27	O	**DATA TRANSMIT/RECEIVE:** is needed in a minimum system that desires to use a data bus transceiver. It is used to control the direction of data flow through the transceiver. Logically, DT/$\overline{R}$ is equivalent to $\overline{S1}$ in the maximum mode, and its timing is the same as for IO/$\overline{M}$ (T = HIGH, R = LOW). This signal floats to 3-state OFF(1) in local "hold acknowledge".
$\overline{DEN}$	26	O	**DATA ENABLE:** is provided as an output enable for the transceiver in a minimum system which uses the transceiver. $\overline{DEN}$ is active LOW during each memory and I/O access, and for $\overline{INTA}$ cycles. For a read or $\overline{INTA}$ cycle, it is active from the middle of T2 until the middle of T4, while for a write cycle, it is active from the beginning of T2 until the middle of T4. $\overline{DEN}$ floats to 3-state OFF(1) during local bus "hold acknowledge".
HOLD, HLDA	30, 31	I, O	**HOLD:** indicates that another master is requesting a local bus "hold". To be acknowledged, HOLD must be active HIGH. The processor receiving the "hold" request will issue HLDA (HIGH) as an acknowledgement, in the middle of a T4 or Ti clock cycle. Simultaneous with the issuance of HLDA the processor will float the local bus and control lines. After HOLD is detected as being LOW, the processor lowers HLDA, and when the processor needs to run another cycle, it will again drive the local bus and control lines. Hold is not an asynchronous input. External synchronization should be provided if the system cannot otherwise guarantee the set up time.
$\overline{SS0}$	34	O	**STATUS LINE:** is logically equivalent to $\overline{S0}$ in the maximum mode. The combination of $\overline{SS0}$, IO/$\overline{M}$ and DT/$\overline{R}$ allows the system to completely decode the current bus cycle status.

IO/$\overline{M}$	DT/$\overline{R}$	$\overline{SS0}$	CHARACTERISTICS
1(HIGH)	0	0	Interrupt Acknowledge
1	0	1	Read I/O port
1	1	0	Write I/O port
1	1	1	Halt
0(LOW)	0	0	Code access
0	0	1	Read memory
0	1	0	Write memory
0	1	1	Passive

Table 1. Pin Description (Continued)

The following pin function descriptions are for the 80C88A/82C88 system in maximum mode (i.e., MN/$\overline{MX}$ = GND.) Only the pin functions which are unique to maximum mode are described; all other pin functions are as described above.

<table>
<tr><th>Symbol</th><th>Pin No.</th><th>Type</th><th>Name and Function</th></tr>
<tr><td>$\overline{S2}$, $\overline{S1}$, $\overline{S0}$</td><td>26–28</td><td>O</td><td>STATUS: is active during clock high of T4, T1, and T2, and is returned to the passive state (1,1,1) during T3 or during Tw when READY is HIGH. This status is used by the 82C88 bus controller to generate all memory and I/O access control signals. Any change by $\overline{S2}$, $\overline{S1}$, or $\overline{S0}$ during T4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T3 or Tw is used to indicate the end of a bus cycle.

These signals float to 3-state OFF(1) during "hold acknowledge". During the first clock cycle after RESET becomes active, these signals are active HIGH. After this first clock, they float to 3-state OFF.
<table>
<tr><th>$\overline{S2}$</th><th>$\overline{S1}$</th><th>$\overline{S0}$</th><th>CHARACTERISTICS</th></tr>
<tr><td>0(LOW)</td><td>0</td><td>0</td><td>Interrupt Acknowledge</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Read I/O port</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Write I/O port</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Halt</td></tr>
<tr><td>1(HIGH)</td><td>0</td><td>0</td><td>Code access</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Read memory</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Write memory</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Passive</td></tr>
</table>
</td></tr>
<tr><td>$\overline{RQ}$/$\overline{GT0}$,
$\overline{RQ}$/$\overline{GT1}$</td><td>30, 31</td><td>I/O</td><td>REQUEST/GRANT: pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional with $\overline{RQ}$/$\overline{GT0}$ having higher priority than $\overline{RQ}$/$\overline{GT1}$. $\overline{RQ}$/$\overline{GT}$ has an internal pull-up resistor, so may be left unconnected. The request/grant sequence is as follows (see timing diagram):

1. A pulse of one CLK wide from another local bus master indicates a local bus request ("hold") to the 80C88A (pulse 1).

2. During a T4 or T1 clock cycle, a pulse one clock wide from the 80C88A to the requesting master (pulse 2), indicates that the 80C88A has allowed the local bus to float and that it will enter the "hold acknowledge" state at the next CLK. The CPU's bus interface unit is disconnected logically from the local bus during "hold acknowledge". The same rules as for HOLD/HOLDA apply as for when the bus is released.

3. A pulse one CLK wide from the requesting master indicates to the 80C88A (pulse 3) that the "hold" request is about to end and that the 80C88A can reclaim the local bus at the next CLK. The CPU then enters T4.</td></tr>
</table>

Table 1. Pin Descriptions (Continued)

Symbol	Pin No.	Type	Name and Function
$\overline{RQ}/\overline{GT0}$, $\overline{RQ}/\overline{GT1}$	30, 31	I/O	Each master-master exchange of the local bus is a sequence of three pulses. There must be one idle CLK cycle after each bus exchange. Pulses are active LOW. If the request is made while the CPU is performing a memory cycle, it will release the local bus during T4 of the cycle when all the following conditions are met: 1. Request occurs on or before T2. 2. Current cycle is not the low bit of a word. 3. Current cycle is not the first acknowledge of an interrupt acknowledge sequence. 4. A locked instruction is not currently executing. If the local bus is idle when the request is made the two possible events will follow: 1. Local bus will be released during the next clock. 2. A memory cycle will start within 3 clocks. Now the four rules for a currently active memory cycle apply with condition number 1 already satisfied.
$\overline{LOCK}$	29	O	**$\overline{LOCK}$:** indicates that other system bus masters are not to gain control of the system bus while $\overline{LOCK}$ is active (LOW). The $\overline{LOCK}$ signal is activated by the "LOCK" prefix instruction and remains active until the completion of the next instruction. This signal is active LOW, and floats to 3-state OFF(1) in "hold acknowledge".
QS1, QS0	24, 25	O	**QUEUE STATUS:** provide status to allow external tracking of the internal 80C88A instruction queue. The queue status is valid during the CLK cycle after which the queue operation is performed.

QS1	QS0	CHARACTERISTICS
0(LOW)	0	No operation
0	1	First byte of opcode from queue
1(HIGH)	0	Empty the queue
1	1	Subsequent byte from queue

—	34	O	Pin 34 is always high in the maximum mode.

NOTE:
1. See the section on Bus Hold Circuitry.

FUNCTIONAL DESCRIPTION

STATIC OPERATION

All 80C88A circuitry is of static design. Internal registers, counters and latches are static and require no refresh as with dynamic circuit design. This eliminates the minimum operating frequency restriction placed on other microprocessors. The CMOS 80C88A can operate from DC to the appropriate upper frequency limit. The processor clock may be stopped in either state (high/low) and held there indefinitely. This type of operation is especially useful for system debug or power critical applications.

The 80C88A can be single stepped using only the CPU clock. This state can be maintained as long as is necessary. Single step clock operation allows simple interface circuitry to provide critical information for bringing up your system.

Static design also allows very low frequency operation. In a power critical situation, this can provide extremely low power operation since 80C88A power dissipation is directly related to operating frequency. As the system frequency is reduced, so is the operating power until ultimately, at a DC input frequency, the 80C88A power requirement is the standby current.

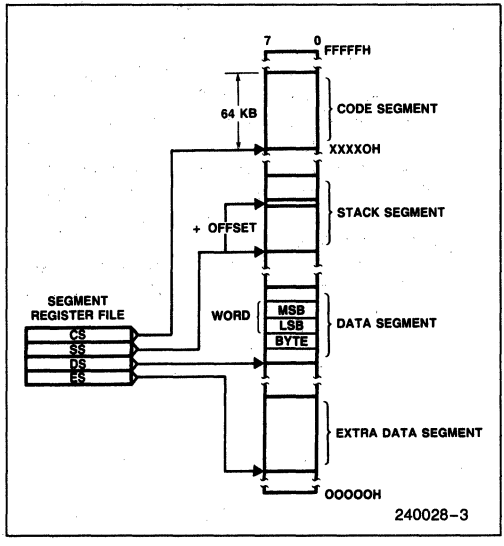


Figure 3. Memory Organization

MEMORY ORGANIZATION

The processor provides a 20-bit address to memory which locates the byte being referenced. The memory is organized as a linear array of up to 1 million bytes, addressed as 00000(H) to FFFFF(H). The memory is logically divided into code, data, extra data, and stack segments of up to 64K bytes each, with each segment falling on 16-byte boundaries. (See Figure 3.)

All memory references are made relative to base addresses contained in high speed segment registers. The segment types were chosen based on the addressing needs of programs. The segment register to be selected is automatically chosen according to the rules of the following table. All information in one segment type share the same logical attributes (e.g. code or data). By structuring memory into relocatable areas of similar characteristics and by automatically selecting segment registers, programs are shorter, faster, and more structured.

Word (16-bit) operands can be located on even or odd address boundaries. For address and data operands, the least significant byte of the word is stored in the lower valued address location and the most significant byte in the next higher address location. The BIU will automatically execute two fetch or write cycles for 16-bit operands.

Certain locations in memory are reserved for specific CPU operations. (See Figure 4.) Locations from addresses FFFF0H through FFFFFH are reserved for operations including a jump to the initial system

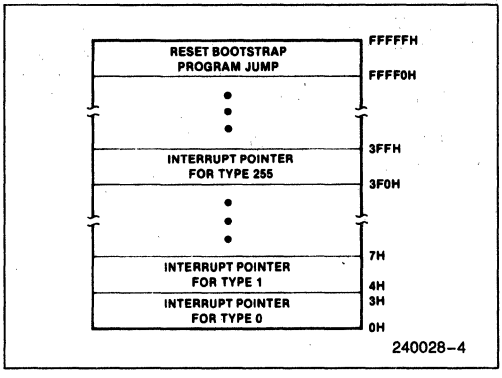


Figure 4. Reserved Memory Locations

Memory Reference Need	Segment Register Used	Segment Selection Rule
Instructions	CODE (CS)	Automatic with all instruction prefetch.
Stack	STACK (SS)	All stack pushes and pops. Memory references relative to BP base register except data references.
Local Data	DATA (DS)	Data references when: relative to stack, destination of string operation, or explicitly overridden.
External (Global) Data	EXTRA (ES)	Destination of string operations: Explicitly selected using a segment override.

initialization routine. Following RESET, the CPU will always begin execution at location FFFF0H where the jump must be located. Locations 00000H through 003FFH are reserved for interrupt operations. Four-byte pointers consisting of a 16-bit segment address and a 16-bit offset address direct program flow to one of the 256 possible interrupt service routines. The pointer elements are assumed to have been stored at their respective places in reserved memory prior to the occurrence of interrupts.

MINIMUM AND MAXIMUM MODES

The requirements for supporting minimum and maximum 80C88A systems are sufficiently different that they cannot be done efficiently with 40 uniquely defined pins. Consequently, the 80C88A is equipped with a strap pin (MN/$\overline{MX}$) which defines the system configuration. The definition of a certain subset of the pins changes, dependent on the condition of the strap pin. When the MN/$\overline{MX}$ pin is strapped to GND, the 80C88A defines pins 24 through 31 and 34 in maximum mode. When the MN/$\overline{MX}$ pin is strapped to V_{CC}, the 80C88A generates bus control signals itself on pins 24 through 31 and 34.

The minimum mode 80C88A can be used with either a multiplexed or demultiplexed bus. The multiplexed bus configuration is compatible with the MCS®-85 multiplexed bus peripherals (8155, 8156, 8355, 8755A, and 8185). This configuration (See Figure 5) provides the user with a minimum chip count system. This architecture provides the 80C88A processing power in a highly integrated form.

The demultiplexed mode requires one latch (for 64k addressability) or two latches (for a full megabyte of addressing). A third latch can be used for buffering if the address bus loading requires it. A transceiver can also be used if data bus buffering is required. (See Figure 6.) The 80C88A provides $\overline{DEN}$ and DT/$\overline{R}$ to control the transceiver, and ALE to latch the addresses. This configuration of the minimum mode provides the standard demultiplexed bus structure with heavy bus buffering and relaxed bus timing requirements.

The maximum mode employs the 82C88 bus controller. (See Figure 7.) The 82C88 decodes status lines $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$, and provides the system with all bus control signals. Moving the bus control to the 82C88 provides better source and sink current capability to the control lines, and frees the 80C88A pins for extended large system features. Hardware lock, queue status, and two request/grant interfaces are provided by the 80C88A in maximum mode. These features allow co-processors in local bus and remote bus configurations.

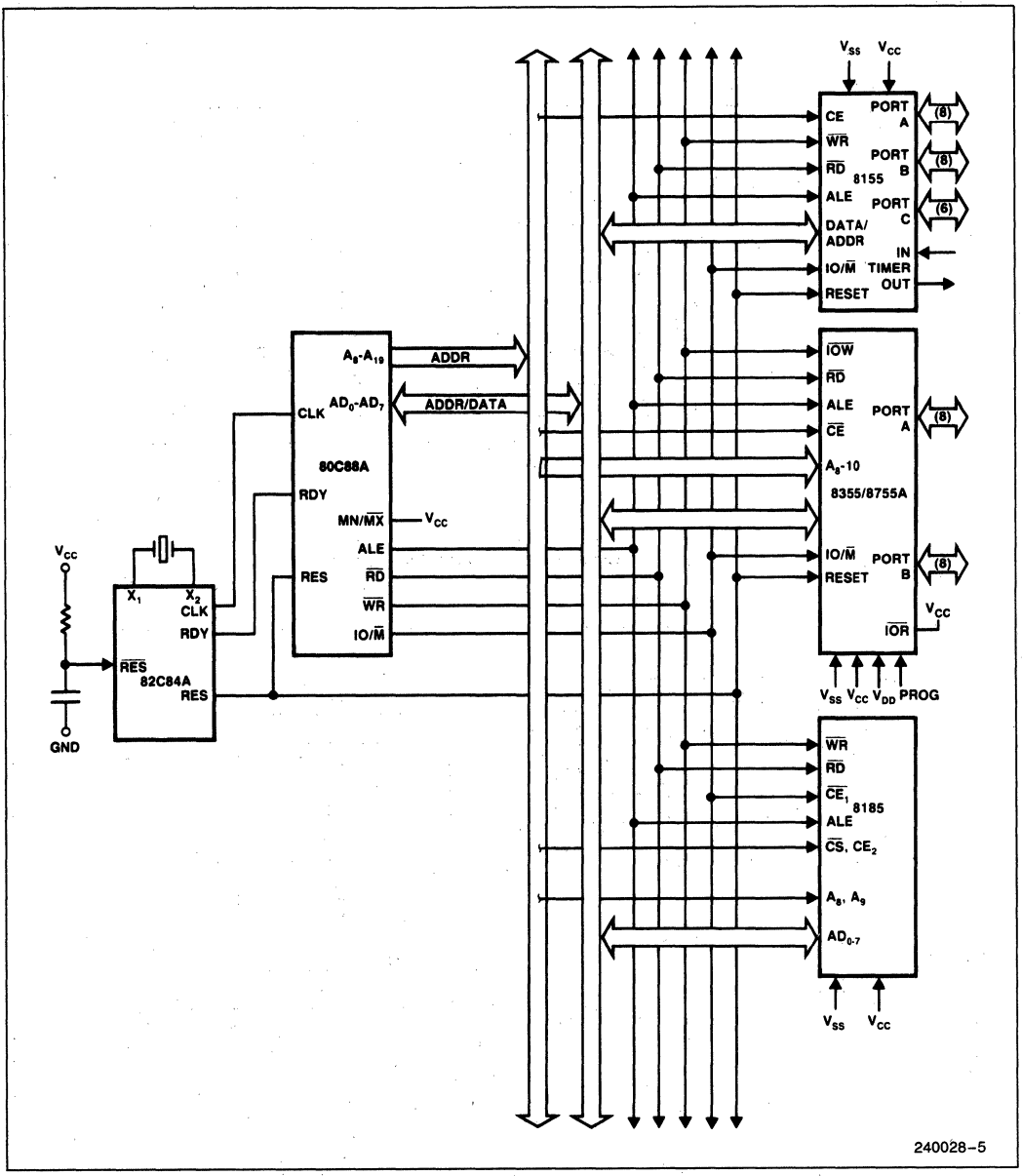


Figure 5. Multiplexed Bus Configuration

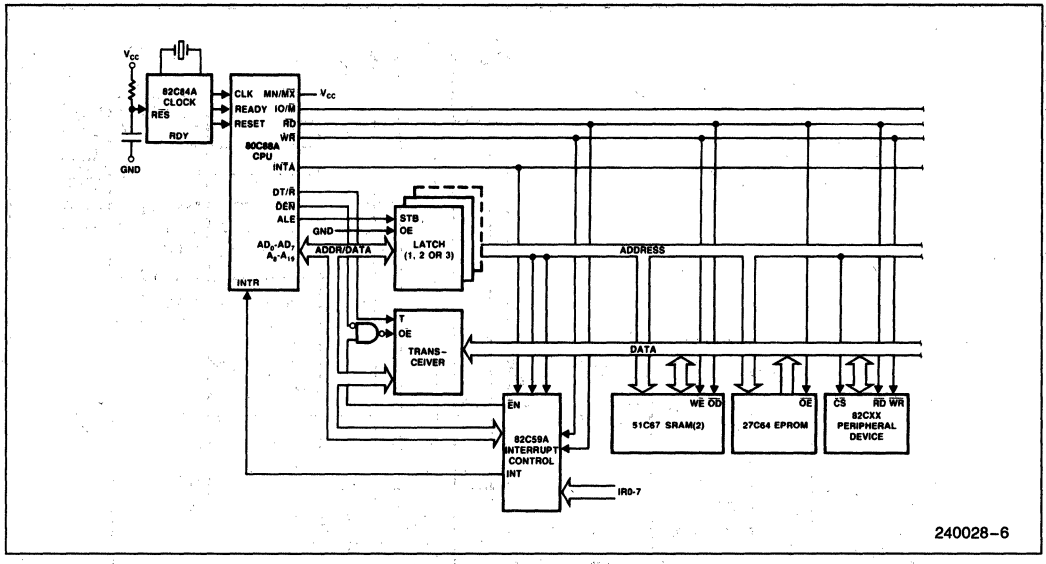


Figure 6. Demultiplexed Bus Configuration

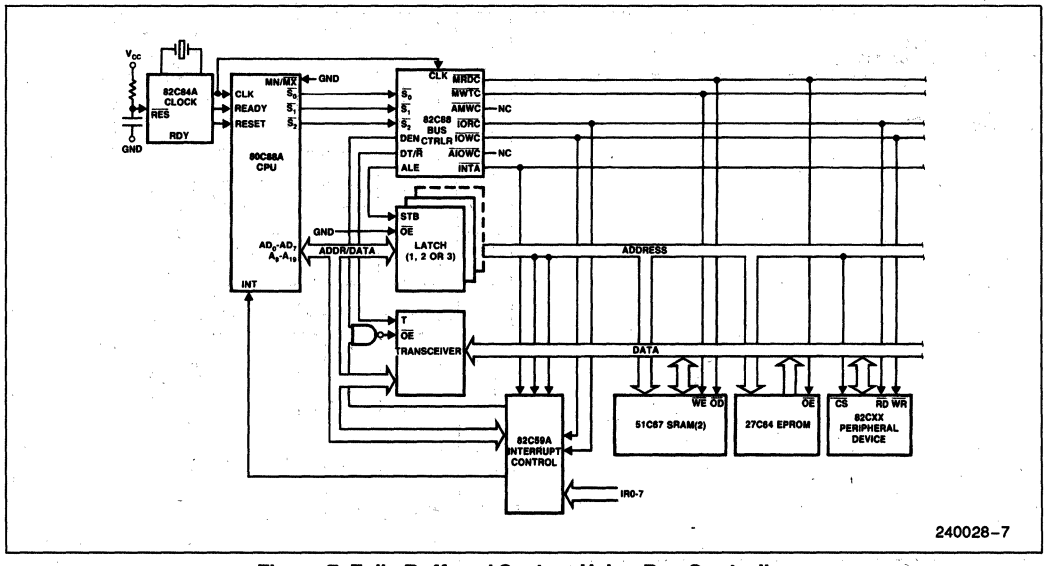


Figure 7. Fully Buffered System Using Bus Controller

Bus Operation

The 80C88A address/data bus is broken into three parts—the lower eight address/data bits (AD0–AD7), the middle eight address bits (A8–A15), and the upper four address bits (A16–A19). The address/data bits and the highest four address bits are time multiplexed. This technique provides the most efficient use of pins on the processor. The middle eight address bits are not multiplexed, i.e. they remain valid throughout each bus cycle. In addition, the bus can be demultiplexed at the processor with a single address latch if a standard, non-multiplexed bus is desired for the system.

Each processor bus cycle consists of at least four CLK cycles. These are referred to as T1, T2, T3, and T4. (See Figure 8). The address is emitted from the processor during T1 and data transfer occurs on the bus during T3 and T4. T2 is used primarily for changing the direction of the bus during read operations. In the event that a "NOT READY" indication is given by the addressed device, "wait" states (Tw) are inserted between T3 and T4. Each inserted "wait" state is of the same duration as a CLK cycle. Periods can occur between 80C88A driven bus cycles. These are referred to as "idle" states (Ti), or inactive CLK cycles. The processor uses these cycles for internal housekeeping.

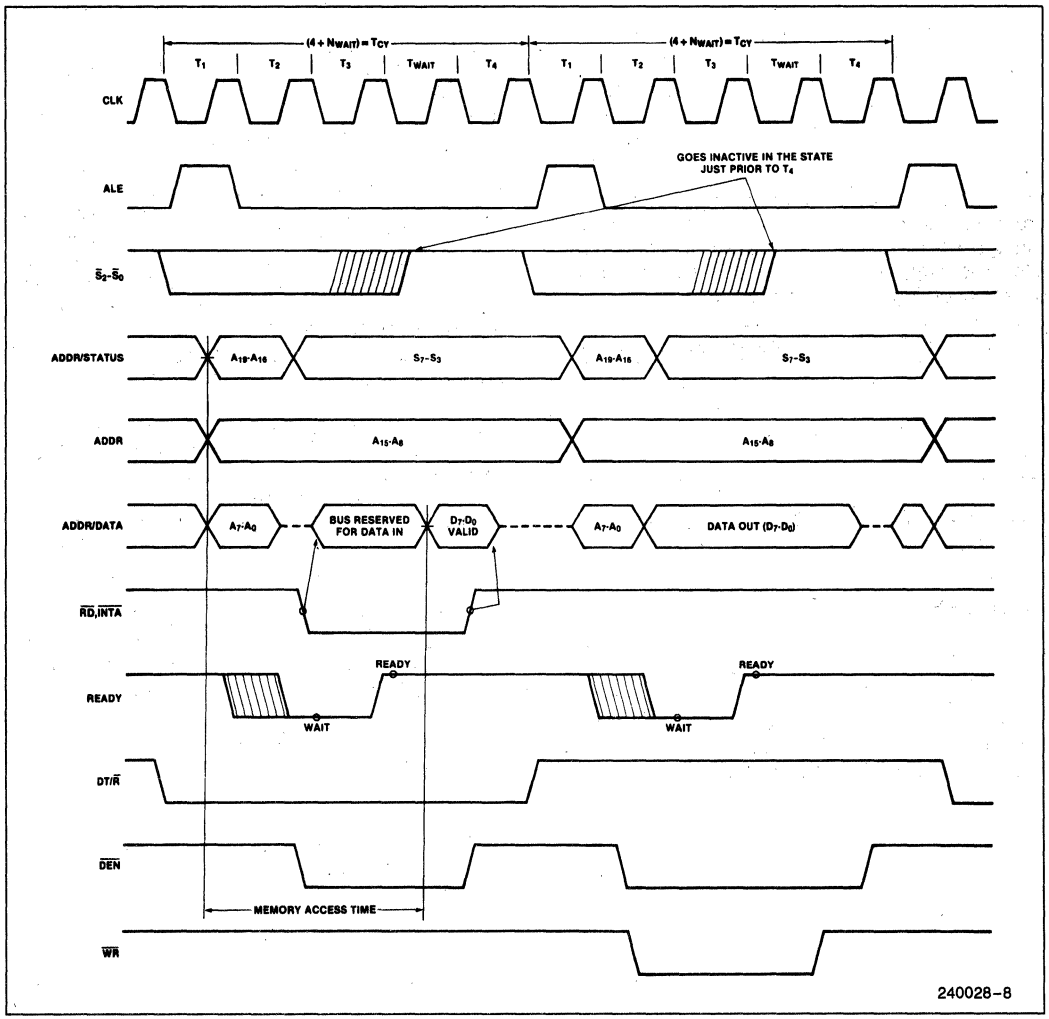


Figure 8. Basic System Timing

During T1 of any bus cycle, the ALE (address latch enable) signal is emitted (by either the processor or the 82C88 bus controller, depending on the MN/$\overline{MX}$ strap). At the trailing edge of this pulse, a valid address and certain status information for the cycle may be latched.

Status bits $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$ are used by the bus controller, in maximum mode, to identify the type of bus transaction according to the following table:

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	CHARACTERISTICS
0 (LOW)	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1 (HIGH)	0	0	Instruction Fetch
1	0	1	Read Data from Memory
1	1	0	Write Data to Memory
1	1	1	Passive (no bus cycle)

Status bits S3 through S6 are multiplexed with high order address bits and are therefore valid during T2 through T4. S3 and S4 indicate which segment register was used for this bus cycle in forming the address according to the following table:

S_4	S_3	CHARACTERISTICS
0 (LOW)	0	Alternate Data (extra segment)
0	1	Stack
1 (HIGH)	0	Code or None
1	1	Data

S5 is a reflection of the PSW interrupt enable bit. S6 is equal to 0.

I/O ADDRESSING

In the 80C88A, I/O operations can address up to a maximum of 64k I/O registers. The I/O address appears in the same format as the memory address on bus lines A15–A0. The address lines A19–A16 are zero in I/O operations. The variable I/O instructions, which use register DX as a pointer, have full address capability, while the direct I/O instructions directly address one or two of the 256 I/O byte locations in page 0 of the I/O address space. I/O ports are addressed in the same manner as memory locations.

Designers familiar with the 8085 or upgrading an 8085 design should note that the 8085 addresses I/O with an 8-bit address on both halves of the 16-bit address bus. The 80C88A uses a full 16-bit address on its lower 16 address lines.

EXTERNAL INTERFACE

PROCESSOR RESET AND INITIALIZATION

Processor initialization or start up is accomplished with activation (HIGH) of the RESET pin. The 80C88A RESET is required to be HIGH for four or more clock cycles. The 80C88A will terminate operations on the high-going edge of RESET and will remain dormant as long as RESET is HIGH. The low-going transition of RESET triggers an internal reset sequence for approximately 7 clock cycles. After this interval the 80C88A operates normally, beginning with the instruction in absolute location FFFF0H. (See Figure 4.) The RESET input is internally synchronized to the processor clock. At initialization, the HIGH to LOW transition of RESET must occur no sooner than 50 μs after power up, to allow complete initialization of the 80C88A.

NMI asserted prior to the 2nd clock after the end of RESET will not be honored. If NMI is asserted after that point and during the internal reset sequence, the processor may execute one instruction before responding to the interrupt. A hold request active immediately after RESET will be honored before the first instruction fetch.

All 3-state outputs float to 3-state OFF(1) during RESET. Status is active in the idle state for the first clock after RESET becomes active and then floats to 3-state OFF(1). ALE and HLDA are driven low.

NOTE:
1. See the section on Bus Hold Circuitry.

BUS HOLD CIRCUITRY

To avoid high current conditions caused by floating inputs to CMOS devices and to eliminate the need for pull-up/down resistors, "bus-hold" circuitry has been used on the 80C88A pins 2–16, 26–32, and 34–39 (Figure 9a, 9b). These circuits will maintain the last valid logic state if no driving source is present (i.e. an unconnected pin or a driving source which goes to a high impedance state). To overdrive the "bus hold" circuits, an external driver must be capable of supplying 350 μA minimum sink or source current at valid input voltage levels. Since this "bus hold" circuitry is active and not a "resistive" type element, the associated power supply current is negligible and power dissipation is significantly reduced when compared to the use of passive pull-up resistors.

INTERRUPT OPERATIONS

Interrupt operations fall into two classes: software or hardware initiated. The software initiated interrupts and software aspects of hardware interrupts are specified in the instruction set description in the iAPX 88 book or the iAPX 86,88 User's Manual. Hardware interrupts can be classified as nonmaskable or maskable.

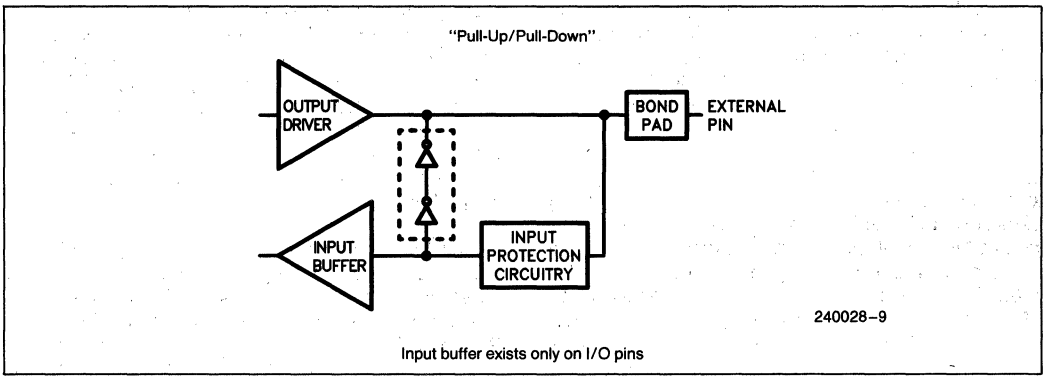


Figure 9a. Bus hold circuitry pin 2–16, 35–39.

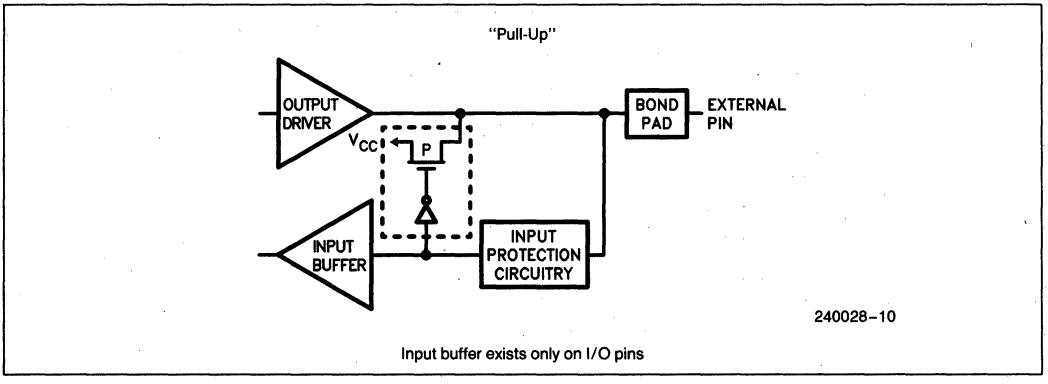


Figure 9b. Bus hold circuitry pin 26–32, 34.

Interrupts result in a transfer of control to a new program location. A 256 element table containing address pointers to the interrupt service program locations resides in absolute locations 0 through 3FFH (See Figure 4), which are reserved for this purpose. Each element in the table is 4 bytes in size and corresponds to an interrupt "type." An interrupting device supplies an 8-bit type number, during the interrupt acknowledge sequence, which is used to vector through the appropriate element to the new interrupt service program location.

NON-MASKABLE INTERRUPT (NMI)

The processor provides a single non-maskable interrupt (NMI) pin which has higher priority than the maskable interrupt request (INTR) pin. A typical use would be to activate a power failure routine. The NMI is edge-triggered on a LOW to HIGH transition. The activation of this pin causes a type 2 interrupt.

NMI is required to have a duration in the HIGH state of greater than two clock cycles, but is not required to be synchronized to the clock. Any higher going transition of NMI is latched on-chip and will be serviced at the end of the current instruction or between whole moves (2 bytes in the case of word moves) of a block type instruction. Worst case response to NMI would be for multiply, divide, and variable shift instructions. There is no specification on the occurrence of the low-going edge; it may occur before, during, or after the servicing of NMI. Another high-going edge triggers another response if it occurs after the start of the NMI procedure. The signal must be free of logical spikes in general and be free of bounces on the low-going edge to avoid triggering extraneous responses.

MASKABLE INTERRUPT (INTR)

The 80C88A provides a single interrupt request input (INTR) which can be masked internally by software with the resetting of the interrupt enable (IF) flag bit. The interrupt request signal is level triggered. It is internally synchronized during each clock cycle on the high-going edge of CLK. To be responded to, INTR must be present (HIGH) during the clock period preceding the end of the current instruction or the end of a whole move for a block type instruction. During interrupt response sequence, further interrupts are disabled. The enable bit is reset as part of the response to any interrupt (INTR, NMI, software interrupt, or single step), although the FLAGS register which is automatically pushed onto the stack reflects the state of the processor prior to the interrupt. Until the old FLAGS register is restored, the enable bit will be zero unless specifically set by an instruction.

During the response sequence (See Figure 10), the processor executes two successive (back to back) interrupt acknowledge cycles. The 80C88A emits the LOCK signal (maximum mode only) from T2 of the first bus cycle until T2 of the second. A local bus "hold" request will not be honored until the end of the second bus cycle. In the second bus cycle, a

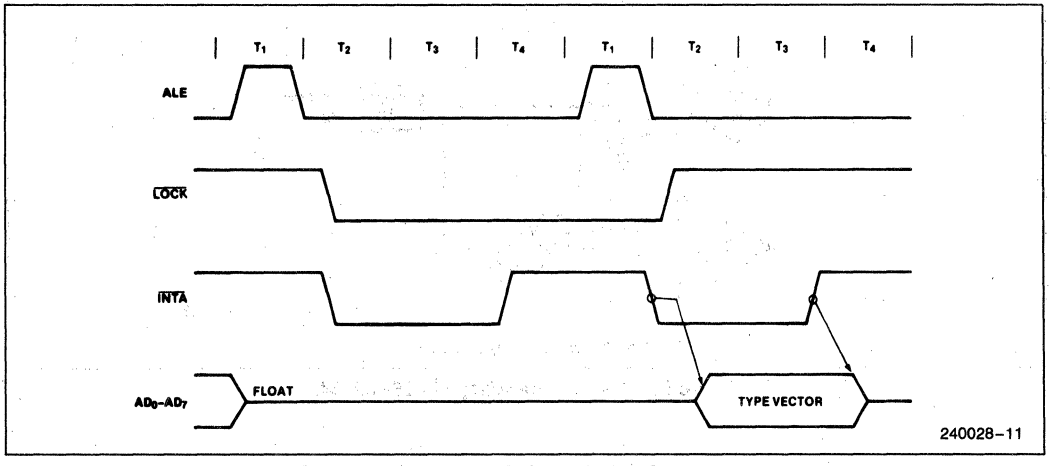


Figure 10. Interrupt Acknowledge Sequence

byte is fetched from the external interrupt system (e.g., 82C59A PIC) which identifies the source (type) of the interrupt. This byte is multiplied by four and used as a pointer into the interrupt vector lookup table. An INTR signal left HIGH will be continually responded to within the limitations of the enable bit and sample period. The interrupt return instruction includes a flags pop which returns the status of the original interrupt enable bit when it restores the flags.

HALT

When a software HALT instruction is executed, the processor indicates that it is entering the HALT state in one of two ways, depending upon which mode is strapped. In minimum mode, the processor issues ALE, delayed by one clock cycle, to allow the system to latch the halt status. Halt status is available on IO/$\overline{M}$, DT/$\overline{R}$, and $\overline{SSO}$. In maximum mode, the processor issues appropriate HALT status on $\overline{S2}$, $\overline{S1}$, and $\overline{S0}$, and the 82C88 bus controller issues one ALE. The 80C88A will not leave the HALT state when a local bus hold is entered while in HALT. In this case, the processor reissues the HALT indicator at the end of the local bus hold. An interrupt request or RESET will force the 80C88A out of the HALT state.

READ/MODIFY/WRITE (SEMAPHORE) OPERATIONS VIA LOCK

The LOCK status information is provided by the processor when consecutive bus cycles are required during the execution of an instruction. This allows the processor to perform read/modify/write operations on memory (via the "exchange register with memory" instruction), without another system bus master receiving intervening memory cycles. This is useful in multiprocessor system configurations to accomplish "test and set lock" operations. The $\overline{LOCK}$ signal is activated (LOW) in the clock cycle following decoding of the LOCK prefix instruction. It is deactivated at the end of the last bus cycle of the instruction following the LOCK prefix. While $\overline{LOCK}$ is active, a request on a $\overline{RQ}/\overline{GT}$ pin will be recorded, and then honored at the end of the LOCK.

EXTERNAL SYNCHRONIZATION VIA $\overline{TEST}$

As an alternative to interrupts, the 80C88A provides a single software-testable input pin ($\overline{TEST}$). This input is utilized by executing a WAIT instruction. The single WAIT instruction is repeatedly executed until the $\overline{TEST}$ input goes active (LOW). The execution of WAIT does not consume bus cycles once the queue is full.

If a local bus request occurs during WAIT execution, the 80C88A 3-states all output drivers. If interrupts are enabled, the 80C88A will recognize interrupts and process them. The WAIT instruction is then refetched, and reexecuted.

BASIC SYSTEM TIMING

In minimum mode, the MN/$\overline{MX}$ pin is strapped to V_{CC} and the processor emits bus control signals compatible with the 8085 bus structure. In maximum mode, the MN/$\overline{MX}$ pin is strapped to GND and the processor emits coded status information which the 82C88 bus controller uses to generate MULTIBUS compatible bus control signals.

System Timing — Minimum System

(See Figure 8.)

The read cycle begins in T1 with the assertion of the address latch enable (ALE) signal. The trailing (low going) edge of this signal is used to latch the address information, which is valid on the address/data bus (AD0–AD7) at this time, into a latch. Address lines A8 through A15 do not need to be latched because they remain valid throughout the bus cycle. From T1 to T4 the IO/$\overline{M}$ signal indicates a memory or I/O operation. At T2 the address is removed from the address/data bus and the bus goes to a high impedance state. The read control signal is also asserted at T2. The read ($\overline{RD}$) signal causes the addressed device to enable its data bus drivers to the local bus. Some time later, valid data will be available on the bus and the addressed device will drive the READY line HIGH. When the processor returns the read signal to a HIGH level, the addressed device will again 3-state its bus drivers. If a transceiver is required to buffer the 80C88A local bus, signals DT/$\overline{R}$ and $\overline{DEN}$ are provided by the 80C88A.

A write cycle also begins with the assertion of ALE and the emission of the address. The IO/$\overline{M}$ signal is again asserted to indicate a memory or I/O write operation. In T2, immediately following the address emission, the processor emits the data to be written into the addressed location. This data remains valid until at least the middle of T4. During T2, T3, and T_W, the processor asserts the write control signal. The write ($\overline{WR}$) signal becomes active at the beginning of T2, as opposed to the read, which is delayed somewhat into T2 to provide time for the bus to float.

The basic difference between the interrupt acknowledge cycle and a read cycle is that the interrupt acknowledge ($\overline{INTA}$) signal is asserted in place of the read ($\overline{RD}$) signal and the address bus is floated. (See Figure 10.) In the second of two successive $\overline{INTA}$ cycles, a byte of information is read from the data bus, as supplied by the interrupt system logic (i.e. 82C59A priority interrupt controller). This byte identifies the source (type) of the interrupt. It is multiplied by four and used as a pointer into the interrupt vector lookup table, as described earlier.

BUS TIMING — MEDIUM COMPLEXITY SYSTEMS

(See Figure 11.)

For medium complexity systems, the MN/$\overline{MX}$ pin is connected to GND and the 82C88 bus controller is added to the system, as well as a latch for latching the system address, and a transceiver to allow for bus loading greater than the 80C88A is capable of handling. Signals ALE, $\overline{DEN}$, and DT/$\overline{R}$ are generated by the 82C88 instead of the processor in this configuration, although their timing remains relatively the same. The 80C88A status outputs ($\overline{S2}$, $\overline{S1}$, and $\overline{S0}$) provide type of cycle information and become 82C88 inputs. This bus cycle information specifies read (code, data, or I/O), write (data or I/O), interrupt acknowledge, or software halt. The 82C88 thus issues control signals specifying memory read or write, I/O read or write, or interrupt acknowledge. The 82C88 provides two types of write strobes, normal and advanced, to be applied as required. The normal write strobes have data valid at the leading edge of write. The advanced write strobes have the same timing as read strobes, and hence, data is not valid at the leading edge of write. The transceiver receives the usual $\overline{T}$ and $\overline{OE}$ inputs from the 82C88's DT/$\overline{R}$ and $\overline{DEN}$ outputs.

The pointer into the interrupt vector table, which is passed during the second $\overline{INTA}$ cycle, can derive from an 82C59A located on either the local bus or the system bus. If the master 82C59A priority interrupt controller is positioned on the local bus, a TTL gate is required to disable the transceiver when reading from the master 82C59A during the interrupt acknowledge sequence and software "poll".

THE 80C88A COMPARED TO THE 80C86

The 80C88A CPU is an 8-bit processor designed around the 80C86 internal structure. Most internal functions of the 80C88A are identical to the equivalent 80C86 functions. The 80C88A handles the external bus the same way the 80C86 does with the distinction of handling only 8 bits at a time. Sixteen-bit operands are fetched or written in two consecutive bus cycles. Both processors will appear identical to the software engineer, with the exception of execution time. The internal register structure is identical and all instructions have the same end result. The differences between the 80C88A and 80C86 are outlined below. The engineer who is unfamiliar with the 80C86 is referred to the iAPX 86, 88 User's Manual, Chapters 2 and 4, for function description and instruction set information. Internally, there are three differences between the 80C88A and the 80C86. All changes are related to the 8-bit bus interface.

- The queue length is 4 bytes in the 80C88A, whereas the 80C86 queue contains 6 bytes, or three words. The queue was shortened to prevent overuse of the bus by the BIU when prefetching instructions. This was required because of the additional time necessary to fetch instructions 8 bits at a time.
- To further optimize the queue, the prefetching algorithm was changed. The 80C88A BIU will fetch a new instruction to load into the queue each time there is a 1 byte hole (space available) in the queue. The 80C86 waits until a 2-byte space is available.
- The internal execution time of the instruction set is affected by the 8-bit interface. All 16-bit fetches and writes from/to memory take an additional four clock cycles. The CPU is also limited by the speed of instruction fetches. This latter problem only occurs when a series of simple operations occur. When the more sophisticated instructions of the 80C88A are being used, the queue has time to fill and the execution proceeds as fast as the execution unit will allow.

The 80C88A and 80C86 are completely software compatible by virture of their identical execution units. Software that is system dependent may not be completely transferable, but software that is not system dependent will operate equally as well on an 80C88A or an 80C86.

The hardware interface of the 80C88A contains the major differences between the two CPUs. The pin assignments are nearly identical, however with the following functional changes:

- A8–A15 — These pins are only address outputs on the 80C88A. These address lines are latched internally and remain valid throughout a bus cycle in a manner similar to the 8085 upper address lines.

- $\overline{BHE}$ has no meaning on the 80C88A and has been eliminated.
- $\overline{SSO}$ provides the $\overline{SO}$ status information in the minimum mode. This output occurs on pin 34 in minimum mode only. DT/$\overline{R}$, IO/$\overline{M}$, and $\overline{SSO}$ provide the complete bus status in minimum mode.
- IO/$\overline{M}$ has been inverted to be compatible with the MCS-85 bus structure.
- ALE is delayed by one clock cycle in the minimum mode when entering HALT, to allow the status to be latched with ALE.

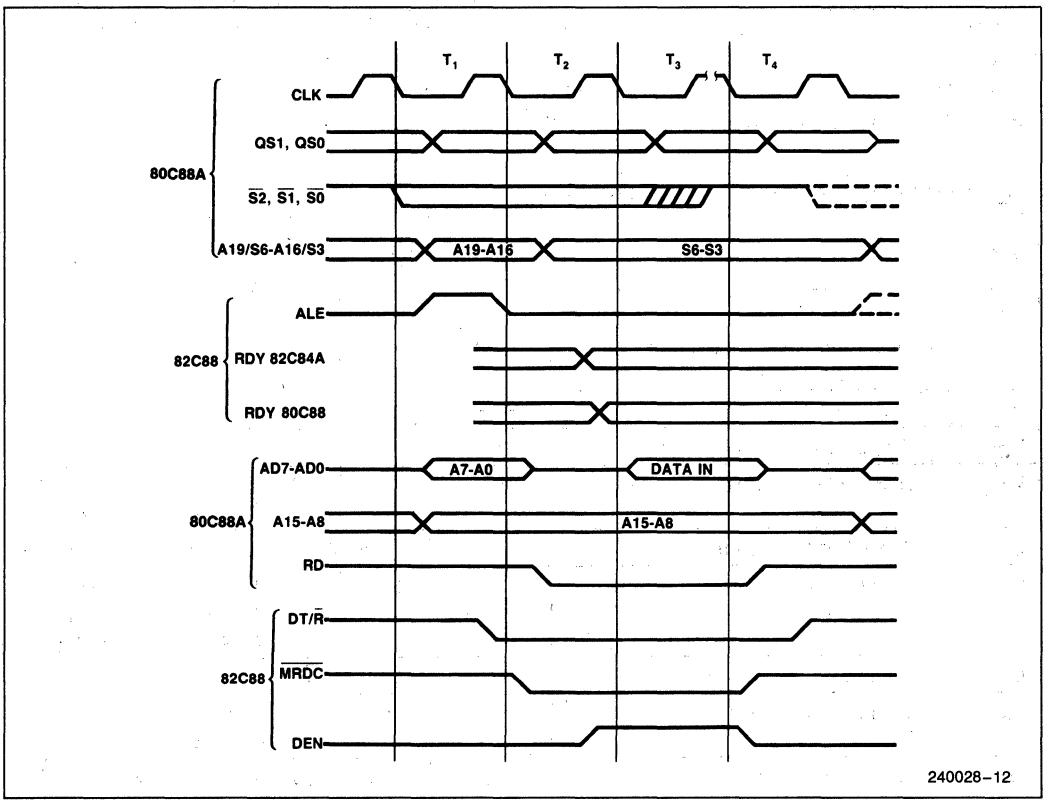


Figure 11. Medium Complexity System Timing

ABSOLUTE MAXIMUM RATINGS*

Supply Voltage
(With respect to ground) ..... −0.5 to 7.0V
Input Voltage Applied
(w.r.t. ground) ..... −0.5 to V_{CC} + 0.5V
Output Voltage Applied
(w.r.t. ground) ..... −0.5 to V_{CC} + 0.5V
Power Dissipation 1.0W
Storage Temperature ..... −65°C to +150°C
Ambient Temperature Under Bias ....0°C to +70°C

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±5%

Symbol	Parameter	80C88A-2		Units	Test Conditions
		Min	Max		
V_{IL}	Input Low Voltage	−0.5	+0.8	V	
V_{IH}	Input High Voltage (All inputs except clock)	2.0		V	
V_{CH}	Clock High Voltage	V_{CC}−0.8		V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.5 mA
V_{OH}	Output High Voltage	3.0 V_{CC}−0.4		V	I_{OH} = −2.5 mA I_{OH} = −100 μA
I_{CC}	Power Supply Current		10 mA/MHz		V_{IL} = GND, V_{IH} = V_{CC}
I_{CCS}	Standby Supply Current		500	μA	V_{IN} = V_{CC} or GND Outputs Unloaded CLK = GND or V_{CC}
I_{LI}	Input Leakage Current		±1.0	μA	$0V \le V_{IN} \le V_{CC}$
I_{BHL}	Input Leakage Current (Bus Hold Low)	50	400	μA	V_{IN} = 0.8V (Note 4)
I_{BHH}	Input Leakage Current (Bus Hold High)	−50	−400	μA	V_{IN} = 3.0V (Note 5)
I_{BHLO}	Bus Hold Low Overdrive		600	μA	(Note 2)
I_{BHHO}	Bus Hold High Overdrive		−600	μA	(Note 3)
I_{LO}	Output Leakage Current		±10	μA	V_{OUT} = GND or V_{CC}
C_{IN}	Capacitance of Input Buffer (All inputs except AD_0–AD_7, $\overline{RQ}/\overline{GT}$)		5	pF	(Note 1)
C_{IO}	Capacitance of I/O Buffer (AD_0–AD_7, $\overline{RQ}/\overline{GT}$)		20	pF	(Note 1)
C_{OUT}	Output Capacitance		15	pF	(Note 1)

NOTES:
1. Characterization conditions are a) Frequency = 1 MHz, b) Unmeasured pins at GND
   c) V_{IN} at +5.0V or GND.
2. An external driver must source at least I_{BHLO} to switch this node from LOW to HIGH.
3. An external driver must sink at least I_{BHHO} to switch this node from HIGH to LOW.
4. Test condition is to lower V_{IN} to GND and then raise V_{IN} to 0.8V on pins 2–16 and 34–39.
5. Test condition is to raise V_{IN} to V_{CC} and then lower V_{IN} to 3.0V on pins 2–16, 26–32 and 34–39.

A.C. CHARACTERISTICS $T_A = 0°C$ to $70°C$, $V_{CC} = 5V \pm 5\%$

MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

Symbol	Parameter	80C88A-2		Units	Test Conditions
		Min	Max		
TCLCL	CLK Cycle Period	125	D.C.	ns	
TCLCH	CLK Low Time	68		ns	
TCHCL	CLK High Time	44		ns	
TCH1CH2	CLK Rise Time		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10	ns	From 3.5V to 1.0V
TDVCL	Data in Setup Time	20		ns	
TCLDX	Data in Hold Time	10		ns	
TR1VCL	RDY Setup Time into 82C84A (Notes 1, 2)	35		ns	
TCLR1X	RDY Hold Time into 82C84A (Notes 1, 2)	0		ns	
TRYHCH	READY Setup Time into 80C88A	68		ns	
TCHRYX	READY Hold Time into 80C88A	20		ns	
TRYLCL	READY Inactive to CLK (Note 3)	−8		ns	
THVCH	HOLD Setup Time	20		ns	
TINVCH	INTR, NMI, $\overline{\text{TEST}}$ Setup Time (Note 2)	15		ns	
TILIH	Input Rise Time (Except CLK) (Note 4)		15	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK) (Note 4)		15	ns	From 2.0V to 0.8V

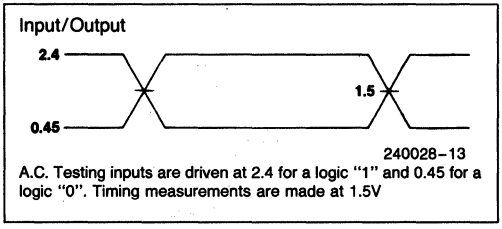
A.C. CHARACTERISTICS (Continued)

TIMING RESPONSES

Symbol	Parameter	80C88A-2		Units	Test Conditions
		Min	Max		
TCLAV	Address Valid Delay	10	60	ns	
TCLAX	Address Hold Time	10		ns	
TCLAZ	Address Float Delay	TCLAX	50	ns	
TLHLL	ALE Width	TCLCH-10		ns	
TCLLH	ALE Active Delay		50	ns	
TCHLL	ALE Inactive Delay		55	ns	
TLLAX	Address Hold Time to ALE Inactive	TCHCL-10		ns	
TCLDV	Data Valid Delay	10	60	ns	
TCHDX	Data Hold Time	10		ns	
TWHDX	Data Hold Time After $\overline{WR}$	TCLCH-30		ns	
TCVCTV	Control Active Delay 1	10	70	ns	
TCHCTV	Control Active Delay 2	10	60	ns	
TCVCTX	Control Inactive Delay	10	70	ns	
TAZRL	Address Float to READ Active	0		ns	
TCLRL	$\overline{RD}$ Active Delay	10	100	ns	
TCLRH	$\overline{RD}$ Inactive Delay	10	80	ns	
TRHAV	$\overline{RD}$ Inactive to Next Address Active	TCLCL-40		ns	
TCLHAV	HLDA Valid Delay	10	100	ns	
TRLRH	$\overline{RD}$ Width	2TCLCL-50		ns	
TWLWH	$\overline{WR}$ Width	2TCLCL-40		ns	
TAVAL	Address Valid to ALE Low	TCLCH-40		ns	
TOLOH	Output Rise Time (Note 4)		15	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time (Note 4)		15	ns	From 2.0V to 0.8V

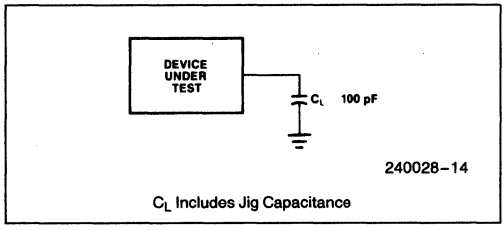
NOTES:
1. Signal at 82C84A shown for reference only. See 82C84A data sheet for the most recent specifications.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T2 state (8 ns into T3 state).
4. These parameters are characterized and not 100% tested.

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. Testing inputs are driven at 2.4 for a logic "1" and 0.45 for a logic "0". Timing measurements are made at 1.5V

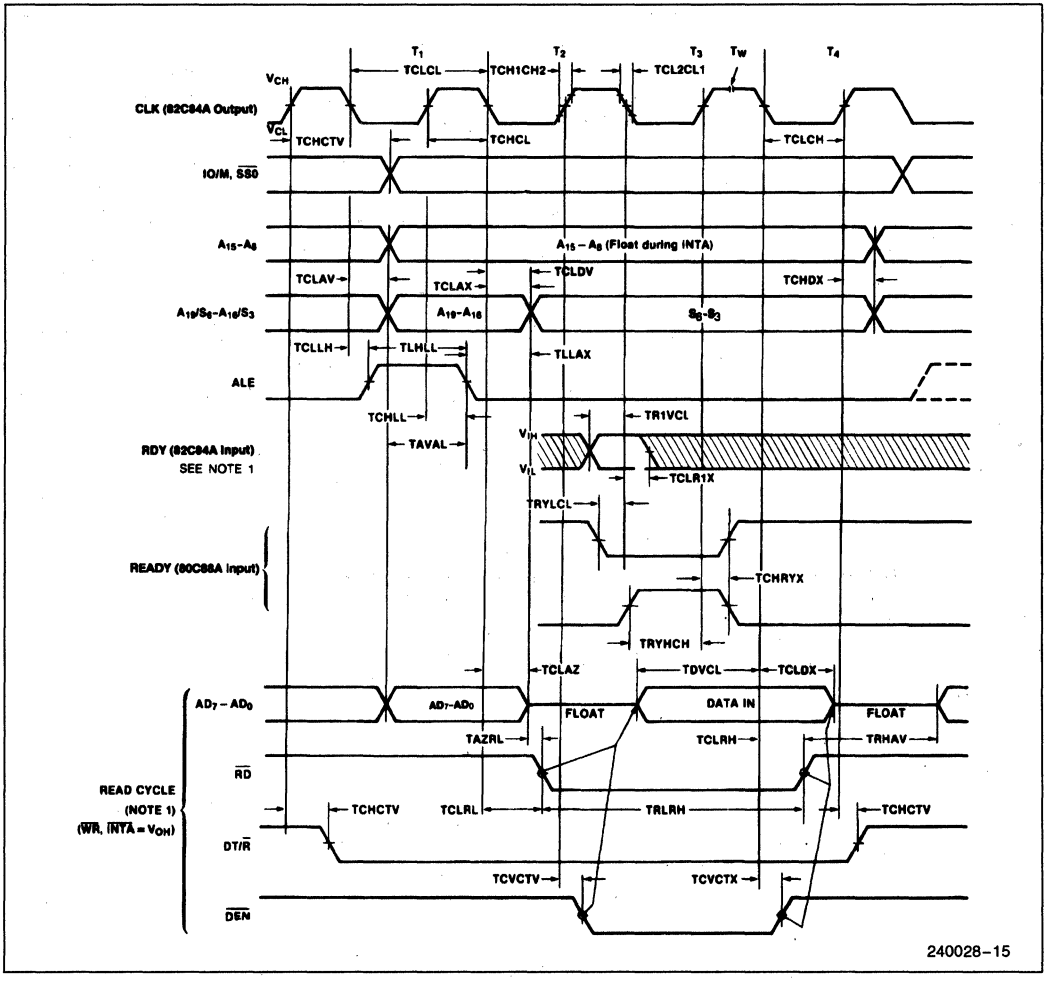
A.C. TESTING LOAD CIRCUIT



C_L Includes Jig Capacitance

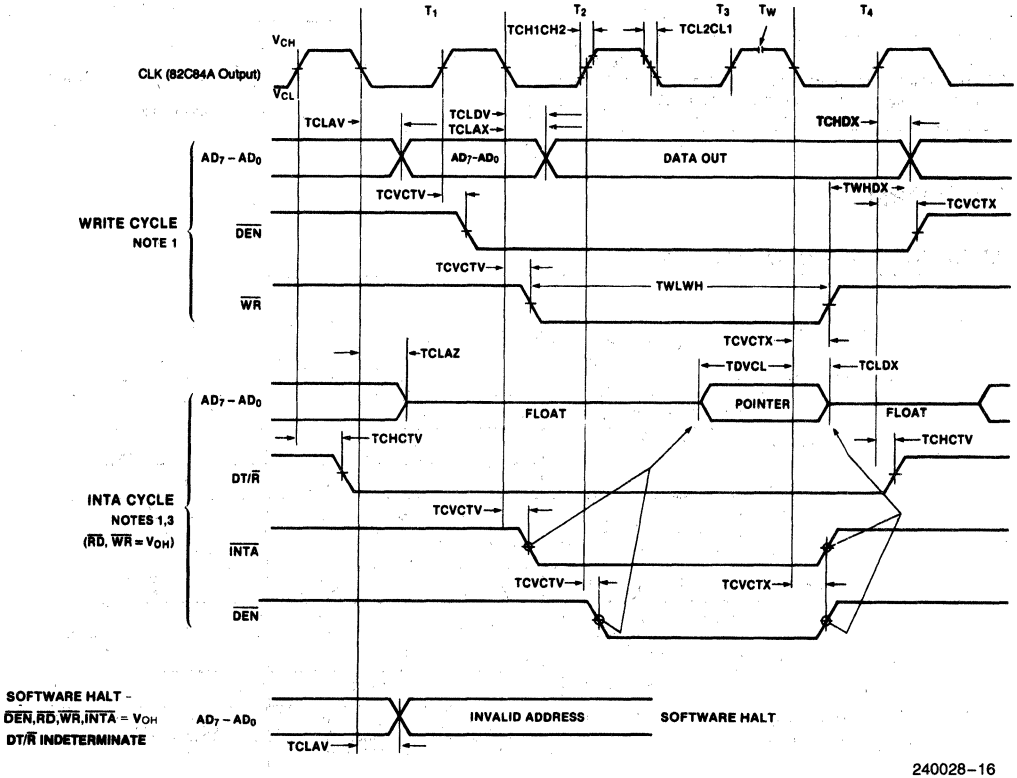
WAVEFORMS

BUS TIMING — MINIMUM MODE SYSTEM



WAVEFORMS (Continued)

BUS TIMING — MINIMUM MODE SYSTEM (Continued)



240028–16

NOTES:
1. All output timing measurements are made at 1.5V unless otherwise noted.
2. RDY is sampled near the end of T_2, T_3, T_W to determine if T_W machines states are to be inserted.
3. Two INTA Cycles run back-to-back. The 80C88A local ADDR/Data bus is floating during both INTA Cycles. Control signals are shown for the second INTA cycle.
4. Signals at 82C84A are shown for reference only.

A.C. CHARACTERISTICS

MAX MODE SYSTEM (USING 82C88 BUS CONTROLLER) TIMING REQUIREMENTS

Symbol	Parameter	80C88A-2		Units	Test Conditions
		Min	Max		
TCLCL	CLK Cycle Period	125	D.C.	ns	
TCLCH	CLK Low Time	68		ns	
TCHCL	CLK High Time	44		ns	
TCH1CH2	CLK Rise Time		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10	ns	From 3.5V to 1.0V
TDVCL	Data In Setup Time	20		ns	
TCLDX	Data In Hold Time	10		ns	
TR1VCL	RDY Setup Time into 82C84 (See Notes 1, 2)	35		ns	
TCLR1X	RDY Hold Time into 82C84 (See Notes 1, 2)	0		ns	
TRYHCH	READY Setup Time into 80C88A	68		ns	
TCHRYX	READY Hold Time into 80C88A	20		ns	
TRYLCL	READY Inactive to CLK (See Note 4)	−8		ns	
TINVCH	Setup Time for Recognition (INTR, NMI, TEST) (See Note 2)	15		ns	
TGVCH	RQ/GT Setup Time	15		ns	
TCHGX	RQ Hold Time into 80C88A	30		ns	
TILIH	Input Rise Time (Except CLK) (Note 5)		15	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK) (Note 5)		15	ns	From 2.0V to 0.8V

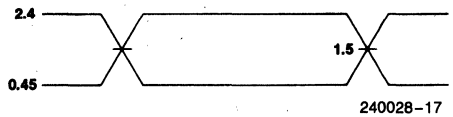
A.C. CHARACTERISTICS

TIMING RESPONSES

Symbol	Parameter	80C88A-2		Units	Test Conditions
		Min	Max		
TCLML	Command Active Delay (Note 1)	5	35	ns	
TCLMH	Command Inactive Delay (Note 1)	5	35	ns	
TRYHSH	READY Active to Status Passive (Note 3)		65	ns	
TCHSV	Status Active Delay	10	60	ns	
TCLSH	Status Inactive Delay	10	70	ns	
TCLAV	Address Valid Delay	10	60	ns	
TCLAX	Address Hold Time	10		ns	
TCLAZ	Address Float Delay	TCLAX	50	ns	
TSVLH	Status Valid to ALE High (Note 1)		20	ns	
TSVMCH	Status Valid to MCE High (Note 1)		30	ns	
TCLLH	CLK Low to ALE Valid (Note 1)		20	ns	
TCLMCH	CLK Low to MCE High (Note 1)		25	ns	
TCHLL	ALE Inactive Delay (Note 1)	4	18	ns	
TCLDV	Data Valid Delay	10	60	ns	
TCHDX	Data Hold Time	10		ns	
TCVNV	Control Active Delay (Note 1)	5	45	ns	
TCVNX	Control Inactive Delay (Note 1)	10	45	ns	
TAZRL	Address Float to Read Active	0		ns	
TCLRL	RD Active Delay	10	100	ns	
TCLRH	RD Inactive Delay	10	80	ns	
TRHAV	RD Inactive to Next Address Active	TCLCL-40		ns	
TCHDTL	Direction Control Active Delay (Note 1)		50	ns	
TCHDTH	Direction Control Inactive Delay (Note 1)		30	ns	
TCLGL	GT Active Delay	0	50	ns	
TCLGH	GT Inactive Delay	0	50	ns	
TRLRH	RD Width	2TCLCL-50		ns	
TOLOH	Output Rise Time (Note 5)		15	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time (Note 5)		15	ns	From 2.0V to 0.8V

NOTES:
1. Signal at 82C84A or 82C88 shown for reference only. See 82C84A and 82C88 data sheets for the most recent specifications.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T3 and wait states (8 ns into T3 state).
4. Applies only to T2 state (8 ns into T3 state).
5. These parameters are characterized and not 100% tested.

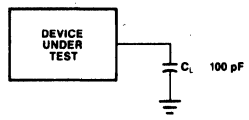
A.C. TESTING INPUT, OUTPUT WAVEFORM



240028-17

A.C. Testing inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0". Timing measurements are made at 1.5V.

A.C. TESTING LOAD CIRCUIT

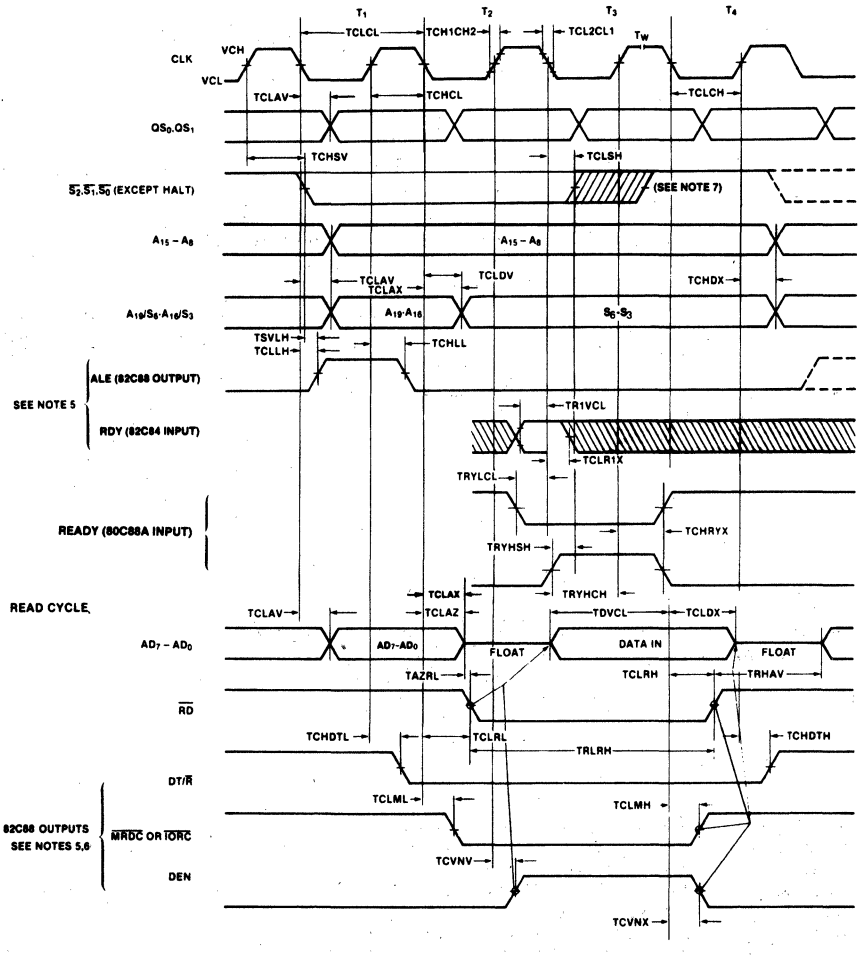


240028-18

C_L Includes Jig Capacitance

WAVEFORMS

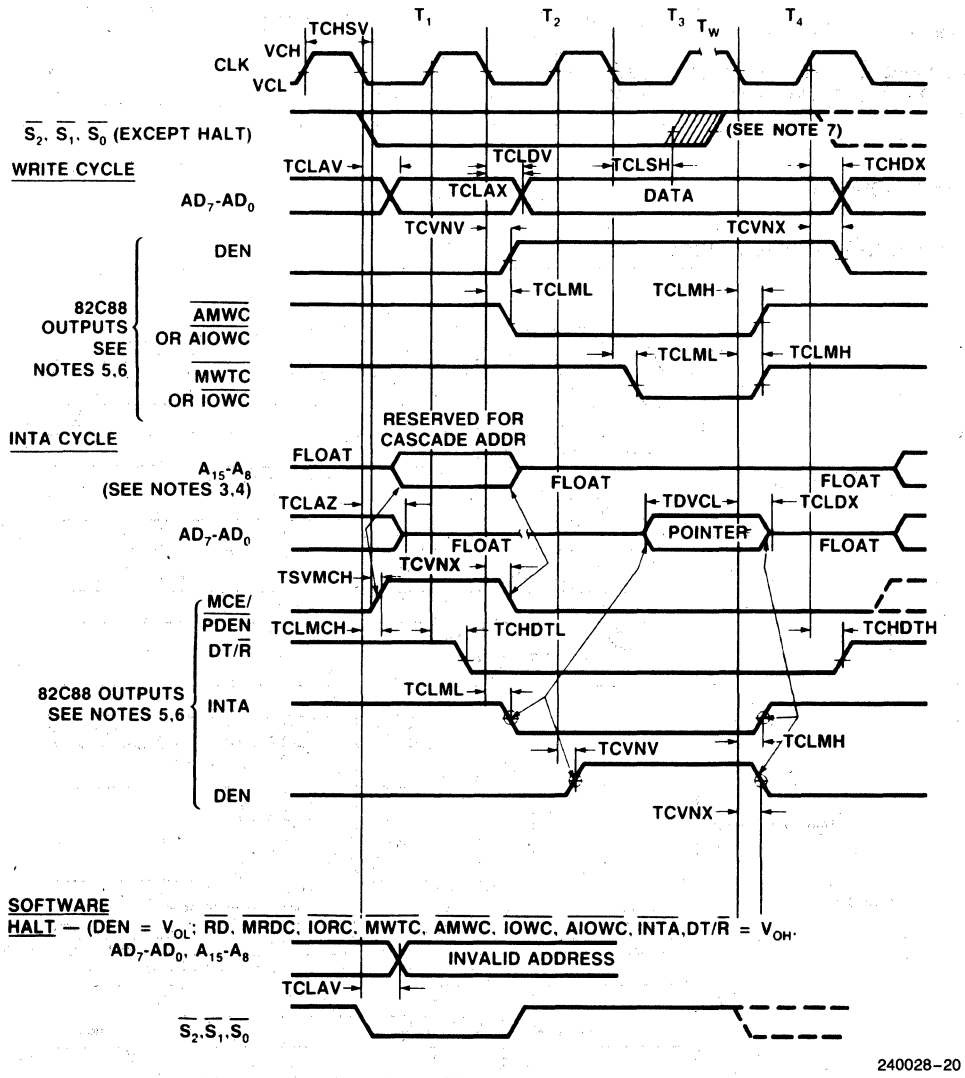
BUS TIMING—MAXIMUM MODE



240028-19

WAVEFORMS (Continued)

BUS TIMING — MAXIMUM MODE SYSTEM (USING 82C88)

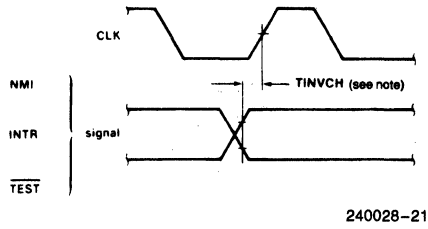


240028-20

NOTES:
1. All output timing measurements are made at 1.5V unless otherwise noted.
2. RDY is sampled near the end of T_2, T_3, T_W to determine if T_W machines states are to be inserted.
3. Cascade address is valid between first and second INTA cycles.
4. Two INTA cycles run back-to-back. The 80C88A local ADDR/Data bus is floating during both INTA cycles. Control for pointer address is shown for second INTA cycle.
5. Signals at 82C84A or 82C88 are shown for reference only.
6. The issuance of the 82C88 command and control signals ($\overline{MRDC}$, $\overline{MWTC}$, $\overline{AMWC}$, $\overline{IORC}$, $\overline{IOWC}$, $\overline{AIOWC}$, $\overline{INTA}$ and DEN) lags the active high 82C88 CEN.
7. Status inactive in state just prior to T_4.

WAVEFORMS (Continued)

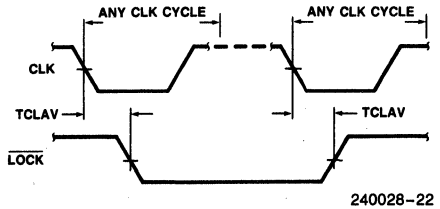
ASYNCHRONOUS SIGNAL RECOGNITION



240028-21

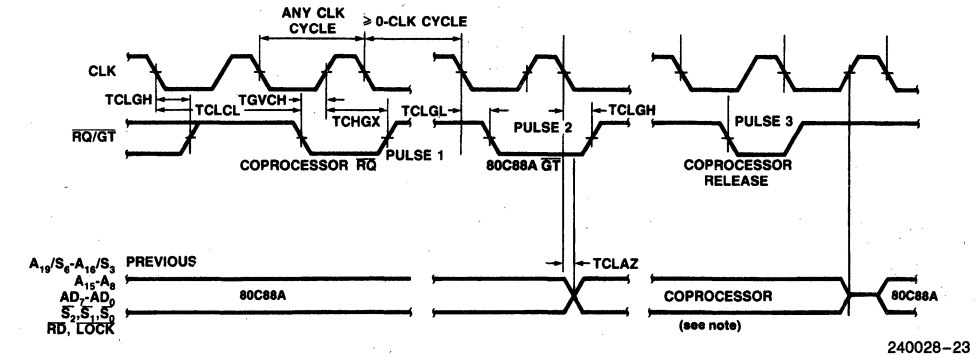
NOTE: Setup requirements for asynchronous signals only to guarantee recognition at next CLK.

BUS LOCK SIGNAL TIMING (MAXIMUM MODE ONLY)



240028-22

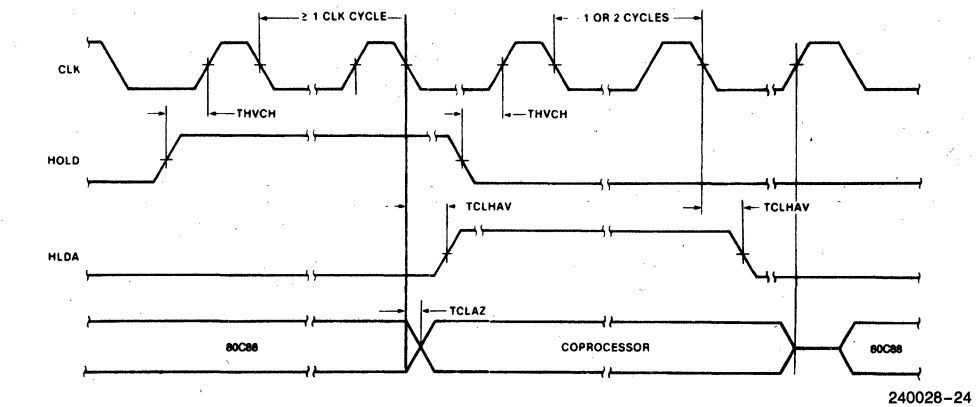
REQUEST/GRANT SEQUENCE TIMING (MAXIMUM MODE ONLY)



240028-23

NOTE: The coprocessor may not drive the busses outside the region shown without risking contention.

HOLD/HOLD ACKNOWLEDGE TIMING (MINIMUM MODE ONLY)



240028-24

80C86A/80C88A INSTRUCTION SET SUMMARY

Mnemonic and Description	Instruction Code			
DATA TRANSFER				
MOV = Move:	76543210	76543210	76543210	76543210
Register/Memory to/from Register	100010dw	mod reg r/m		
Immediate to Register/Memory	1100011w	mod 000 r/m	data	data if w 1
Immediate to Register	1011w reg	data	data if w 1	
Memory to Accumulator	1010000w	add-low	addr-high	
Accumulator to Memory	1010001w	addr-low	addr-high	
Register/Memory to Segment Register**	10001110	mod 0 reg r/m		
Segment Register to Register/Memory	10001100	mod 0 reg r/m		
PUSH = Push:				
Register/Memory	11111111	mod 110 r/m		
Register	01010 reg			
Segment Register	000 reg 110			
POP = Pop:				
Register/Memory	10001111	mod 000 r/m		
Register	01011 reg			
Segment Register	000 reg 111			
XCHG = Exchange:				
Register/Memory with Register	1000011w	mod reg r/m		
Register with Accumulator	10010 reg			
IN = Input from:				
Fixed Port	1110010w	port		
Variable Port	1110110w			
OUT = Output to:				
Fixed Port	1110011w	port		
Variable Port	1110111w			
XLAT = Translate Byte to AL	11010111			
LEA = Load EA to Register	10001101	mod reg r/m		
LDS = Load Pointer to DS	11000101	mod reg r/m		
LES = Load Pointer to ES	11000100	mod reg r/m		
LAHF = Load AH with Flags	10011111			
SAHF = Store AH into Flags	10011110			
PUSHF = Push Flags	10011100			
POPF = Pop Flags	10011101			

80C86A/80C88A INSTRUCTION SET SUMMARY (Continued)

Mnemonic and Description	Instruction Code			
ARITHMETIC	76543210	76543210	76543210	76543210
ADD = Add:				
Reg./Memory with Register to Either	000000dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod 000 r/m	data	data if s:w = 01
Immediate to Accumulator	0000010w	data	data if w = 1	
ADC = Add with Carry:				
Reg./Memory with Register to Either	000100dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod 010 r/m	data	data if s:w = 01
Immediate to Accumulator	0001010w	data	data if w = 1	
INC = Increment:				
Register/Memory	1111111w	mod 000 r/m		
Register	01000reg			
AAA = ASCII Adjust for Add	00110111			
DAA = Decimal Adjust for Add	00100111			
SUB = Subtract:				
Reg./Memory and Register to Either	001010dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod 101 r/m	data	data if s:w = 01
Immediate from Accumulator	0010110w	data	data if w = 1	
SBB = Subtract with Borrow				
Reg./Memory and Register to Either	000110dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod 011 r/m	data	data if s:w = 01
Immediate from Accumulator	0001110w	data	data if w = 1	
DEC = Decrement:				
Register/Memory	1111111w	mod 001 r/m		
Register	01001 reg			
NEG = Change Sign	1111011w	mod 011 r/m		
CMP = Compare:				
Register/Memory and Register	001110dw	mod reg r/m		
Immediate with Register/Memory	100000sw	mod 111 r/m	data	data if s:w = 01
Immediate with Accumulator	0011110w	data	data if w = 1	
AAS = ASCII Adjust for Subtract	00111111			
DAS = Decimal Adjust for Subtract	00101111			
MUL = Multiply (Unsigned)	1111011w	mod 100 r/m		
IMUL = Integer Multiply (Signed)	1111011w	mod 101 r/m		
AAM = ASCII Adjust for Multiply	11010100	00001010		
DIV = Divide (Unsigned)	1111011w	mod 110 r/m		
IDIV = Integer Divide (Signed)	1111011w	mod 111 r/m		
AAD = ASCII Adjust for Divide	11010101	00001010		
CBW = Convert Byte to Word	10011000			
CWD = Convert Word to Double Word	10011001			

80C86A/80C88A INSTRUCTION SET SUMMARY (Continued)

Mnemonic and Description	Instruction Code			
LOGIC	76543210	76543210	76543210	76543210
NOT = Invert	1111011w	mod 010 r/m		
SHL/SAL = Shift Logical/Arithmetic Left	110100vw	mod 100 r/m		
SHR = Shift Logical Right	110100vw	mod 101 r/m		
SAR = Shift Arithmetic Right	110100vw	mod 111 r/m		
ROL = Rotate Left	110100vw	mod 000 r/m		
ROR = Rotate Right	110100vw	mod 001 r/m		
RCL = Rotate Through Carry Flag Left	110100vw	mod 010 r/m		
RCR = Rotate Through Carry Right	110100vw	mod 011 r/m		
AND = **And:**				
Reg./Memory and Register to Either	001000dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 100 r/m	data	data if w = 1
Immediate to Accumulator	0010010w	data	data if w = 1	
TEST = **And Function to Flags, No Result:**				
Register/Memory and Register	1000010w	mod reg r/m		
Immediate Data and Register/Memory	1111011w	mod 000 r/m	data	data if w = 1
Immediate Data and Accumulator	1010100w	data	data if w = 1	
OR = **Or:**				
Reg./Memory and Register to Either	000010dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 001 r/m	data	data if w = 1
Immediate to Accumulator	0000110w	data	data if w = 1	
XOR = **Exclusive or:**				
Reg./Memory and Register to Either	001100dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 110 r/m	data	data if w = 1
Immediate to Accumulator	0011010w	data	data if w = 1	
STRING MANIPULATION				
REP = Repeat	1111001z			
MOVS = Move Byte/Word	1010010w			
CMPS = Compare Byte/Word	1010011w			
SCAS = Scan Byte/Word	1010111w			
LODS = Load Byte/Wd to AL/AX	1010110w			
STOS = Stor Byte/Wd from AL/A	1010101w			
CONTROL TRANSFER				
CALL = **Call:**				
Direct Within Segment	11101000	disp-low	disp-high	
Indirect Within Segment	11111111	mod 010 r/m		
Direct Intersegment	10011010	offset-low	offset-high	
		seg-low	seg-high	
Indirect Intersegment	11111111	mod 011 r/m		

80C86A/80C88A INSTRUCTION SET SUMMARY (Continued)

Mnemonic and Description	Instruction Code		
JMP = Unconditional Jump:	76543210	76543210	76543210
Direct Within Segment	11101001	disp-low	disp-high
Direct Within Segment-Short	11101011	disp	
Indirect Within Segment	11111111	mod 100 r/m	
Direct Intersegment	11101010	offset-low	offset-high
		seg-low	seg-high
Indirect Intersegment	11111111	mod 101 r/m	
RET = Return from CALL:			
Within Segment	11000011		
Within Seg Adding Immed to SP	11000010	data-low	data-high
Intersegment	11001011		
Intersegment Adding Immediate to SP	11001010	data-low	data-high
JE/JZ = Jump on Equal/Zero	01110100	disp	
JL/JNGE = Jump on Less/Not Greater or Equal	01111100	disp	
JLE/JNG = Jump on Less or Equal/ Not Greater	01111110	disp	
JB/JNAE = Jump on Below/Not Above or Equal	01110010	disp	
JBE/JNA = Jump on Below or Equal/ Not Above	01110110	disp	
JP/JPE = Jump on Parity/Parity Even	01111010	disp	
JO = Jump on Overflow	01110000	disp	
JS = Jump on Sign	01111000	disp	
JNE/JNZ = Jump on Not Equal/Not Zero	01110101	disp	
JNL/JGE = Jump on Not Less/Greater or Equal	01111101	disp	
JNLE/JG = Jump on Not Less or Equal/ Greater	01111111	disp	
JNB/JAE = Jump on Not Below/Above or Equal	01110011	disp	
JNBE/JA = Jump on Not Below or Equal/Above	01110111	disp	
JNP/JPO = Jump on Not Par/Par Odd	01111011	disp	
JNO = Jump on Not Overflow	01110001	disp	
JNS = Jump on Not Sign	01111001	disp	
LOOP = Loop CX Times	11100010	disp	
LOOPZ/LOOPE = Loop While Zero/Equal	11100001	disp	
LOOPNZ/LOOPNE = Loop While Not Zero/Equal	11100000	disp	
JCXZ = Jump on CX Zero	11100011	disp	
INT = Interrupt			
Type Specified	11001101	type	
Type 3	11001100		
INTO = Interrupt on Overflow	11001110		
IRET = Interrupt Return	11001111		

80C86A/80C88A INSTRUCTION SET SUMMARY (Continued)

Mnemonic and Description	Instruction Code	
	76543210	76543210
PROCESSOR CONTROL		
CLC = Clear Carry	11111000	
CMC = Complement Carry	11110101	
STC = Set Carry	11111001	
CLD = Clear Direction	11111100	
STD = Set Direction	11111101	
CLI = Clear Interrupt	11111010	
STI = Set Interrupt	11111011	
HLT = Halt	11110100	
WAIT = Wait	10011011	
ESC = Escape (to External Device)	11011xxx	mod x x x r/m
LOCK = Bus Lock Prefix	11110000	

NOTES:
AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value.
Greater = more positive;
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
if w = 1 then word instruction; if w = 0 then byte instruction
if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16 bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low
if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)
*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.
**MOV CS, REG/MEMORY not allowed.
if s:w = 01 then 16 bits of immediate data form the operand.
if s:w = 11 then an immediate data byte is sign extended to form the 16-bit operand.
if v = 0 then "count" = 1; if v = 1 then "count" in (CL)
x = don't care
z is used for string primitives for comparison with ZF FLAG.

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:
FLAGS =
X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

DATA SHEET REVISION REVIEW

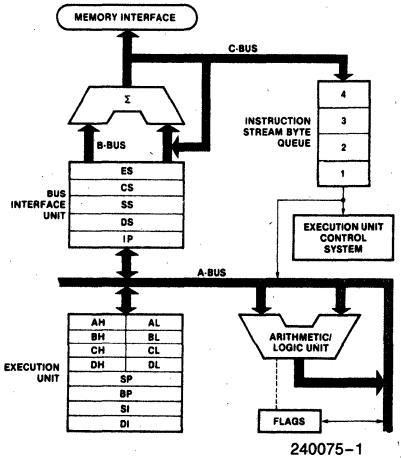
The following list represents key differences between this and the -001 data sheet. Please review this summary carefully.

1. In the Pin Description Table (Table 1), the description of the HLDA signal being issued has been corrected. HLDA will be issued in the middle of either the T4 or Ti state.

80C88AL
8-BIT CHMOS MICROPROCESSOR

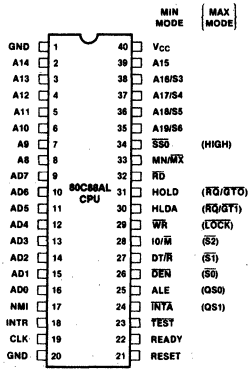
- **Pin-for-Pin and Functionally Compatible to Industry Standard HMOS 8088**
- **Direct Software Compatibility with 80C86AL, 8086, 8088**
- **Fully Static Design with Frequency Range from D.C. to:**
  — 5 MHz for 80C88AL
  — 8 MHz for 80C88AL-2
- **Low Power Operation**
  — Operating I_{CC} = 10 mA/MHz
  — Standby I_{CCS} = 750 μA max
- **Bus-Hold Circuitry Eliminates Pull-Up Resistors**
- **Direct Addressing Capability of 1 MByte of Memory**
- **Architecture Designed for Powerful Assembly Language and Efficient High Level Languages**
- **24 Operand Addressing Modes**
- **Byte, Word and Block Operations**
- **8 and 16-Bit Signed and Unsigned Arithmetic**
  — Binary or Decimal
  — Multiply and Divide
- **Available in 40-Lead Plastic DIP and 44-Lead PLCC Packages**
  (See Packaging Spec., Order #231369)

The Intel 80C88AL is a high performance, CHMOS version of the industry standard HMOS 8088 8-bit CPU. The processor has attributes of both 8 and 16-bit microprocessors. It is available in 5 and 8 MHz clock rates. The 80C88AL offers two modes of operation: MINimum for small systems and MAXimum for larger applications such as multi-processing. It is available in 40-pin DIP and 44-pin plastic leaded chip carrier (PLCC) package.



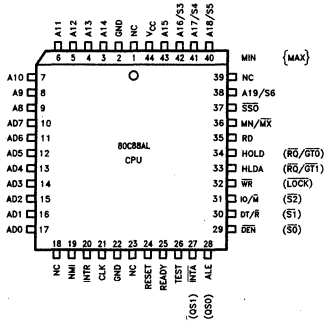
240075-1

Figure 1. 80C88AL CPU Functional Block Diagram



240075-2

Figure 2a. 80C88AL 40-Lead DIP Configuration



240075-3

Figure 2b. 80C88AL 44-Lead PLCC Configuration

September 1988
Order Number: 240075-002

Table 1. Pin Description

The following pin function descriptions are for 80C88AL systems in either minimum or maximum mode. The "local bus" in these descriptions is the direct multiplexed bus interface connection to the 80C88AL (without regard to additional bus buffers).

Symbol	P-DIP Config. Pin No.	Type	Name and Function
AD7–AD0	9–16	I/O	**ADDRESS DATA BUS:** These lines constitute the time multiplexed memory/IO address (T1) and data (T2, T3, Tw, and T4) bus. These lines are active HIGH and float to 3-state OFF(1) during interrupt acknowledge and local bus "hold acknowledge".
A15–A8	2–8, 39	O	**ADDRESS BUS:** These lines provide address bits 8 through 15 for the entire bus cycle (T1–T4). These lines do not have to be latched by ALE to remain valid. A15–A8 are active HIGH and float to 3-state OFF(1) during interrupt acknowledge and local bus "hold acknowledge".
A19/S6, A18/S5, A17/S4, A16/S3	35–38	O	**ADDRESS/STATUS:** During T1, these are the four most significant address lines for memory operations. During I/O operations, these lines are LOW. During memory and I/O operations, status information is available on these lines during T2, T3, Tw, and T4. S6 is always low. The status of the interrupt enable flag bit (S5) is updated at the beginning of each clock cycle. S4 and S3 are encoded as shown. This information indicates which segment register is presently being used for data accessing. These lines float to 3-state OFF(1) during local bus "hold acknowledge".

S4	S3	CHARACTERISTICS
0(LOW)	0	Alternate Data
0	1	Stack
1(HIGH)	0	Code or None
1	1	Data
S6 is 0(LOW)		

$\overline{RD}$	32	O	**READ:** Read strobe indicates that the processor is performing a memory or I/O read cycle, depending on the state of the IO/$\overline{M}$ pin or S2. This signal is used to read devices which reside on the 80C88AL local bus. $\overline{RD}$ is active LOW during T2, T3 and Tw of any read cycle, and is guaranteed to remain HIGH in T2 until the 80C88AL local bus has floated. This signal floats to 3-state OFF(1) in "hold acknowledge".
READY	22	I	**READY:** is the acknowledgement from the addressed memory or I/O device that it will complete the data transfer. The RDY signal from memory or I/O is synchronized by the 82C84A clock generator to form READY. This signal is active HIGH. The 80C88AL READY input is not synchronized. Correct operation is not guaranteed if the set up and hold times are not met.

Table 1. Pin Description (Continued)

Symbol	P-DIP Config. Pin No.	Type	Name and Function
INTR	18	I	**INTERRUPT REQUEST:** is a level triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored to via an interrupt vector lookup table located in system memory. It can be internally masked by software resetting the interrupt enable bit. INTR is internally synchronized. This signal is active HIGH.
$\overline{TEST}$	23	I	**$\overline{TEST}$:** input is examined by the "wait for test" instruction. If the $\overline{TEST}$ input is LOW, execution continues, otherwise the processor waits in an "idle" state. This input is synchronized internally during each clock cycle on the leading edge of CLK.
NMI	17	I	**NON-MASKABLE INTERRUPT:** is an edge triggered input which causes a type 2 interrupt. A subroutine is vectored to via an interrupt vector lookup table located in system memory. NMI is not maskable internally by software. A transition from a LOW to HIGH initiates the interrupt at the end of the current instruction. This input is internally synchronized.
RESET	21	I	**RESET:** causes the processor to immediately terminate its present activity. The signal must be active HIGH for at least four clock cycles. It restarts execution, as described in the instruction set description, when RESET returns LOW. RESET is internally synchronized.
CLK	19	I	**CLOCK:** provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing.
V_{CC}	40		**V_{CC}:** is the +5V ±10% power supply pin.
GND	1, 20		**GND:** are the ground pins. Both must be connected.
MN/$\overline{MX}$	33	I	**MINIMUM/MAXIMUM:** indicates what mode the processor is to operate in. The two modes are discussed in the following sections.

The following pin function descriptions are for the 80C88AL minimum mode (i.e., MN/$\overline{MX}$ = V_{CC}). Only the pin functions which are unique to minimum mode are described; all other pin functions are as described above.

IO/$\overline{M}$	28	O	**STATUS LINE:** is an inverted maximum mode $\overline{S2}$. It is used to distinguish a memory access from an I/O access. IO/$\overline{M}$ becomes valid in the T4 preceding a bus cycle and remains valid until the final T4 of the cycle (I/O = HIGH, M = LOW). IO/$\overline{M}$ floats to 3-state OFF(1) in local bus "hold acknowledge".
$\overline{WR}$	29	O	**WRITE:** strobe indicates that the processor is performing a write memory or write I/O cycle, depending on the state of the IO/$\overline{M}$ signal. WR is active for T2, T3, and Tw of any write cycle. It is active LOW, and floats to 3-state OFF(1) in local bus "hold acknowledge".
$\overline{INTA}$	24	O	**INTA:** is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T2, T3, and Tw of each interrupt acknowledge cycle.

Table 1. Pin Description (Continued)

Symbol	P-DIP Config. Pin No.	Type	Name and Function
ALE	25	O	**ADDRESS LATCH ENABLE:** is provided by the processor to latch the address into an address latch. It is a HIGH pulse active during clock low of T1 of any bus cycle. Note that ALE is never floated.
DT/$\overline{R}$	27	O	**DATA TRANSMIT/RECEIVE:** is needed in a minimum system that desires to use a data bus transceiver. It is used to control the direction of data flow through the transceiver. Logically, DT/$\overline{R}$ is equivalent to $\overline{S1}$ in the maximum mode, and its timing is the same as for IO/$\overline{M}$ (T = HIGH, R = LOW). This signal floats to 3-state OFF(1) in local "hold acknowledge".
$\overline{DEN}$	26	O	**DATA ENABLE:** is provided as an output enable for the transceiver in a minimum system which uses the transceiver. $\overline{DEN}$ is active LOW during each memory and I/O access, and for $\overline{INTA}$ cycles. For a read or $\overline{INTA}$ cycle, it is active from the middle of T2 until the middle of T4, while for a write cycle, it is active from the beginning of T2 until the middle of T4. $\overline{DEN}$ floats to 3-state OFF(1) during local bus "hold acknowledge".
HOLD, HLDA	30, 31	I, O	**HOLD:** indicates that another master is requesting a local bus "hold". To be acknowledged, HOLD must be active HIGH. The processor receiving the "hold" request will issue HLDA (HIGH) as an acknowledgement, in the middle of a T4 or Ti clock cycle. Simultaneous with the issuance of HLDA the processor will float the local bus and control lines. After HOLD is detected as being LOW, the processor lowers HLDA, and when the processor needs to run another cycle, it will again drive the local bus and control lines. Hold is not an asynchronous input. External synchronization should be provided if the system cannot otherwise guarantee the set up time.
$\overline{SS0}$	34	O	**STATUS LINE:** is logically equivalent to $\overline{S0}$ in the maximum mode. The combination of $\overline{SS0}$, IO/$\overline{M}$ and DT/$\overline{R}$ allows the system to completely decode the current bus cycle status.

IO/$\overline{M}$	DT/$\overline{R}$	$\overline{SS0}$	CHARACTERISTICS
1(HIGH)	0	0	Interrupt Acknowledge
1	0	1	Read I/O port
1	1	0	Write I/O port
1	1	1	Halt
0(LOW)	0	0	Code access
0	0	1	Read memory
0	1	0	Write memory
0	1	1	Passive

Table 1. Pin Description (Continued)

The following pin function descriptions are for the 80C88AL/82C88 system in maximum mode (i.e., MN/$\overline{MX}$ = GND.) Only the pin functions which are unique to maximum mode are described; all other pin functions are as described above.

<table>
<tr><th>Symbol</th><th>P-DIP Config. Pin No.</th><th>Type</th><th>Name and Function</th></tr>
<tr><td>$\overline{S2}$, $\overline{S1}$, $\overline{S0}$</td><td>26–28</td><td>O</td><td>STATUS: is active during clock high of T4, T1, and T2, and is returned to the passive state (1,1,1) during T3 or during Tw when READY is HIGH. This status is used by the 82C88 bus controller to generate all memory and I/O access control signals. Any change by $\overline{S2}$, $\overline{S1}$, or $\overline{S0}$ during T4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T3 or Tw is used to indicate the end of a bus cycle.

These signals float to 3-state OFF(1) during "hold acknowledge". During the first clock cycle after RESET becomes active, these signals are active HIGH. After this first clock, they float to 3-state OFF.
<table>
<tr><th>$\overline{S2}$</th><th>$\overline{S1}$</th><th>$\overline{S0}$</th><th>CHARACTERISTICS</th></tr>
<tr><td>0(LOW)</td><td>0</td><td>0</td><td>Interrupt Acknowledge</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Read I/O port</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Write I/O port</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Halt</td></tr>
<tr><td>1(HIGH)</td><td>0</td><td>0</td><td>Code access</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Read memory</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Write memory</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Passive</td></tr>
</table>
</td></tr>
<tr><td>$\overline{RQ}$/$\overline{GT0}$,
$\overline{RQ}$/$\overline{GT1}$</td><td>30, 31</td><td>I/O</td><td>REQUEST/GRANT: pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional with $\overline{RQ}$/$\overline{GT0}$ having higher priority than $\overline{RQ}$/$\overline{GT1}$. $\overline{RQ}$/$\overline{GT}$ has an internal pull-up resistor, so may be left unconnected. The request/grant sequence is as follows (see timing diagram):

1. A pulse of one CLK wide from another local bus master indicates a local bus request ("hold") to the 80C88AL (pulse 1).

2. During a T4 or T1 clock cycle, a pulse one clock wide from the 80C88AL to the requesting master (pulse 2), indicates that the 80C88AL has allowed the local bus to float and that it will enter the "hold acknowledge" state at the next CLK. The CPU's bus interface unit is disconnected logically from the local bus during "hold acknowledge". The same rules as for HOLD/HOLDA apply as for when the bus is released.

3. A pulse one CLK wide from the requesting master indicates to the 80C88AL (pulse 3) that the "hold" request is about to end and that the 80C88AL can reclaim the local bus at the next CLK. The CPU then enters T4.</td></tr>
</table>

Table 1. Pin Description (Continued)

Symbol	P-DIP Config. Pin No.	Type	Name and Function
$\overline{RQ}/\overline{GT0}$, $\overline{RQ}/\overline{GT1}$	30, 31	I/O	Each master-master exchange of the local bus is a sequence of three pulses. There must be one idle CLK cycle after each bus exchange. Pulses are active LOW. If the request is made while the CPU is performing a memory cycle, it will release the local bus during T4 of the cycle when all the following conditions are met: 1. Request occurs on or before T2. 2. Current cycle is not the low bit of a word. 3. Current cycle is not the first acknowledge of an interrupt acknowledge sequence. 4. A locked instruction is not currently executing. If the local bus is idle when the request is made the two possible events will follow: 1. Local bus will be released during the next clock. 2. A memory cycle will start within 3 clocks. Now the four rules for a currently active memory cycle apply with condition number 1 already satisfied.
$\overline{LOCK}$	29	O	**$\overline{LOCK}$:** indicates that other system bus masters are not to gain control of the system bus while $\overline{LOCK}$ is active (LOW). The $\overline{LOCK}$ signal is activated by the "LOCK" prefix instruction and remains active until the completion of the next instruction. This signal is active LOW, and floats to 3-state OFF(1) in "hold acknowledge".
QS1, QS0	24, 25	O	**QUEUE STATUS:** provide status to allow external tracking of the internal 80C88AL instruction queue. The queue status is valid during the CLK cycle after which the queue operation is performed.

QS1	QS0	CHARACTERISTICS
0(LOW)	0	No operation
0	1	First byte of opcode from queue
1(HIGH)	0	Empty the queue
1	1	Subsequent byte from queue

—	34	O	Pin 34 is always high in the maximum mode.

NOTE:
1. See the section on Bus Hold Circuitry.

FUNCTIONAL DESCRIPTION

STATIC OPERATION

All 80C88AL circuitry is of static design. Internal registers, counters and latches are static and require no refresh as with dynamic circuit design. This eliminates the minimum operating frequency restriction placed on other microprocessors. The CMOS 80C88AL can operate from DC to the appropriate upper frequency limit. The processor clock may be stopped in either state (high/low) and held there indefinitely. This type of operation is especially useful for system debug or power critical applications.

The 80C88AL can be single stepped using only the CPU clock. This state can be maintained as long as is necessary. Single step clock operation allows simple interface circuitry to provide critical information for bringing up your system.

Static design also allows very low frequency operation. In a power critical situation, this can provide extremely low power operation since 80C88AL power dissipation is directly related to operating frequency. As the system frequency is reduced, so is the operating power until ultimately, at a DC input frequency, the 80C88AL power requirement is the standby current.

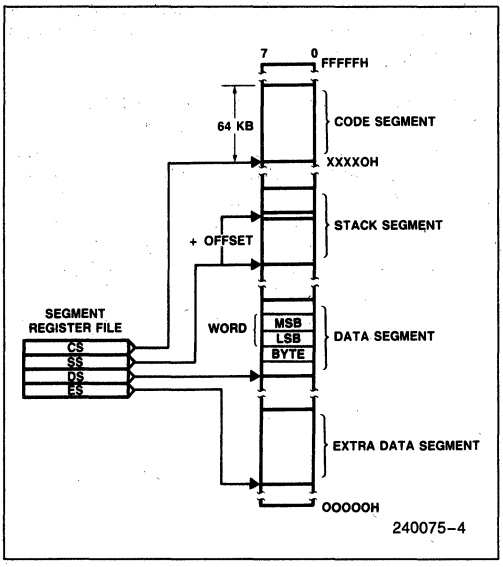


Figure 3. Memory Organization

MEMORY ORGANIZATION

The processor provides a 20-bit address to memory which locates the byte being referenced. The memory is organized as a linear array of up to 1 million bytes, addressed as 00000(H) to FFFFF(H). The memory is logically divided into code, data, extra data, and stack segments of up to 64K bytes each, with each segment falling on 16-byte boundaries. (See Figure 3.)

All memory references are made relative to base addresses contained in high speed segment registers. The segment types were chosen based on the addressing needs of programs. The segment register to be selected is automatically chosen according to the rules of the following table. All information in one segment type share the same logical attributes (e.g. code or data). By structuring memory into relocatable areas of similar characteristics and by automatically selecting segment registers, programs are shorter, faster, and more structured.

Word (16-bit) operands can be located on even or odd address boundaries. For address and data operands, the least significant byte of the word is stored in the lower valued address location and the most significant byte in the next higher address location. The BIU will automatically execute two fetch or write cycles for 16-bit operands.

Certain locations in memory are reserved for specific CPU operations. (See Figure 4.) Locations from addresses FFFF0H through FFFFFH are reserved for operations including a jump to the initial system

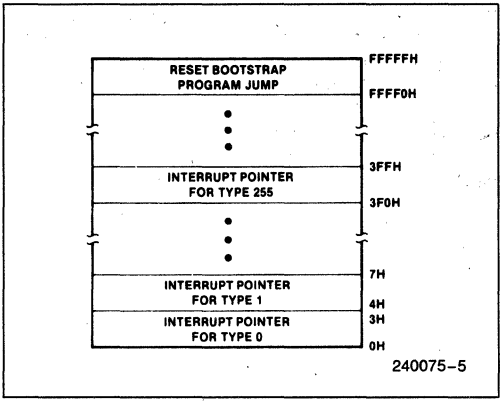


Figure 4. Reserved Memory Locations

Memory Reference Need	Segment Register Used	Segment Selection Rule
Instructions	CODE (CS)	Automatic with all instruction prefetch.
Stack	STACK (SS)	All stack pushes and pops. Memory references relative to BP base register except data references.
Local Data	DATA (DS)	Data references when: relative to stack, destination of string operation, or explicitly overridden.
External (Global) Data	EXTRA (ES)	Destination of string operations: Explicitly selected using a segment override.

initialization routine. Following RESET, the CPU will always begin execution at location FFFF0H where the jump must be located. Locations 00000H through 003FFH are reserved for interrupt operations. Four-byte pointers consisting of a 16-bit segment address and a 16-bit offset address direct program flow to one of the 256 possible interrupt service routines. The pointer elements are assumed to have been stored at their respective places in reserved memory prior to the occurrence of interrupts.

MINIMUM AND MAXIMUM MODES

The requirements for supporting minimum and maximum 80C88AL systems are sufficiently different that they cannot be done efficiently with 40 uniquely defined pins. Consequently, the 80C88AL is equipped with a strap pin (MN/$\overline{MX}$) which defines the system configuration. The definition of a certain subset of the pins changes, dependent on the condition of the strap pin. When the MN/$\overline{MX}$ pin is strapped to GND, the 80C88AL defines pins 24 through 31 and 34 in maximum mode. When the MN/$\overline{MX}$ pin is strapped to V_{CC}, the 80C88AL generates bus control signals itself on pins 24 through 31 and 34.

The minimum mode 80C88AL can be used with either a multiplexed or demultiplexed bus. The multiplexed bus configuration is compatible with the MCS®-85 multiplexed bus peripherals (8155, 8156, 8355, 8755A, and 8185). This configuration (See Figure 5) provides the user with a minimum chip count system. This architecture provides the 80C88AL processing power in a highly integrated form.

The demultiplexed mode requires one latch (for 64k addressability) or two latches (for a full megabyte of addressing). A third latch can be used for buffering if the address bus loading requires it. A transceiver can also be used if data bus buffering is required. (See Figure 6.) The 80C88AL provides $\overline{DEN}$ and DT/$\overline{R}$ to control the transceiver, and ALE to latch the addresses. This configuration of the minimum mode provides the standard demultiplexed bus structure with heavy bus buffering and relaxed bus timing requirements.

The maximum mode employs the 82C88 bus controller. (See Figure 7.) The 82C88 decodes status lines $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$, and provides the system with all bus control signals. Moving the bus control to the 82C88 provides better source and sink current capability to the control lines, and frees the 80C88AL pins for extended large system features. Hardware lock, queue status, and two request/grant interfaces are provided by the 80C88AL in maximum mode. These features allow co-processors in local bus and remote bus configurations.

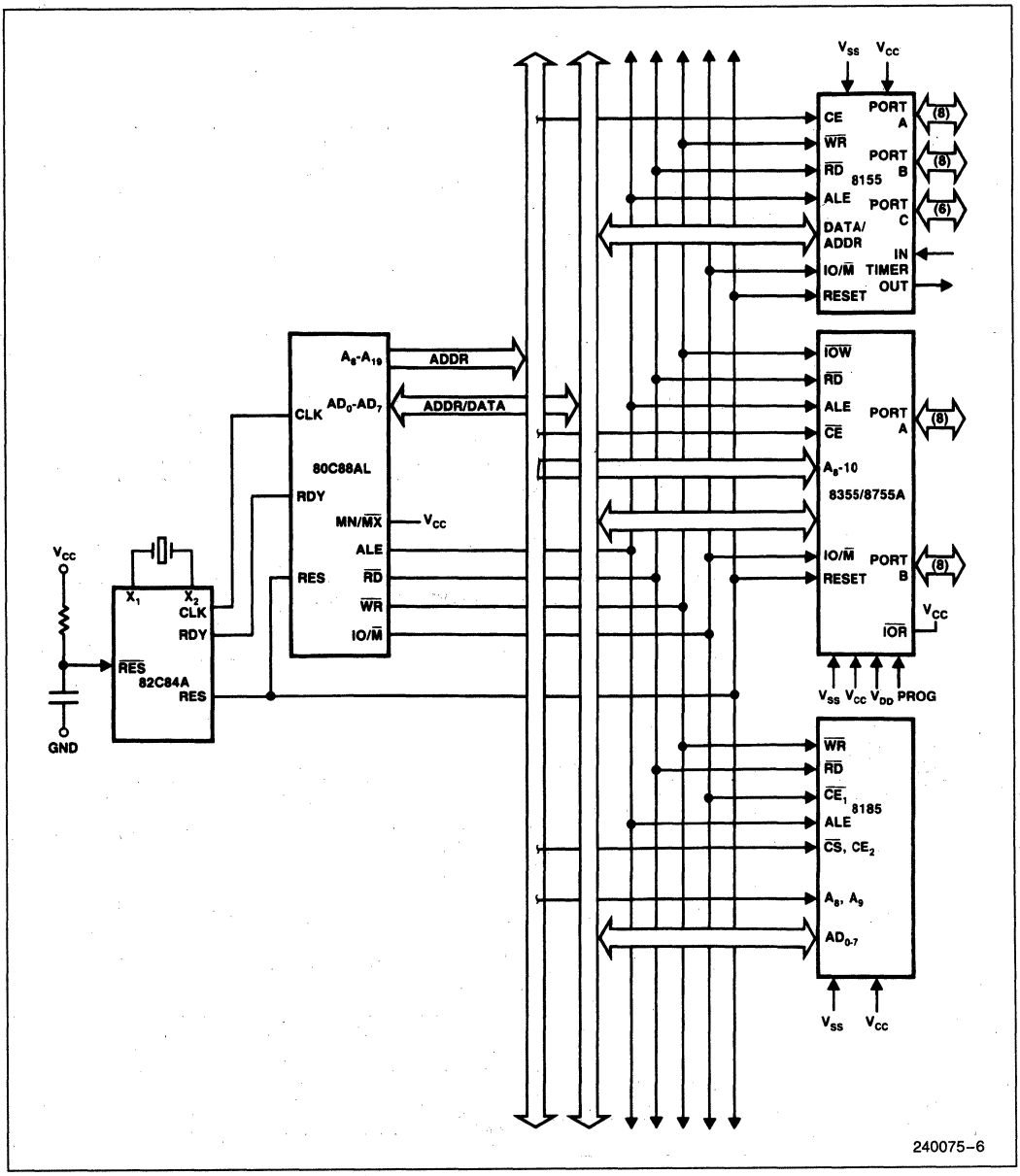


Figure 5. Multiplexed Bus Configuration

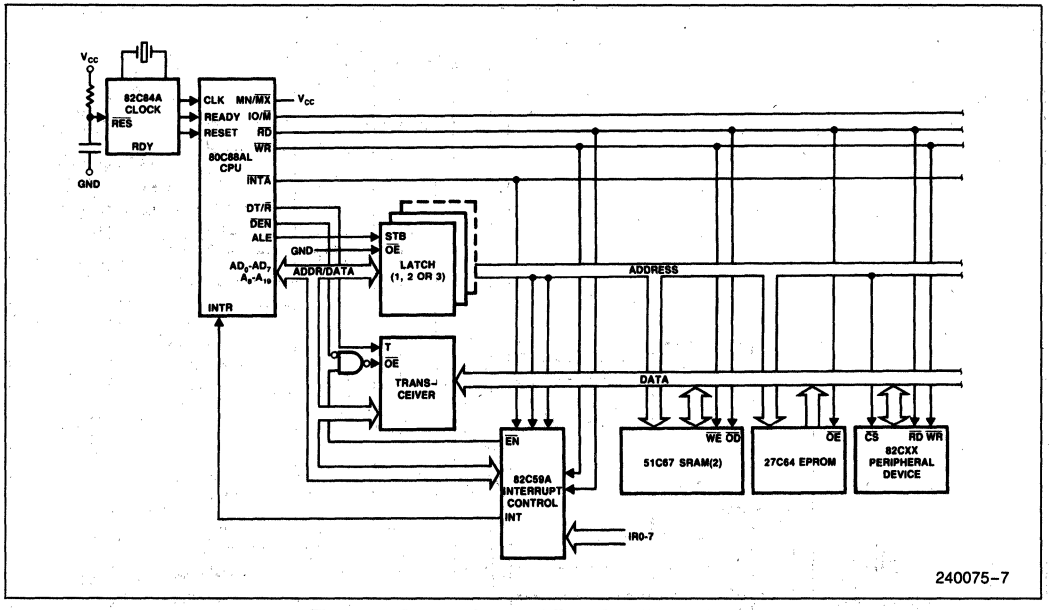


Figure 6. Demultiplexed Bus Configuration

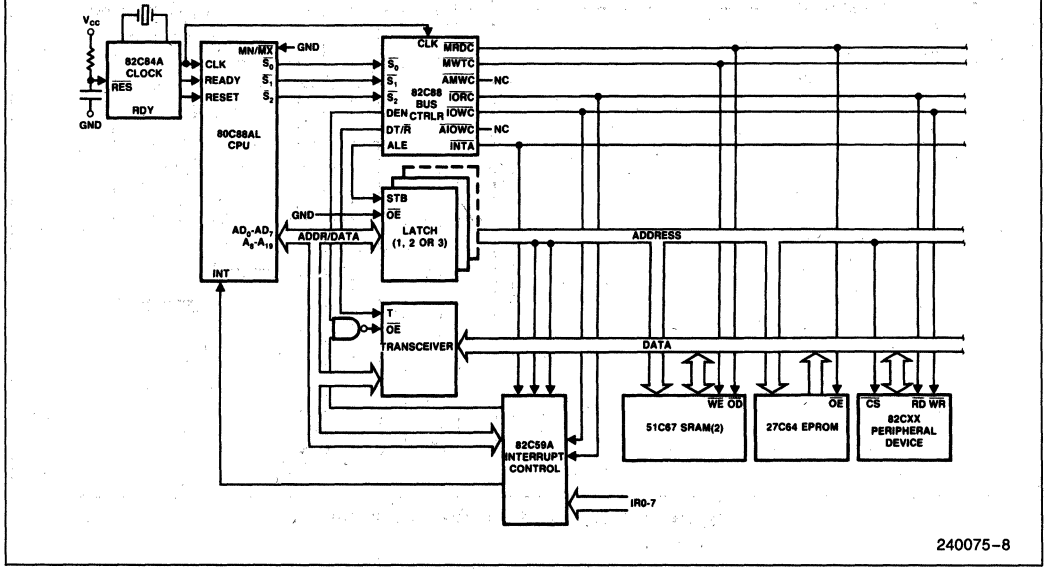


Figure 7. Fully Buffered System Using Bus Controller

Bus Operation

The 80C88AL address/data bus is broken into three parts—the lower eight address/data bits (AD0–AD7), the middle eight address bits (A8–A15), and the upper four address bits (A16–A19). The address/data bits and the highest four address bits are time multiplexed. This technique provides the most efficient use of pins on the processor. The middle eight address bits are not multiplexed, i.e. they remain valid throughout each bus cycle. In addition, the bus can be demultiplexed at the processor with a single address latch if a standard, non-multiplexed bus is desired for the system.

Each processor bus cycle consists of at least four CLK cycles. These are referred to as T1, T2, T3, and T4. (See Figure 8). The address is emitted from the processor during T1 and data transfer occurs on the bus during T3 and T4. T2 is used primarily for changing the direction of the bus during read operations. In the event that a "NOT READY" indication is given by the addressed device, "wait" states (Tw) are inserted between T3 and T4. Each inserted "wait" state is of the same duration as a CLK cycle. Periods can occur between 80C88AL driven bus cycles. These are referred to as "idle" states (Ti), or inactive CLK cycles. The processor uses these cycles for internal housekeeping.

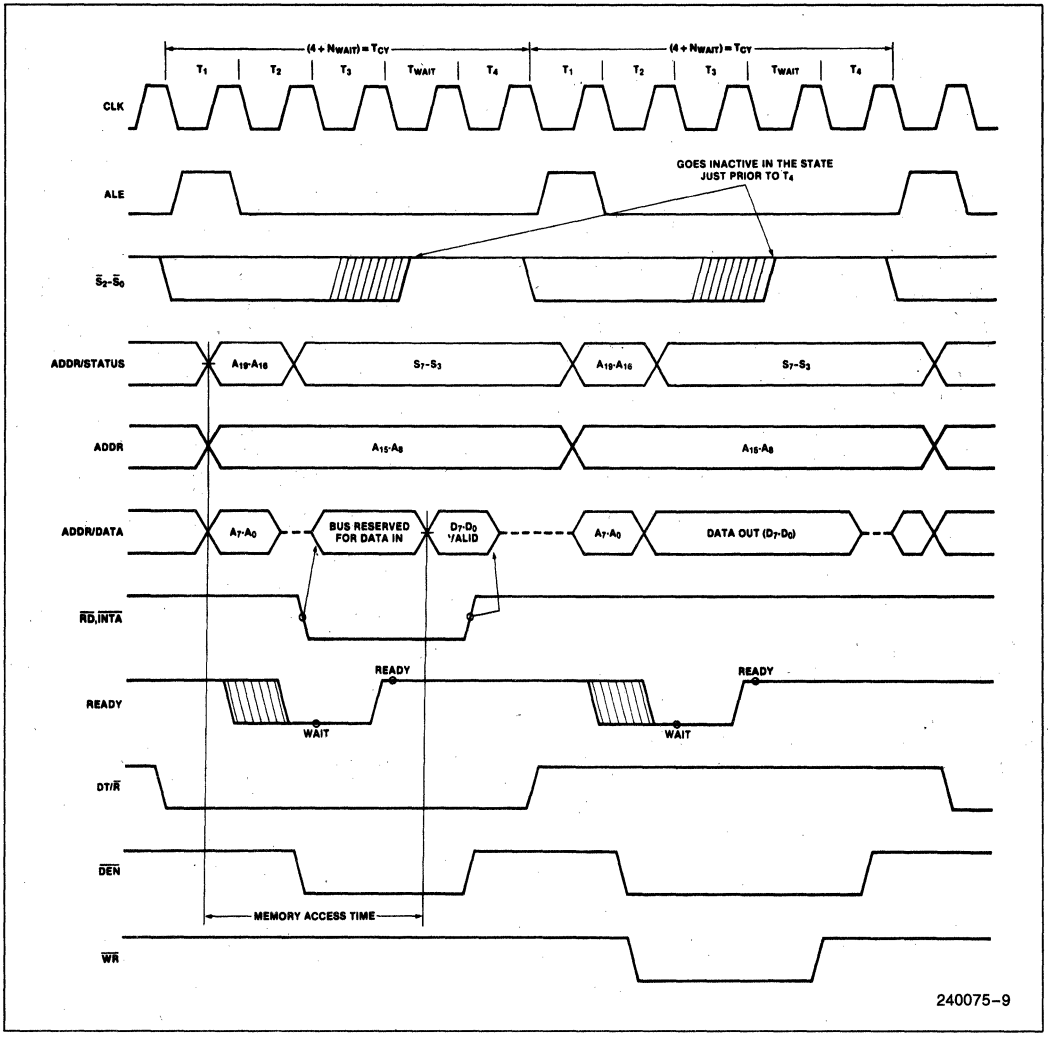


Figure 8. Basic System Timing

During T1 of any bus cycle, the ALE (address latch enable) signal is emitted (by either the processor or the 82C88 bus controller, depending on the MN/$\overline{MX}$ strap). At the trailing edge of this pulse, a valid address and certain status information for the cycle may be latched.

Status bits $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$ are used by the bus controller, in maximum mode, to identify the type of bus transaction according to the following table:

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	CHARACTERISTICS
0 (LOW)	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1 (HIGH)	0	0	Instruction Fetch
1	0	1	Read Data from Memory
1	1	0	Write Data to Memory
1	1	1	Passive (no bus cycle)

Status bits S3 through S6 are multiplexed with high order address bits and are therefore valid during T2 through T4. S3 and S4 indicate which segment register was used for this bus cycle in forming the address according to the following table:

S_4	S_3	CHARACTERISTICS
0 (LOW)	0	Alternate Data (extra segment)
0	1	Stack
1 (HIGH)	0	Code or None
1	1	Data

S5 is a reflection of the PSW interrupt enable bit. S6 is equal to 0.

I/O ADDRESSING

In the 80C88AL, I/O operations can address up to a maximum of 64k I/O registers. The I/O address appears in the same format as the memory address on bus lines A15–A0. The address lines A19–A16 are zero in I/O operations. The variable I/O instructions, which use register DX as a pointer, have full address capability, while the direct I/O instructions directly address one or two of the 256 I/O byte locations in page 0 of the I/O address space. I/O ports are addressed in the same manner as memory locations.

Designers familiar with the 8085 or upgrading an 8085 design should note that the 8085 addresses I/O with an 8-bit address on both halves of the 16-bit address bus. The 80C88AL uses a full 16-bit address on its lower 16 address lines.

EXTERNAL INTERFACE

PROCESSOR RESET AND INITIALIZATION

Processor initialization or start up is accomplished with activation (HIGH) of the RESET pin. The 80C88AL RESET is required to be HIGH for four or more clock cycles. The 80C88AL will terminate operations on the high-going edge of RESET and will remain dormant as long as RESET is HIGH. The low-going transition of RESET triggers an internal reset sequence for approximately 7 clock cycles. After this interval the 80C88AL operates normally, beginning with the instruction in absolute location FFFF0H. (See Figure 4.) The RESET input is internally synchronized to the processor clock. At initialization, the HIGH to LOW transition of RESET must occur no sooner than 50 μs after power up, to allow complete initialization of the 80C88AL.

NMI asserted prior to the 2nd clock after the end of RESET will not be honored. If NMI is asserted after that point and during the internal reset sequence, the processor may execute one instruction before responding to the interrupt. A hold request active immediately after RESET will be honored before the first instruction fetch.

All 3-state outputs float to 3-state OFF(1) during RESET. Status is active in the idle state for the first clock after RESET becomes active and then floats to 3-state OFF(1). ALE and HLDA are driven low.

NOTE:
1. See the section on Bus Hold Circuitry.

BUS HOLD CIRCUITRY

To avoid high current conditions caused by floating inputs to CMOS devices and to eliminate the need for pull-up/down resistors, "bus-hold" circuitry has been used on the 80C88AL pins 2–16, 26–32, and 34–39 (Figure 9a, 9b). These circuits will maintain the last valid logic state if no driving source is present (i.e. an unconnected pin or a driving source which goes to a high impedance state). To overdrive the "bus hold" circuits, an external driver must be capable of supplying 350 μA minimum sink or source current at valid input voltage levels. Since this "bus hold" circuitry is active and not a "resistive" type element, the associated power supply current is negligible and power dissipation is significantly reduced when compared to the use of passive pull-up resistors.

INTERRUPT OPERATIONS

Interrupt operations fall into two classes: software or hardware initiated. The software initiated interrupts and software aspects of hardware interrupts are specified in the instruction set description in the iAPX 88 book or the iAPX 86,88 User's Manual. Hardware interrupts can be classified as nonmaskable or maskable.

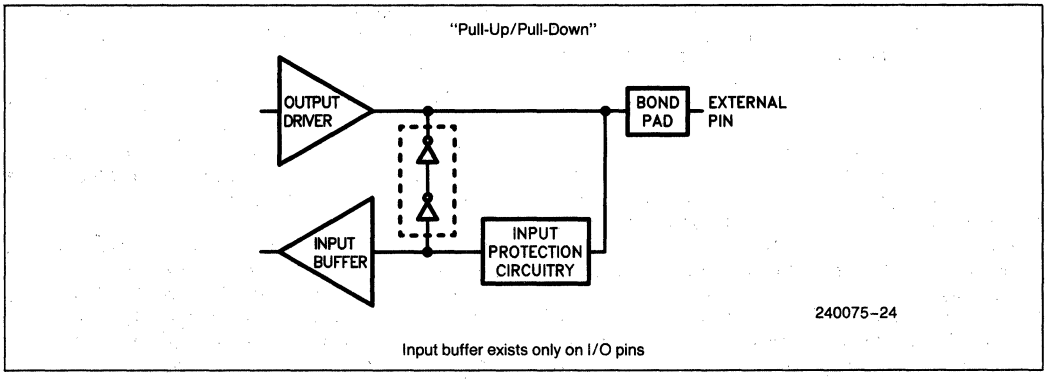


Figure 9a. Bus hold circuitry pin 2–16, 35–39 for P-DIP package.

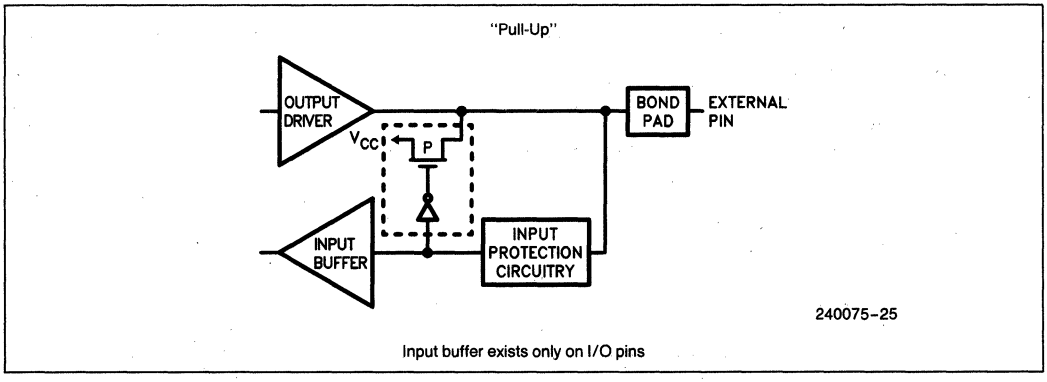


Figure 9b. Bus hold circuitry pin 26–32, 34 for P-DIP package.

Interrupts result in a transfer of control to a new program location. A 256 element table containing address pointers to the interrupt service program locations resides in absolute locations 0 through 3FFH (See Figure 4), which are reserved for this purpose. Each element in the table is 4 bytes in size and corresponds to an interrupt "type." An interrupting device supplies an 8-bit type number, during the interrupt acknowledge sequence, which is used to vector through the appropriate element to the new interrupt service program location.

NON-MASKABLE INTERRUPT (NMI)

The processor provides a single non-maskable interrupt (NMI) pin which has higher priority than the maskable interrupt request (INTR) pin. A typical use would be to activate a power failure routine. The NMI is edge-triggered on a LOW to HIGH transition. The activation of this pin causes a type 2 interrupt.

NMI is required to have a duration in the HIGH state of greater than two clock cycles, but is not required to be synchronized to the clock. Any higher going transition of NMI is latched on-chip and will be serviced at the end of the current instruction or between whole moves (2 bytes in the case of word moves) of a block type instruction. Worst case response to NMI would be for multiply, divide, and variable shift instructions. There is no specification on the occurrence of the low-going edge; it may occur before, during, or after the servicing of NMI. Another high-going edge triggers another response if it occurs after the start of the NMI procedure. The signal must be free of logical spikes in general and be free of bounces on the low-going edge to avoid triggering extraneous responses.

MASKABLE INTERRUPT (INTR)

The 80C88AL provides a single interrupt request input (INTR) which can be masked internally by software with the resetting of the interrupt enable (IF) flag bit. The interrupt request signal is level triggered. It is internally synchronized during each clock cycle on the high-going edge of CLK. To be responded to, INTR must be present (HIGH) during the clock period preceding the end of the current instruction or the end of a whole move for a block type instruction. During interrupt response sequence, further interrupts are disabled. The enable bit is reset as part of the response to any interrupt (INTR, NMI, software interrupt, or single step), although the FLAGS register which is automatically pushed onto the stack reflects the state of the processor prior to the interrupt. Until the old FLAGS register is restored, the enable bit will be zero unless specifically set by an instruction.

During the response sequence (See Figure 10), the processor executes two successive (back to back) interrupt acknowledge cycles. The 80C88AL emits the LOCK signal (maximum mode only) from T2 of the first bus cycle until T2 of the second. A local bus "hold" request will not be honored until the end of the second bus cycle. In the second bus cycle, a

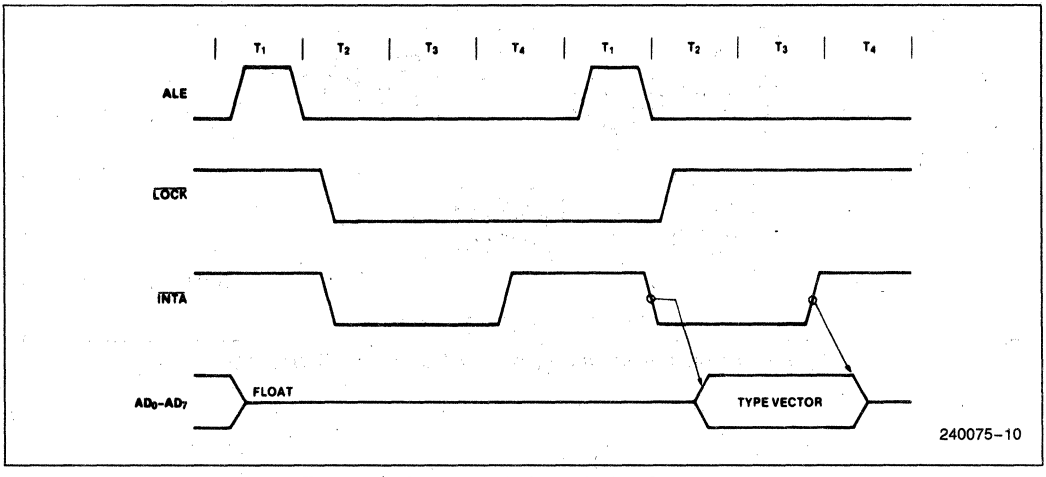


Figure 10. Interrupt Acknowledge Sequence

byte is fetched from the external interrupt system (e.g., 82C59A PIC) which identifies the source (type) of the interrupt. This byte is multiplied by four and used as a pointer into the interrupt vector lookup table. An INTR signal left HIGH will be continually responded to within the limitations of the enable bit and sample period. The interrupt return instruction includes a flags pop which returns the status of the original interrupt enable bit when it restores the flags.

HALT

When a software HALT instruction is executed, the processor indicates that it is entering the HALT state in one of two ways, depending upon which mode is strapped. In minimum mode, the processor issues ALE, delayed by one clock cycle, to allow the system to latch the halt status. Halt status is available on IO/$\overline{M}$, DT/$\overline{R}$, and $\overline{SSO}$. In maximum mode, the processor issues appropriate HALT status on $\overline{S2}$, $\overline{S1}$, and $\overline{S0}$, and the 82C88 bus controller issues one ALE. The 80C88AL will not leave the HALT state when a local bus hold is entered while in HALT. In this case, the processor reissues the HALT indicator at the end of the local bus hold. An interrupt request or RESET will force the 80C88AL out of the HALT state.

READ/MODIFY/WRITE (SEMAPHORE) OPERATIONS VIA LOCK

The LOCK status information is provided by the processor when consecutive bus cycles are required during the execution of an instruction. This allows the processor to perform read/modify/write operations on memory (via the "exchange register with memory" instruction), without another system bus master receiving intervening memory cycles. This is useful in multiprocessor system configurations to accomplish "test and set lock" operations. The $\overline{LOCK}$ signal is activated (LOW) in the clock cycle following decoding of the LOCK prefix instruction. It is deactivated at the end of the last bus cycle of the instruction following the LOCK prefix. While $\overline{LOCK}$ is active, a request on a $\overline{RQ}/\overline{GT}$ pin will be recorded, and then honored at the end of the LOCK.

EXTERNAL SYNCHRONIZATION VIA $\overline{TEST}$

As an alternative to interrupts, the 80C88AL provides a single software-testable input pin ($\overline{TEST}$). This input is utilized by executing a WAIT instruction. The single WAIT instruction is repeatedly executed until the $\overline{TEST}$ input goes active (LOW). The execution of WAIT does not consume bus cycles once the queue is full.

If a local bus request occurs during WAIT execution, the 80C88AL 3-states all output drivers. If interrupts are enabled, the 80C88AL will recognize interrupts and process them. The WAIT instruction is then refetched, and reexecuted.

BASIC SYSTEM TIMING

In minimum mode, the MN/$\overline{MX}$ pin is strapped to V_{CC} and the processor emits bus control signals compatible with the 8085 bus structure. In maximum mode, the MN/$\overline{MX}$ pin is strapped to GND and the processor emits coded status information which the 82C88 bus controller uses to generate MULTIBUS compatible bus control signals.

System Timing — Minimum System

(See Figure 8.)

The read cycle begins in T1 with the assertion of the address latch enable (ALE) signal. The trailing (low going) edge of this signal is used to latch the address information, which is valid on the address/data bus (AD0–AD7) at this time, into a latch. Address lines A8 through A15 do not need to be latched because they remain valid throughout the bus cycle. From T1 to T4 the IO/$\overline{M}$ signal indicates a memory or I/O operation. At T2 the address is removed from the address/data bus and the bus goes to a high impedance state. The read control signal is also asserted at T2. The read ($\overline{RD}$) signal causes the addressed device to enable its data bus drivers to the local bus. Some time later, valid data will be available on the bus and the addressed device will drive the READY line HIGH. When the processor returns the read signal to a HIGH level, the addressed device will again 3-state its bus drivers. If a transceiver is required to buffer the 80C88AL local bus, signals DT/$\overline{R}$ and $\overline{DEN}$ are provided by the 80C88AL.

A write cycle also begins with the assertion of ALE and the emission of the address. The IO/$\overline{M}$ signal is again asserted to indicate a memory or I/O write operation. In T2, immediately following the address emission, the processor emits the data to be written into the addressed location. This data remains valid until at least the middle of T4. During T2, T3, and T_W, the processor asserts the write control signal. The write ($\overline{WR}$) signal becomes active at the beginning of T2, as opposed to the read, which is delayed somewhat into T2 to provide time for the bus to float.

The basic difference between the interrupt acknowledge cycle and a read cycle is that the interrupt acknowledge ($\overline{INTA}$) signal is asserted in place of the read ($\overline{RD}$) signal and the address bus is floated. (See Figure 10.) In the second of two successive $\overline{INTA}$ cycles, a byte of information is read from the data bus, as supplied by the interrupt system logic (i.e. 82C59A priority interrupt controller). This byte identifies the source (type) of the interrupt. It is multiplied by four and used as a pointer into the interrupt vector lookup table, as described earlier.

BUS TIMING — MEDIUM COMPLEXITY SYSTEMS

(See Figure 11.)

For medium complexity systems, the MN/$\overline{MX}$ pin is connected to GND and the 82C88 bus controller is added to the system, as well as a latch for latching the system address, and a transceiver to allow for bus loading greater than the 80C88AL is capable of handling. Signals ALE, $\overline{DEN}$, and DT/$\overline{R}$ are generated by the 82C88 instead of the processor in this configuration, although their timing remains relatively the same. The 80C88AL status outputs ($\overline{S2}$, $\overline{S1}$, and $\overline{S0}$) provide type of cycle information and become 82C88 inputs. This bus cycle information specifies read (code, data, or I/O), write (data or I/O), interrupt acknowledge, or software halt. The 82C88 thus issues control signals specifying memory read or write, I/O read or write, or interrupt acknowledge. The 82C88 provides two types of write strobes, normal and advanced, to be applied as required. The normal write strobes have data valid at the leading edge of write. The advanced write strobes have the same timing as read strobes, and hence, data is not valid at the leading edge of write. The transceiver receives the usual $\overline{T}$ and $\overline{OE}$ inputs from the 82C88's DT/$\overline{R}$ and $\overline{DEN}$ outputs.

The pointer into the interrupt vector table, which is passed during the second $\overline{INTA}$ cycle, can derive from an 82C59A located on either the local bus or the system bus. If the master 82C59A priority interrupt controller is positioned on the local bus, a TTL gate is required to disable the transceiver when reading from the master 82C59A during the interrupt acknowledge sequence and software "poll".

THE 80C88AL COMPARED TO THE 80C86AL

The 80C88AL CPU is an 8-bit processor designed around the 80C86AL internal structure. Most internal functions of the 80C88AL are identical to the equivalent 80C86AL functions. The 80C88AL handles the external bus the same way the 80C86AL does with the distinction of handling only 8 bits at a time. Sixteen-bit operands are fetched or written in two consecutive bus cycles. Both processors will appear identical to the software engineer, with the exception of execution time. The internal register structure is identical and all instructions have the same end result. The differences between the 80C88AL and 80C86AL are outlined below. The engineer who is unfamiliar with the 80C86AL is referred to the iAPX 86, 88 User's Manual, Chapters 2 and 4, for function description and instruction set information. Internally, there are three differences between the 80C88AL and the 80C86AL. All changes are related to the 8-bit bus interface.

- The queue length is 4 bytes in the 80C88AL, whereas the 80C86AL queue contains 6 bytes, or three words. The queue was shortened to prevent overuse of the bus by the BIU when prefetching instructions. This was required because of the additional time necessary to fetch instructions 8 bits at a time.
- To further optimize the queue, the prefetching algorithm was changed. The 80C88AL BIU will fetch a new instruction to load into the queue each time there is a 1 byte hole (space available) in the queue. The 80C86AL waits until a 2-byte space is available.
- The internal execution time of the instruction set is affected by the 8-bit interface. All 16-bit fetches and writes from/to memory take an additional four clock cycles. The CPU is also limited by the speed of instruction fetches. This latter problem only occurs when a series of simple operations occur. When the more sophisticated instructions of the 80C88AL are being used, the queue has time to fill and the execution proceeds as fast as the execution unit will allow.

The 80C88AL and 80C86AL are completely software compatible by virtue of their identical execution units. Software that is system dependent may not be completely transferable, but software that is not system dependent will operate equally as well on an 80C88AL or an 80C86AL.

The hardware interface of the 80C88AL contains the major differences between the two CPUs. The pin assignments are nearly identical, however with the following functional changes:

- A8–A15 — These pins are only address outputs on the 80C88AL. These address lines are latched internally and remain valid throughout a bus cycle in a manner similar to the 8085 upper address lines.

- $\overline{BHE}$ has no meaning on the 80C88AL and has been eliminated.
- $\overline{SSO}$ provides the $\overline{SO}$ status information in the minimum mode. This output occurs on pin 34 in minimum mode only. DT/$\overline{R}$, IO/$\overline{M}$, and $\overline{SSO}$ provide the complete bus status in minimum mode.
- IO/$\overline{M}$ has been inverted to be compatible with the MCS-85 bus structure.
- ALE is delayed by one clock cycle in the minimum mode when entering HALT, to allow the status to be latched with ALE.

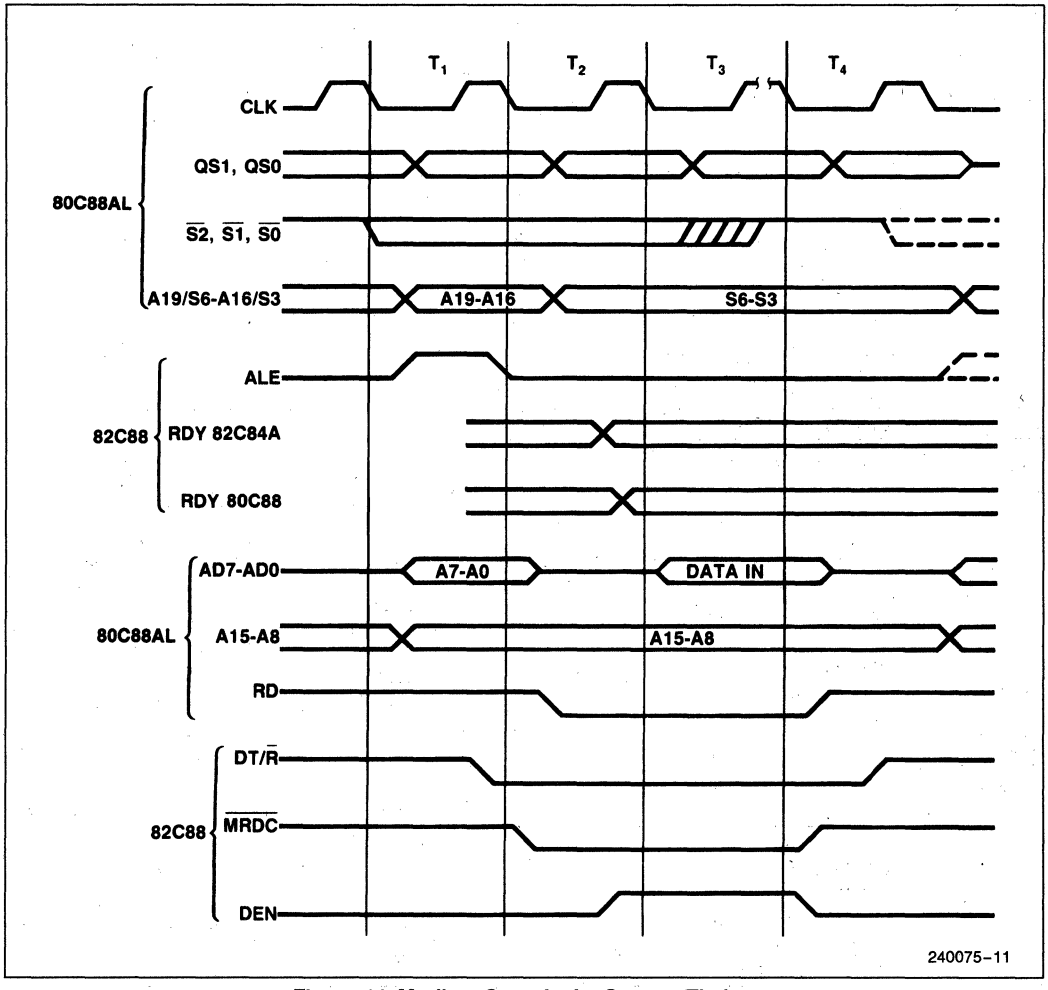


Figure 11. Medium Complexity System Timing

ABSOLUTE MAXIMUM RATINGS*

Supply Voltage
(With respect to ground) . . . . . −0.5 to 8.0V
Input Voltage Applied
(w.r.t. ground) . . . . . −2.0 to V_{CC} + 0.5V
Output Voltage Applied
(w.r.t. ground) . . . . . −0.5 to V_{CC} + 0.5V
Power Dissipation 1.0W
Storage Temperature . . . . . −65°C to +150°C
Ambient Temperature Under Bias . . . . 0°C to +70°C
Case Temperature (Plastic) . . . . . 0°C to 80°C
Case Temperature (PLCC) . . . . . 0°C to 85°C

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, T_{CASE} (Plastic) = 0°C to 80°C, T_{CASE} (PLCC) = 0°C to 85°C, V_{CC} = 5V ±10% for 80C88AL, V_{CC} = 5V ±5% for 80C88AL-2

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage		+0.8	V	(Note 4)
V_{IH}	Input High Voltage (All inputs except clock and MN/$\overline{MX}$)	2.0		V	(Note 5)
V_{CH}	Clock and MN/$\overline{MX}$ Input High Voltage	V_{CC}−0.8		V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2.5 mA
V_{OH}	Output High Voltage	3.0 V_{CC}−0.4		V	I_{OH} = −2.5 mA I_{OH} = −100 μA
I_{CC}	Power Supply Current		10 mA/MHz		V_{IL} = GND, V_{IH} = V_{CC}
I_{CCS}	Standby Supply Current		750	μA	V_{IN} = V_{CC} or GND Outputs Unloaded CLK = GND or V_{CC}
I_{LI}	Input Leakage Current		±1.0	μA	0V≤V_{IN}≤V_{CC}
I_{BHL}	Input Leakage Current (Bus Hold Low)	50	300	μA	V_{IN} = 0.8V
I_{BHH}	Input Leakage Current (Bus Hold High)	−50	−300	μA	V_{IN} = 3.0V
I_{BHLO}	Bus Hold Low Overdrive		400	μA	(Note 2)
I_{BHHO}	Bus Hold High Overdrive		−400	μA	(Note 3)
I_{LO}	Output Leakage Current		±10	μA	V_{OUT} = GND or V_{CC}
C_{IN}	Capacitance of Input Buffer (All inputs except AD_0–AD_7, $\overline{RQ}/\overline{GT}$)		5	pF	(Note 1)
C_{IO}	Capacitance of I/O Buffer (AD_0–AD_7, $\overline{RQ}/\overline{GT}$)		20	pF	(Note 1)
C_{OUT}	Output Capacitance		15	pF	(Note 1)

NOTES:
1. Characterization conditions are a) Frequency = 1 MHz, b) Unmeasured pins at GND
   c) V_{IN} at +5.0V or GND.
2. An external driver must source at least I_{BHLO} to switch this node from LOW to HIGH.
3. An external driver must sink at least I_{BHHO} to switch this node from HIGH to LOW.
4. V_{IL} for all input pins (except MN/$\overline{MX}$ pin) tested with MN/$\overline{MX}$ pin = GND.
5. V_{IH} tested with MN/$\overline{MX}$ pin = V_{CC}.

A.C. CHARACTERISTICS T_A = 0°C to 70°C, T_{CASE} (Plastic) = 0°C to 80°C, T_{CASE} (PLCC) = 0°C to 85°C, V_{CC} = 5V ±10% for 80C88AL, V_{CC} = 5V ±5% for 80C88AL-2

MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

		80C88AL		80C88AL-2			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Units**	**Test Conditions**
TCLCL	CLK Cycle Period	200	D.C.	125	D.C.	ns	
TCLCH	CLK Low Time	118		68		ns	
TCHCL	CLK High Time	69		44		ns	
TCH1CH2	CLK Rise Time		10		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10		10	ns	From 3.5V to 1.0V
TDVCL	Data in Setup Time	30		20		ns	
TCLDX	Data in Hold Time	10		10		ns	
TR1VCL	RDY Setup Time into 82C84A (Notes 1, 2)	35		35		ns	
TCLR1X	RDY Hold Time into 82C84A (Notes 1, 2)	0		0		ns	
TRYHCH	READY Setup Time into 80C88AL	118		68		ns	
TCHRYX	READY Hold Time into 80C88AL	30		20		ns	
TRYLCL	READY Inactive to CLK (Note 3)	−8		−8		ns	
THVCH	HOLD Setup Time	35		20		ns	
TINVCH	INTR, NMI, $\overline{\text{TEST}}$ Setup Time (Note 2)	30		15		ns	
TILIH	Input Rise Time (Except CLK) (Note 4)		15		15	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK) (Note 4)		15		15	ns	From 2.0V to 0.8V

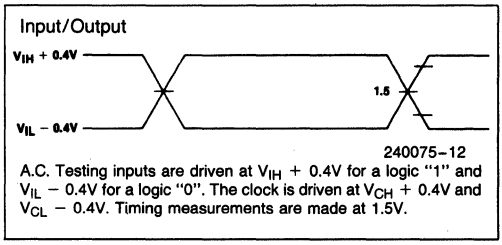
A.C. CHARACTERISTICS (Continued)

TIMING RESPONSES

		80C88AL		80C88AL-2			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Units**	**Test Conditions**
TCLAV	Address Valid Delay	10	70	10	60	ns	
TCLAX	Address Hold Time	10		10		ns	
TCLAZ	Address Float Delay	TCLAX	80	TCLAX	50	ns	
TLHLL	ALE Width	TCLCH-20		TCLCH-10		ns	
TCLLH	ALE Active Delay		80		50	ns	
TCHLL	ALE Inactive Delay		85		55	ns	
TLLAX	Address Hold Time to ALE Inactive	TCHCL-25		TCHCL-25		ns	
TCLDV	Data Valid Delay	10	110	10	60	ns	
TCHDX	Data Hold Time	10		10		ns	
TWHDX	Data Hold Time After $\overline{WR}$	TCLCH-30		TCLCH-30		ns	
TCVCTV	Control Active Delay 1	10	110	10	70	ns	
TCHCTV	Control Active Delay 2	10	110	10	60	ns	
TCVCTX	Control Inactive Delay	10	110	10	70	ns	
TAZRL	Address Float to READ Active	0		0		ns	
TCLRL	$\overline{RD}$ Active Delay	10	165	10	100	ns	
TCLRH	$\overline{RD}$ Inactive Delay	10	150	10	80	ns	
TRHAV	$\overline{RD}$ Inactive to Next Address Active	TCLCL-45		TCLCL-40		ns	
TCLHAV	HLDA Valid Delay	10	160	10	100	ns	
TRLRH	$\overline{RD}$ Width	2TCLCL-75		2TCLCL-50		ns	
TWLWH	$\overline{WR}$ Width	2TCLCL-60		2TCLCL-40		ns	
TAVAL	Address Valid to ALE Low	TCLCH-60		TCLCH-40		ns	
TOLOH	Output Rise Time (Note 4)		15		15	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time (Note 4)		15		15	ns	From 2.0V to 0.8V

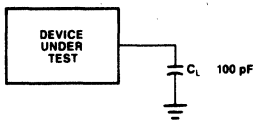
NOTES:
1. Signal at 82C84A shown for reference only. See 82C84A data sheet for the most recent specifications.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T2 state (8 ns into T3 state).
4. These parameters are characterized and not 100% tested.

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. Testing inputs are driven at V_{IH} + 0.4V for a logic "1" and V_{IL} − 0.4V for a logic "0". The clock is driven at V_{CH} + 0.4V and V_{CL} − 0.4V. Timing measurements are made at 1.5V.

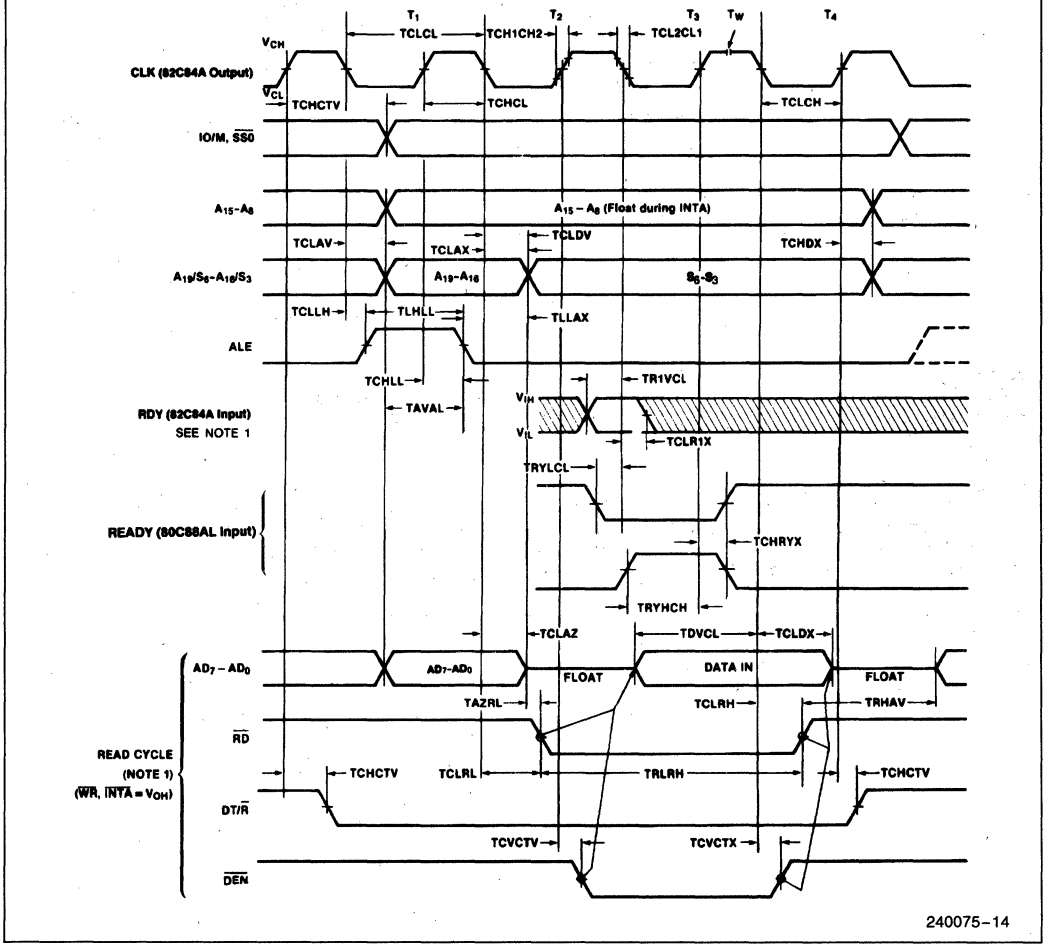
A.C. TESTING LOAD CIRCUIT



C_L Includes Jig Capacitance

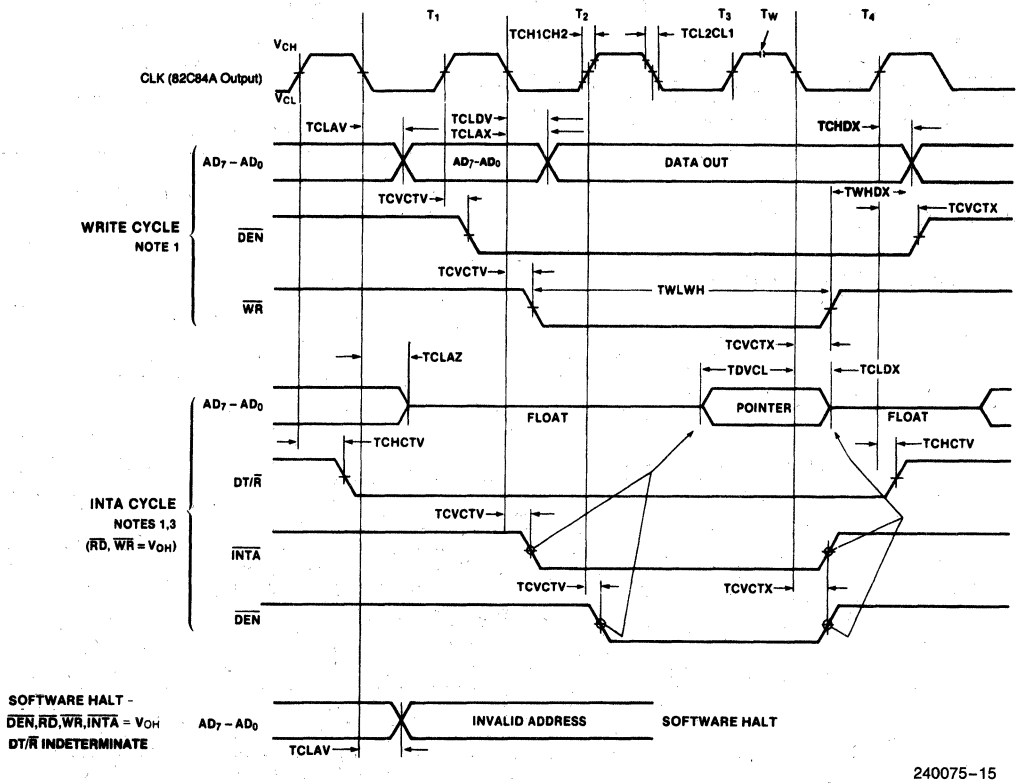
WAVEFORMS

BUS TIMING — MINIMUM MODE SYSTEM



WAVEFORMS (Continued)

BUS TIMING — MINIMUM MODE SYSTEM (Continued)



240075-15

NOTES:
1. All output timing measurements are made at 1.5V unless otherwise noted.
2. RDY is sampled near the end of T_2, T_3, T_W to determine if T_W machines states are to be inserted.
3. Two INTA Cycles run back-to-back. The 80C88AL local ADDR/Data bus is floating during both INTA Cycles. Control signals are shown for the second INTA cycle.
4. Signals at 82C84A are shown for reference only.

A.C. CHARACTERISTICS

MAX MODE SYSTEM (USING 82C88 BUS CONTROLLER) TIMING REQUIREMENTS

Symbol	Parameter	80C88AL		80C88AL-2		Units	Test Conditions
		Min	Max	Min	Max		
TCLCL	CLK Cycle Period	200	D.C.	125	D.C.	ns	
TCLCH	CLK Low Time	118		68		ns	
TCHCL	CLK High Time	69		44		ns	
TCH1CH2	CLK Rise Time		10		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10		10	ns	From 3.5V to 1.0V
TDVCL	Data In Setup Time	30		20		ns	
TCLDX	Data In Hold Time	10		10		ns	
TR1VCL	RDY Setup Time into 82C84 (See Notes 1, 2)	35		35		ns	
TCLR1X	RDY Hold Time into 82C84 (See Notes 1, 2)	0		0		ns	
TRYHCH	READY Setup Time into 80C88AL	118		68		ns	
TCHRYX	READY Hold Time into 80C88AL	30		20		ns	
TRYLCL	READY Inactive to CLK (See Note 4)	−8		−8		ns	
TINVCH	Setup Time for Recognition (INTR, NMI, TEST) (See Note 2)	30		15		ns	
TGVCH	RQ/GT Setup Time	30		15		ns	
TCHGX	RQ Hold Time into 80C88AL	40		30		ns	
TILIH	Input Rise Time (Except CLK) (Note 5)		15		15	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK) (Note 5)		15		15	ns	From 2.0V to 0.8V

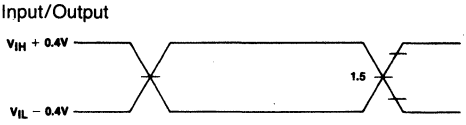
A.C. CHARACTERISTICS (Continued)

TIMING RESPONSES

		80C88AL		80C88AL-2			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Units**	**Test Conditions**
TCLML	Command Active Delay (Note 1)	5	45	5	35	ns	
TCLMH	Command Inactive Delay (Note 1)	5	45	5	35	ns	
TRYHSH	READY Active to Status Passive (Note 3)		110		65	ns	
TCHSV	Status Active Delay	10	100	10	60	ns	
TCLSH	Status Inactive Delay	10	130	10	70	ns	
TCLAV	Address Valid Delay	10	70	10	60	ns	
TCLAX	Address Hold Time	10		10		ns	
TCLAZ	Address Float Delay	TCLAX	80	TCLAX	50	ns	
TSVLH	Status Valid to ALE High (Note 1)		35		20	ns	
TSVMCH	Status Valid to MCE High (Note 1)		35		30	ns	
TCLLH	CLK Low to ALE Valid (Note 1)		35		20	ns	
TCLMCH	CLK Low to MCE High (Note 1)		35		25	ns	
TCHLL	ALE Inactive Delay (Note 1)	4	35	4	25	ns	
TCLDV	Data Valid Delay	10	110	10	60	ns	
TCHDX	Data Hold Time	10		10		ns	
TCVNV	Control Active Delay (Note 1)	5	45	5	45	ns	
TCVNX	Control Inactive Delay (Note 1)	5	45	10	45	ns	
TAZRL	Address Float to Read Active	0		0		ns	
TCLRL	RD Active Delay	10	165	10	100	ns	
TCLRH	RD Inactive Delay	10	150	10	80	ns	
TRHAV	RD Inactive to Next Address Active	TCLCL-45		TCLCL-40		ns	
TCHDTL	Direction Control Active Delay (Note 1)		50		50	ns	
TCHDTH	Direction Control Inactive Delay (Note 1)		35		30	ns	
TCLGL	GT Active Delay	0	85	0	50	ns	
TCLGH	GT Inactive Delay	0	85	0	50	ns	
TRLRH	RD Width	2TCLCL-75		2TCLCL-50		ns	
TOLOH	Output Rise Time (Note 5)		15		15	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time (Note 5)		15		15	ns	From 2.0V to 0.8V

NOTES:
1. Signal at 82C84A or 82C88 shown for reference only. See 82C84A and 82C88 data sheets for the most recent specifications.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T3 and wait states (8 ns into T3 state).
4. Applies only to T2 state (8 ns into T3 state).
5. These parameters are characterized and not 100% tested.

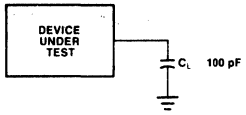
A.C. TESTING INPUT, OUTPUT WAVEFORM



240075-12

A.C. Testing inputs are driven at V_{IH} + 0.4V for a logic "1" and V_{IL} − 0.4V for a logic "0". The clock is driven at V_{CH} + 0.4V and V_{CL} − 0.4V. Timing measurements are made at 1.5V.

A.C. TESTING LOAD CIRCUIT

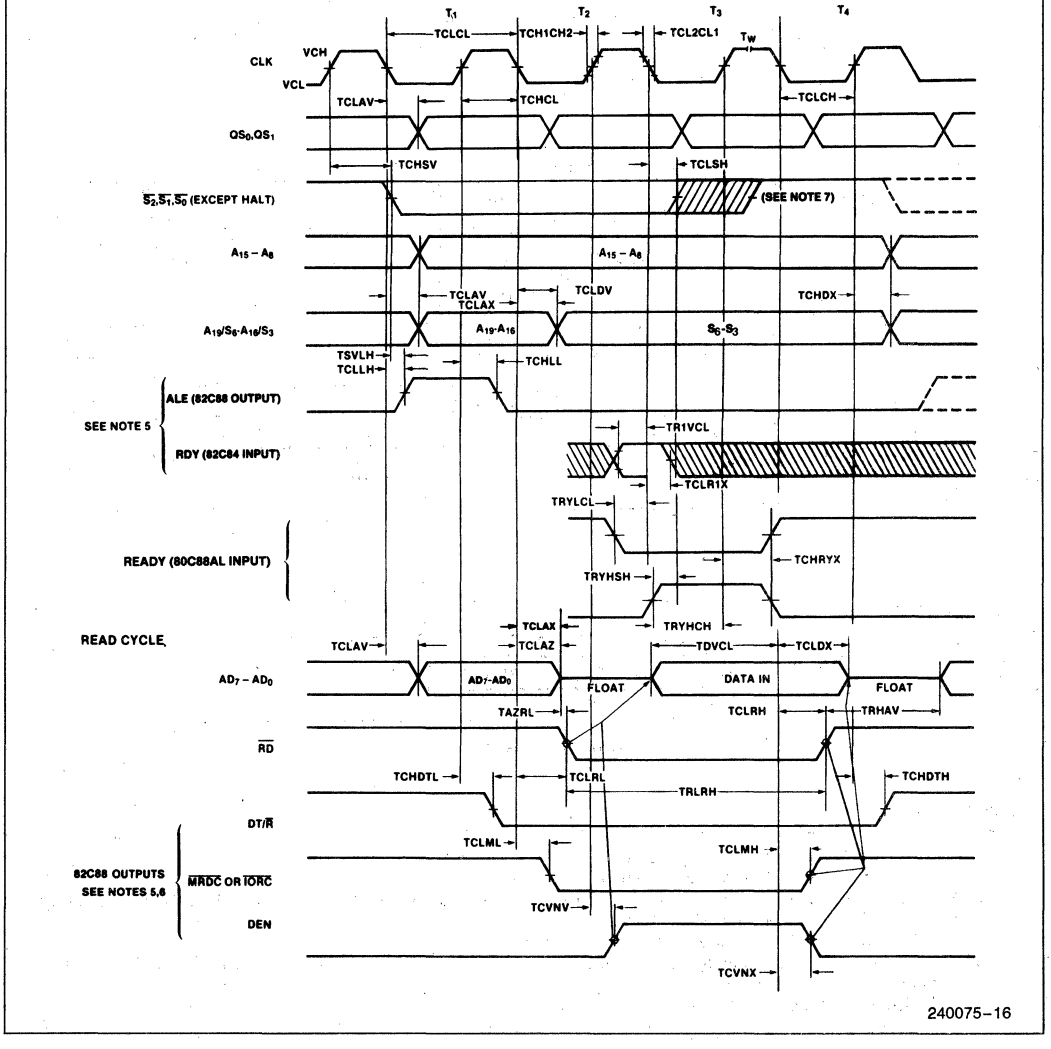


240075-13

C_L Includes Jig Capacitance

WAVEFORMS

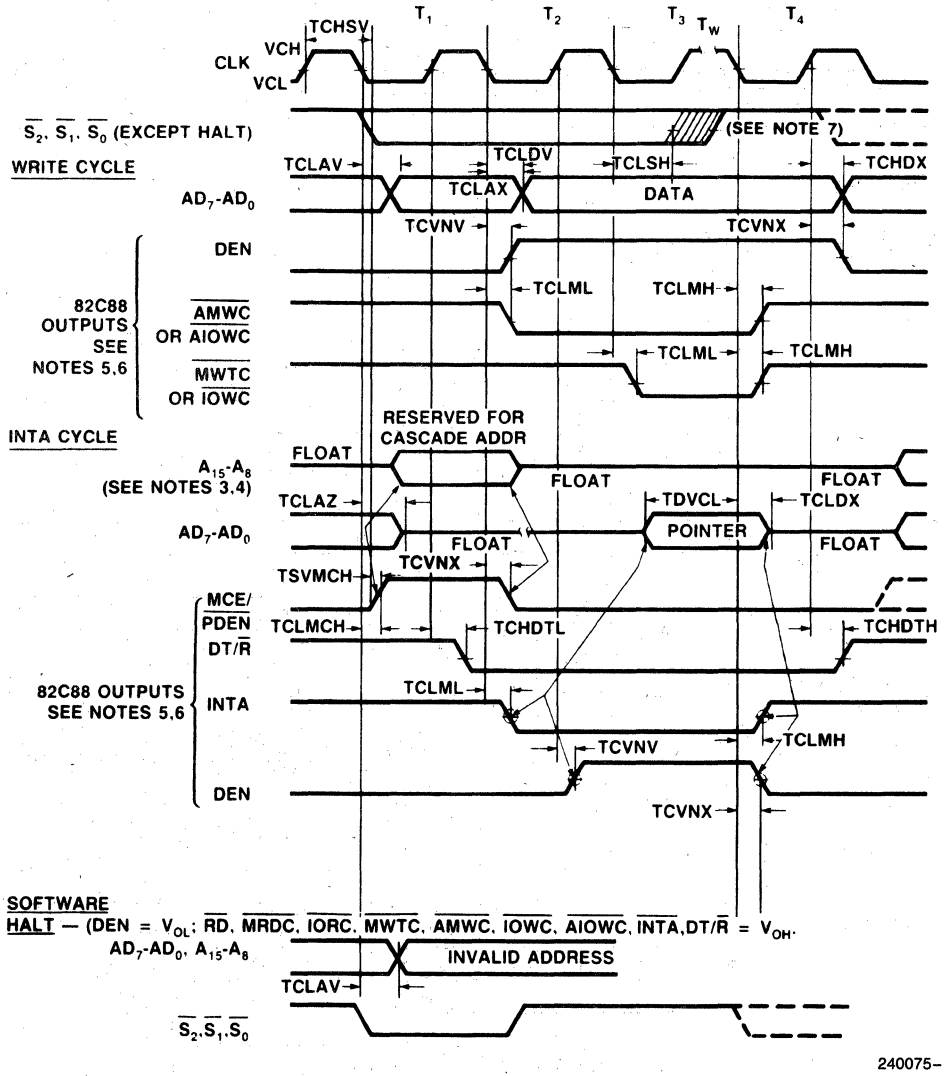
BUS TIMING—MAXIMUM MODE



240075-16

WAVEFORMS (Continued)

BUS TIMING — MAXIMUM MODE SYSTEM (USING 82C88)

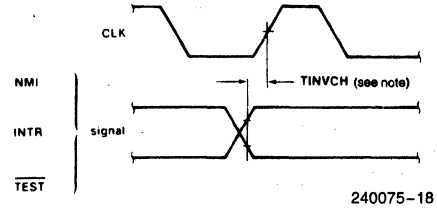


240075-17

NOTES:
1. All output timing measurements are made at 1.5V unless otherwise noted.
2. RDY is sampled near the end of T_2, T_3, T_W to determine if T_W machines states are to be inserted.
3. Cascade address is valid between first and second INTA cycles.
4. Two INTA cycles run back-to-back. The 80C88AL local ADDR/Data bus is floating during both INTA cycles. Control for pointer address is shown for second INTA cycle.
5. Signals at 82C84A or 82C88 are shown for reference only.
6. The issuance of the 82C88 command and control signals ($\overline{MRDC}$, $\overline{MWTC}$, $\overline{AMWC}$, $\overline{IORC}$, $\overline{IOWC}$, $\overline{AIOWC}$, $\overline{INTA}$ and DEN) lags the active high 82C88 CEN.
7. Status inactive in state just prior to T_4.

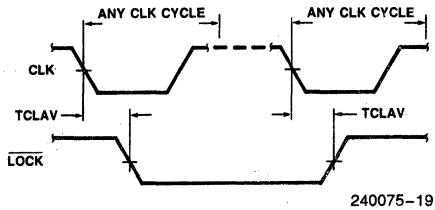
WAVEFORMS (Continued)

ASYNCHRONOUS SIGNAL RECOGNITION

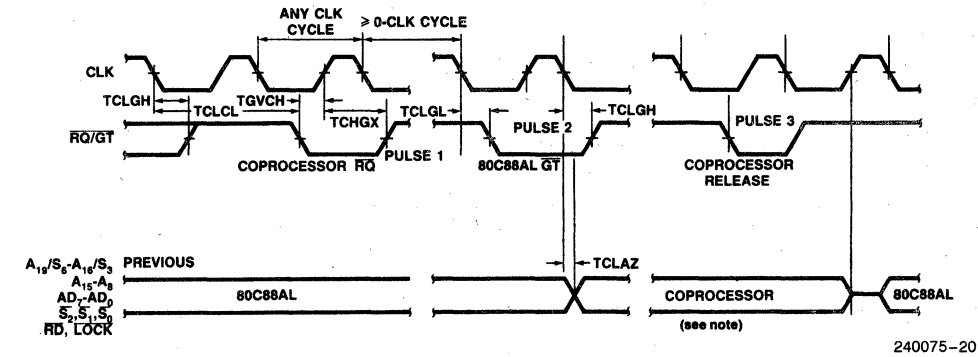


NOTE: Setup requirements for asynchronous signals only to guarantee recognition at next CLK.

BUS LOCK SIGNAL TIMING (MAXIMUM MODE ONLY)

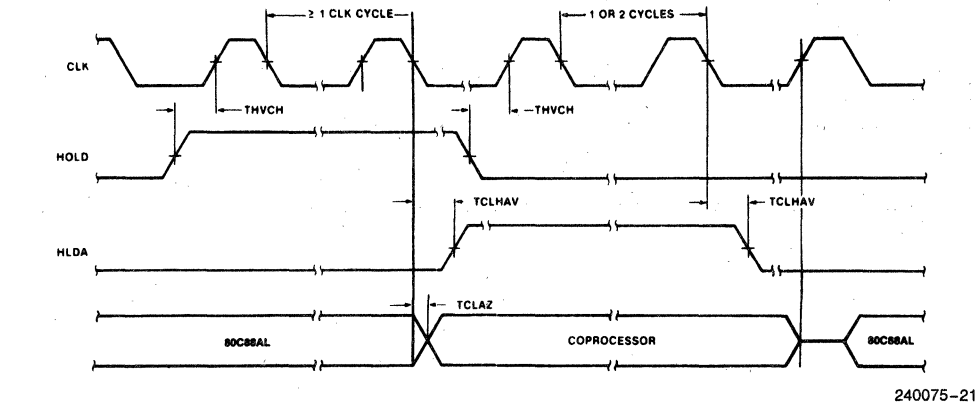


REQUEST/GRANT SEQUENCE TIMING (MAXIMUM MODE ONLY)



NOTE: The coprocessor may not drive the busses outside the region shown without risking contention.

HOLD/HOLD ACKNOWLEDGE TIMING (MINIMUM MODE ONLY)



80C86AL/80C88AL INSTRUCTION SET SUMMARY

Mnemonic and Description	Instruction Code			
DATA TRANSFER				
MOV = Move:	**76543210**	**76543210**	**76543210**	**76543210**
Register/Memory to/from Register	100010dw	mod reg r/m		
Immediate to Register/Memory	1100011w	mod 000 r/m	data	data if w 1
Immediate to Register	1011w reg	data	data if w 1	
Memory to Accumulator	1010000w	add-low	addr-high	
Accumulator to Memory	1010001w	addr-low	addr-high	
Register/Memory to Segment Register**	10001110	mod 0 reg r/m		
Segment Register to Register/Memory	10001100	mod 0 reg r/m		
PUSH = Push:				
Register/Memory	11111111	mod 110 r/m		
Register	01010 reg			
Segment Register	000 reg 110			
POP = Pop:				
Register/Memory	10001111	mod 000 r/m		
Register	01011 reg			
Segment Register	000 reg 111			
XCHG = Exchange:				
Register/Memory with Register	1000011w	mod reg r/m		
Register with Accumulator	10010 reg			
IN = Input from:				
Fixed Port	1110010w	port		
Variable Port	1110110w			
OUT = Output to:				
Fixed Port	1110011w	port		
Variable Port	1110111w			
XLAT = Translate Byte to AL	11010111			
LEA = Load EA to Register	10001101	mod reg r/m		
LDS = Load Pointer to DS	11000101	mod reg r/m		
LES = Load Pointer to ES	11000100	mod reg r/m		
LAHF = Load AH with Flags	10011111			
SAHF = Store AH into Flags	10011110			
PUSHF = Push Flags	10011100			
POPF = Pop Flags	10011101			

80C86AL/80C88AL INSTRUCTION SET SUMMARY (Continued)

Mnemonic and Description	Instruction Code			
ARITHMETIC	76543210	76543210	76543210	76543210
ADD = Add:				
Reg./Memory with Register to Either	000000dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod 000 r/m	data	data if s:w = 01
Immediate to Accumulator	0000010w	data	data if w = 1	
ADC = Add with Carry:				
Reg./Memory with Register to Either	000100dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod 010 r/m	data	data if s:w = 01
Immediate to Accumulator	0001010w	data	data if w = 1	
INC = Increment:				
Register/Memory	1111111w	mod 000 r/m		
Register	01000reg			
AAA = ASCII Adjust for Add	00110111			
DAA = Decimal Adjust for Add	00100111			
SUB = Subtract:				
Reg./Memory and Register to Either	001010dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod 101 r/m	data	data if s:w = 01
Immediate from Accumulator	0010110w	data	data if w = 1	
SBB = Subtract with Borrow				
Reg./Memory and Register to Either	000110dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod 011 r/m	data	data if s:w = 01
Immediate from Accumulator	0001110w	data	data if w = 1	
DEC = Decrement:				
Register/Memory	1111111w	mod 001 r/m		
Register	01001 reg			
NEG = Change Sign	1111011w	mod 011 r/m		
CMP = Compare:				
Register/Memory and Register	001110dw	mod reg r/m		
Immediate with Register/Memory	100000sw	mod 111 r/m	data	data if s:w = 01
Immediate with Accumulator	0011110w	data	data if w = 1	
AAS = ASCII Adjust for Subtract	00111111			
DAS = Decimal Adjust for Subtract	00101111			
MUL = Multiply (Unsigned)	1111011w	mod 100 r/m		
IMUL = Integer Multiply (Signed)	1111011w	mod 101 r/m		
AAM = ASCII Adjust for Multiply	11010100	00001010		
DIV = Divide (Unsigned)	1111011w	mod 110 r/m		
IDIV = Integer Divide (Signed)	1111011w	mod 111 r/m		
AAD = ASCII Adjust for Divide	11010101	00001010		
CBW = Convert Byte to Word	10011000			
CWD = Convert Word to Double Word	10011001			

80C86AL/80C88AL INSTRUCTION SET SUMMARY (Continued)

Mnemonic and Description	Instruction Code			
LOGIC	76543210	76543210	76543210	76543210
NOT = Invert	1111011w	mod 010 r/m		
SHL/SAL = Shift Logical/Arithmetic Left	110100vw	mod 100 r/m		
SHR = Shift Logical Right	110100vw	mod 101 r/m		
SAR = Shift Arithmetic Right	110100vw	mod 111 r/m		
ROL = Rotate Left	110100vw	mod 000 r/m		
ROR = Rotate Right	110100vw	mod 001 r/m		
RCL = Rotate Through Carry Flag Left	110100vw	mod 010 r/m		
RCR = Rotate Through Carry Right	110100vw	mod 011 r/m		
AND = **And:**				
Reg./Memory and Register to Either	001000dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 100 r/m	data	data if w = 1
Immediate to Accumulator	0010010w	data	data if w = 1	
TEST = **And Function to Flags, No Result:**				
Register/Memory and Register	1000010w	mod reg r/m		
Immediate Data and Register/Memory	1111011w	mod 000 r/m	data	data if w = 1
Immediate Data and Accumulator	1010100w	data	data if w = 1	
OR = **Or:**				
Reg./Memory and Register to Either	000010dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 001 r/m	data	data if w = 1
Immediate to Accumulator	0000110w	data	data if w = 1	
XOR = **Exclusive or:**				
Reg./Memory and Register to Either	001100dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 110 r/m	data	data if w = 1
Immediate to Accumulator	0011010w	data	data if w = 1	
STRING MANIPULATION				
REP = Repeat	1111001z			
MOVS = Move Byte/Word	1010010w			
CMPS = Compare Byte/Word	1010011w			
SCAS = Scan Byte/Word	1010111w			
LODS = Load Byte/Wd to AL/AX	1010110w			
STOS = Stor Byte/Wd from AL/A	1010101w			
CONTROL TRANSFER				
CALL = **Call:**				
Direct Within Segment	11101000	disp-low	disp-high	
Indirect Within Segment	11111111	mod 010 r/m		
Direct Intersegment	10011010	offset-low	offset-high	
		seg-low	seg-high	
Indirect Intersegment	11111111	mod 011 r/m		

80C86AL/80C88AL INSTRUCTION SET SUMMARY (Continued)

Mnemonic and Description	Instruction Code		
JMP = Unconditional Jump:	76543210	76543210	76543210
Direct Within Segment	11101001	disp-low	disp-high
Direct Within Segment-Short	11101011	disp	
Indirect Within Segment	11111111	mod 100 r/m	
Direct Intersegment	11101010	offset-low	offset-high
		seg-low	seg-high
Indirect Intersegment	11111111	mod 101 r/m	
RET = Return from CALL:			
Within Segment	11000011		
Within Seg Adding Immed to SP	11000010	data-low	data-high
Intersegment	11001011		
Intersegment Adding Immediate to SP	11001010	data-low	data-high
JE/JZ = Jump on Equal/Zero	01110100	disp	
JL/JNGE = Jump on Less/Not Greater or Equal	01111100	disp	
JLE/JNG = Jump on Less or Equal/ Not Greater	01111110	disp	
JB/JNAE = Jump on Below/Not Above or Equal	01110010	disp	
JBE/JNA = Jump on Below or Equal/ Not Above	01110110	disp	
JP/JPE = Jump on Parity/Parity Even	01111010	disp	
JO = Jump on Overflow	01110000	disp	
JS = Jump on Sign	01111000	disp	
JNE/JNZ = Jump on Not Equal/Not Zero	01110101	disp	
JNL/JGE = Jump on Not Less/Greater or Equal	01111101	disp	
JNLE/JG = Jump on Not Less or Equal/ Greater	01111111	disp	
JNB/JAE = Jump on Not Below/Above or Equal	01110011	disp	
JNBE/JA = Jump on Not Below or Equal/Above	01110111	disp	
JNP/JPO = Jump on Not Par/Par Odd	01111011	disp	
JNO = Jump on Not Overflow	01110001	disp	
JNS = Jump on Not Sign	01111001	disp	
LOOP = Loop CX Times	11100010	disp	
LOOPZ/LOOPE = Loop While Zero/Equal	11100001	disp	
LOOPNZ/LOOPNE = Loop While Not Zero/Equal	11100000	disp	
JCXZ = Jump on CX Zero	11100011	disp	
INT = Interrupt			
Type Specified	11001101	type	
Type 3	11001100		
INTO = Interrupt on Overflow	11001110		
IRET = Interrupt Return	11001111		

80C86AL/80C88AL INSTRUCTION SET SUMMARY (Continued)

Mnemonic and Description	Instruction Code	
	76543210	76543210
PROCESSOR CONTROL		
CLC = Clear Carry	11111000	
CMC = Complement Carry	11110101	
STC = Set Carry	11111001	
CLD = Clear Direction	11111100	
STD = Set Direction	11111101	
CLI = Clear Interrupt	11111010	
STI = Set Interrupt	11111011	
HLT = Halt	11110100	
WAIT = Wait	10011011	
ESC = Escape (to External Device)	11011xxx	mod x x x r/m
LOCK = Bus Lock Prefix	11110000	

NOTES:
AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value.
Greater = more positive;
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
if w = 1 then word instruction; if w = 0 then byte instruction
if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16 bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low
if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)
*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.
**MOV CS, REG/MEMORY not allowed.
if s:w = 01 then 16 bits of immediate data form the operand.
if s:w = 11 then an immediate data byte is sign extended to form the 16-bit operand.
if v = 0 then "count" = 1; if v = 1 then "count" in (CL)
x = don't care
z is used for string primitives for comparison with ZF FLAG.

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:
FLAGS =
X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

DATA SHEET REVISION REVIEW

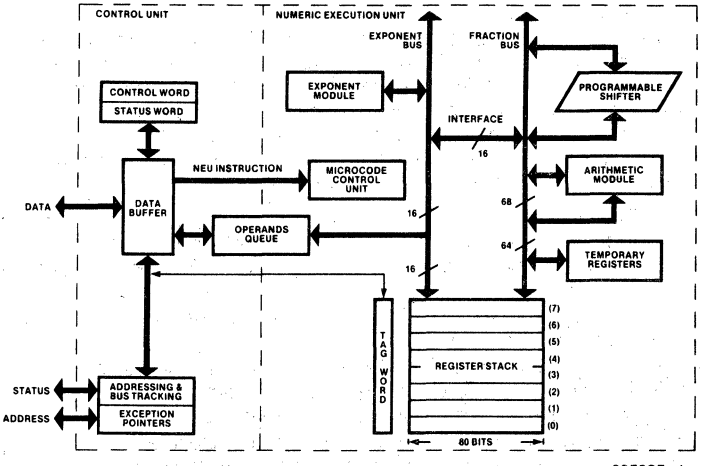
The following list represents key differences between this and the -001 data sheet. Please review this summary carefully.

1. In the Pin Description Table (Table 1), the description of the HLDA signal being issued has been corrected. HLDA will be issued in the middle of either T4 or Ti state.

8087
NUMERIC DATA COPROCESSOR
8087/8087-2/8087-1

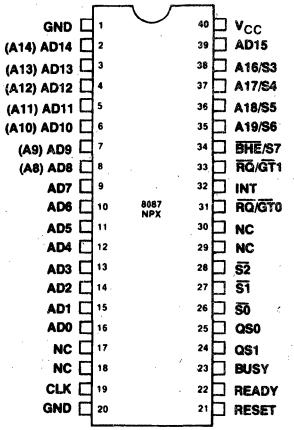
- **High Performance Numeric Data Coprocessor**
- **Adds Arithmetic, Trigonometric, Exponential, and Logarithmic Instructions to the Standard 8086/8088 and 80186/80188 Instruction Set for All Data Types**
- **CPU/8087 Supports 7 Data Types: 16-, 32-, 64-Bit Integers, 32-, 64-, 80-Bit Floating Point, and 18-Digit BCD Operands**
- **Compatible with IEEE Floating Point Standard 754**
- **Available in 5 MHz (8087), 8 MHz (8087-2) and 10 MHz (8087-1): 8 MHz 80186/80188 System Operation Supported with the 8087-1**
- **Adds 8 x 80-Bit Individually Addressable Register Stack to the 8086/8088 and 80186/80188 Architecture**
- **7 Built-In Exception Handling Functions**
- **MULTIBUS® System Compatible Interface**

The 8087 Numeric Data Coprocessor provides the instructions and data types needed for high performance numeric applications, providing up to 100 times the performance of a CPU alone. The 8087 is implemented in N-channel, depletion load, silicon gate technology (HMOS III), housed in a 40-pin package. Sixty-eight numeric processing instructions are added to the 8086/8088, 80186/80188 instruction sets and eight 80-bit registers are added to the register set. The 8087 is compatible with the IEEE Floating Point Standard 754.



205835-1

Figure 1. 8087 Block Diagram



205835-2

Figure 2. 8087 Pin Configuration

October 1986
Order Number: 205835-006

Table 1. 8087 Pin Description

Symbol	Type	Name and Function
AD15–AD0	I/O	**ADDRESS DATA:** These lines constitute the time multiplexed memory address (T_1) and data (T_2, T_3, T_W, T_4) bus. A0 is analogous to the $\overline{BHE}$ for the lower byte of the data bus, pins D7–D0. It is LOW during T_1 when a byte is to be transferred on the lower portion of the bus in memory operations. Eight-bit oriented devices tied to the lower half of the bus would normally use A0 to condition chip select functions. These lines are active HIGH. They are input/output lines for 8087-driven bus cycles and are inputs which the 8087 monitors when the CPU is in control of the bus. A15–A8 do not require an address latch in an 8088/8087 or 80188/8087. The 8087 will supply an address for the T_1–T_4 period.
A19/S6, A18/S5, A17/S4, A16/S3	I/O	**ADDRESS MEMORY:** During T_1 these are the four most significant address lines for memory operations. During memory operations, status information is available on these lines during T_2, T_3, T_W, and T_4. For 8087-controlled bus cycles, S6, S4, and S3 are reserved and currently one (HIGH), while S5 is always LOW. These lines are inputs which the 8087 monitors when the CPU is in control of the bus.
BHE/S7	I/O	**BUS HIGH ENABLE:** During T_1 the bus high enable signed ($\overline{BHE}$) should be used to enable data onto the most significant half of the data bus, pins D15–D8. Eight-bit-oriented devices tied to the upper half of the bus would normally use $\overline{BHE}$ to condition chip select functions. $\overline{BHE}$ is LOW during T_1 for read and write cycles when a byte is to be transferred on the high portion of the bus. The S7 status information is available during T_2, T_3, T_W, and T_4. The signal is active LOW. S7 is an input which the 8087 monitors during the CPU-controlled bus cycles.
$\overline{S2}$, $\overline{S1}$, $\overline{S0}$	I/O	**STATUS:** For 8087-driven, these status lines are encoded as follows: $\overline{S2}$ $\overline{S1}$ $\overline{S0}$ 0 (LOW) X X Unused 1 (HIGH) 0 0 Unused 1 0 1 Read Memory 1 1 0 Write Memory 1 1 1 Passive Status is driven active during T_4, remains valid during T_1 and T_2, and is returned to the passive state (1, 1, 1) during T_3 or during T_W when READY is HIGH. This status is used by the 8288 Bus Controller (or the 82188 Integrated Bus Controller with an 80186/80188 CPU) to generate all memory access control signals. Any change in $\overline{S2}$, $\overline{S1}$, or $\overline{S0}$ during T_4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T_3 or T_W is used to indicate the end of a bus cycle. These signals are monitored by the 8087 when the CPU is in control of the bus.
$\overline{RQ}/\overline{GT0}$	I/O	**REQUEST/GRANT:** This request/grant pin is used by the 8087 to gain control of the local bus from the CPU for operand transfers or on behalf of another bus master. It must be connected to one of the two processor request/grant pins. The request/grant sequence on this pin is as follows: 1. A pulse one clock wide is passed to the CPU to indicate a local bus request by either the 8087 or the master connected to the 8087 $\overline{RQ}/\overline{GT1}$ pin. 2. The 8087 waits for the grant pulse and when it is received will either initiate bus transfer activity in the clock cycle following the grant or pass the grant out on the $\overline{RQ}/\overline{GT1}$ pin in this clock if the initial request was for another bus master. 3. The 8087 will generate a release pulse to the CPU one clock cycle after the completion of the last 8087 bus cycle or on receipt of the release pulse from the bus master on $\overline{RQ}/\overline{GT1}$. For 80186/80188 systems the same sequence applies except $\overline{RQ}/\overline{GT}$ signals are converted to appropriate HOLD, HLDA signals by the 82188 Integrated Bus Controller. This is to conform with 80186/80188's HOLD, HLDA bus exchange protocol. Refer to the 82188 data sheet for further information.

Table 1. 8087 Pin Description (Continued)

Symbol	Type	Name and Function
$\overline{RQ}/\overline{GT}1$	I/O	**REQUEST/GRANT:** This request/grant pin is used by another local bus master to force the 8087 to request the local bus. If the 8087 is not in control of the bus when the request is made the request/grant sequence is passed through the 8087 on the $\overline{RQ}/\overline{GT}0$ pin one cycle later. Subsequent grant and release pulses are also passed through the 8087 with a two and one clock delay, respectively, for resynchronization. $\overline{RQ}/\overline{GT}1$ has an internal pullup resistor, and so may be left unconnected. If the 8087 has control of the bus the request/grant sequence is as follows: 1. A pulse 1 CLK wide from another local bus master indicates a local bus request to the 8087 (pulse 1). 2. During the 8087's next T_4 or T_1 a pulse 1 CLK wide from the 8087 to the requesting master (pulse 2) indicates that the 8087 has allowed the local bus to float and that it will enter the "RQ/GT acknowledge" state at the next CLK. The 8087's control unit is disconnected logically from the local bus during "RQ/GT acknowledge." 3. A pulse 1 CLK wide from the requesting master indicates to the 8087 (pulse 3) that the "RQ/GT" request is about to end and that the 8087 can reclaim the local bus at the next CLK. Each master-master exchange of the local bus is a sequence of 3 pulses. There must be one dead CLK cycle after each bus exchange. Pulses are active LOW. For 80186/80188 system, the $\overline{RQ}/\overline{GT}1$ line may be connected to the 82188 Integrated Bus Controller. In this case, a third processor with a HOLD, HLDA bus exchange system may acquire the bus from the 8087. For this configuration, $\overline{RQ}/\overline{GT}1$ will only be used if the 8087 is the bus master. Refer to 82188 data sheet for further information.
QS1, QS0	I	**QS1, QS0:** QS1 and QS0 provide the 8087 with status to allow tracking of the CPU instruction queue. **QS1** **QS0** 0 (LOW) 0 No Operation 0 1 First Byte of Op Code from Queue 1 (HIGH) 0 Empty the Queue 1 1 Subsequent Byte from Queue
INT	O	**INTERRUPT:** This line is used to indicate that an unmasked exception has occurred during numeric instruction execution when 8087 interrupts are enabled. This signal is typically routed to an 8259A for 8086/8088 systems and to INT0 for 80186/80188 systems. INT is active HIGH.
BUSY	O	**BUSY:** This signal indicates that the 8087 NEU is executing a numeric instruction. It is connected to the CPU's $\overline{TEST}$ pin to provide synchronization. In the case of an unmasked exception BUSY remains active until the exception is cleared. BUSY is active HIGH.
READY	I	**READY:** READY is the acknowledgement from the addressed memory device that it will complete the data transfer. The RDY signal from memory is synchronized by the 8284A Clock Generator to form READY for 8086 systems. For 80186/80188 systems, RDY is synchronized by the 82188 Integrated Bus Controller to form READY. This signal is active HIGH.
RESET	I	**RESET:** RESET causes the processor to immediately terminate its present activity. The signal must be active HIGH for at least four clock cycles. RESET is internally synchronized.
CLK	I	**CLOCK:** The clock provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing.
V_{CC}		**POWER:** V_{CC} is the +5V power supply pin.
GND		**GROUND:** GND are the ground pins.

NOTE:
For the pin descriptions of the 8086, 8088, 80186 and 80188 CPUs, reference the respective data sheets (8086, 8088, 80186, 80188).

APPLICATION AREAS

The 8087 provides functions meant specifically for high performance numeric processing requirements. Trigonometric, logarithmic, and exponential functions are built into the coprocessor hardware. These functions are essential in scientific, engineering, navigational, or military applications.

The 8087 also has capabilities meant for business or commercial computing. An 8087 can process Binary Coded Decimal (BCD) numbers up to 18 digits without roundoff errors. It can also perform arithmetic on integers as large as 64 bits $\pm 10^{18}$).

PROGRAMMING LANGUAGE SUPPORT

Programs for the 8087 can be written in Intel's high-level languages for 8086/8088 and 80186/80188 Systems; ASM-86 (the 8086, 8088 assembly language), PL/M-86, FORTRAN-86, and PASCAL-86.

RELATED INFORMATION

For 8086, 8088, 80186 or 80188 details, refer to the respective data sheets. For 80186 or 80188 systems, also refer to the 82188 Integrated Bus Controller data sheet.

FUNCTIONAL DESCRIPTION

The 8087 Numeric Data Processor's architecture is designed for high performance numeric computing in conjunction with general purpose processing.

The 8087 is a numeric processor extension that provides arithmetic and logical instruction support for a variety of numeric data types. It also executes numerous built-in transcendental functions (e.g., tangent and log functions). The 8087 executes instructions as a coprocessor to a maximum mode CPU. It effectively extends the register and instruction set of the system and adds several new data types as well. Figure 3 presents the registers of the CPU + 8087. Table 2 shows the range of data types supported by the 8087. The 8087 is treated as an extension to the CPU, providing register, data types, control, and instruction capabilities at the hardware level. At the programmer's level the CPU and the 8087 are viewed as a single unified processor.

System Configuration

As a coprocessor to an 8086 or 8088, the 8087 is wired in parallel with the CPU as shown in Figure 4. Figure 5 shows the 80186/80188 system configuration. The CPU's status (S0–S2) and queue status lines (QS0–QS1) enable the 8087 to monitor and decode instructions in synchronization with the CPU and without any CPU overhead. For 80186/80188 systems, the queue status signals of the 80186/80188 are synchronized to 8087 requirements by the 8288 Integrated Bus Controller. Once started, the 8087 can process in parallel with, and independent of, the host CPU. For resynchronization, the 8087's BUSY signal informs the CPU that the 8087 is executing an instruction and the CPU WAIT instruction tests this signal to insure that the 8087 is ready to execute subsequent instructions. The 8087 can interrupt the CPU when it detects an error or exception. The 8087's interrupt request line is typically routed to the CPU through an 8259A Programmable Interrupt Controller for 8086, 8088 systems and INT0 for 80186/80188.

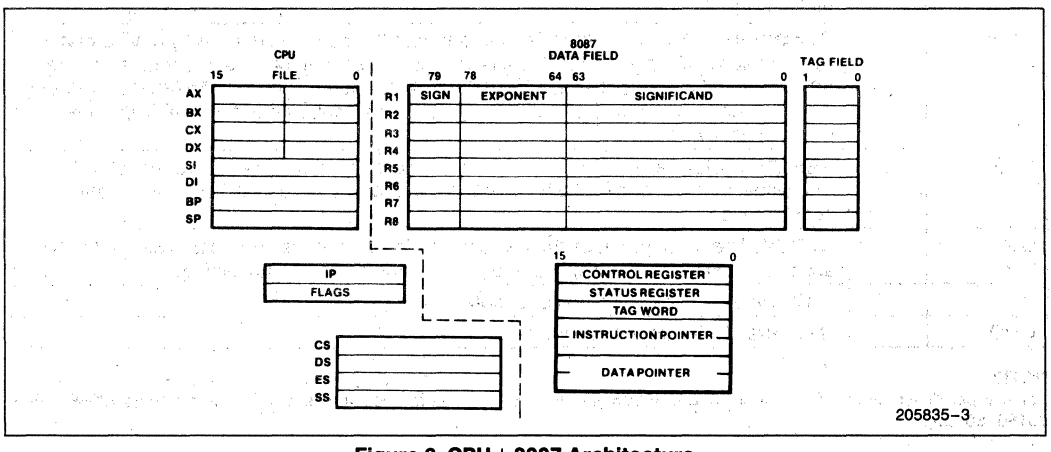


Figure 3. CPU + 8087 Architecture

The 8087 uses one of the request/grant lines of the 8086/8088 architecture (typically $\overline{RQ}/\overline{GT}0$) to obtain control of the local bus for data transfers. The other request/grant line is available for general system use (for instance by an I/O processor in LOCAL mode). A bus master can also be connected to the 8087's $\overline{RQ}/\overline{GT}1$ line. In this configuration the 8087 will pass the request/grant handshake signals between the CPU and the attached master when the 8087 is not in control of the bus and will relinquish the bus to the master directly when the 8087 is in control. In this way two additional masters can be configured in an 8086/8088 system; one will share the 8086/8088 bus with the 8087 on a first-come first-served basis, and the second will be guaranteed to be higher in priority than the 8087.

For 80186/80188 systems, $\overline{RQ}/\overline{GT}0$ and $\overline{RQ}/\overline{GT}1$ are connected to the corresponding inputs of the 82188 Integrated Bus Controller. Because the 80186/80188 has a HOLD, HLDA bus exchange protocol, an interface is needed which will translate $\overline{RQ}/\overline{GT}$ signals to corresponding HOLD, HLDA signals and vice versa. One of the functions of the 82188 IBC is to provide this translation. $\overline{RQ}/\overline{GT}0$ is translated to HOLD, HLDA signals which are then directly connected to the 80186/80188. The $\overline{RQ}/\overline{GT}1$ line is also translated into HOLD, HLDA signals (referred to as SYSHOLD, SYSHLDA signals) by the 82188 IBC. This allows a third processor (using a HOLD, HLDA bus exchange protocol) to gain control of the bus.

Unlike an 8086/8087 system, $\overline{RQ}/\overline{GT}$ is only used when the 8087 has bus control. If the third processor requests the bus when the current bus master is the 80186/80188, the 82188 IBC will directly pass the request onto the 80186/80188 without going through the 8087. The third processor has the highest bus priority in the system. If the 8087 requests the bus while the third processor has bus control, the grant pulse will not be issued until the third processor releases the bus (using SYSHOLD). In this configuration, the third processor has the highest priority, the 8087 has the next highest, and the 80186/80188 has the lowest bus priority.

Bus Operation

The 8087 bus structure, operation and timing are identical to all other processors in the 8086/8088 series (maximum mode configuration). The address is time multiplexed with the data on the first 16/8 lines of the address/data bus. A16 through A19 are time multiplexed with four status lines S3–S6. S3, S4 and S6 are always one (HIGH) for 8087-driven bus cycles while S5 is always zero (LOW). When the 8087 is monitoring CPU bus cycles (passive mode) S6 is also monitored by the 8087 to differentiate 8086/8088 activity from that of a local I/O processor or any other local bus master. (The 8086/8088 must be the only processor on the local bus to drive S6 LOW). S7 is multiplexed with and has the same value as $\overline{BHE}$ for all 8087 bus cycles.

Table 2. 8087 Data Types

Data Formats	Range	Precision	Most Significant Byte (7 0 7 0 7 0 7 0 7 0 7 0 7 0 7 0 7 0)
Word Integer	10^4	16 Bits	I_{15} … I_0 Two's Complement
Short Integer	10^9	32 Bits	I_{31} … I_0 Two's Complement
Long Integer	10^{18}	64 Bits	I_{63} … I_0 Two's Complement
Packed BCD	10^{18}	18 Digits	S — $D_{17}D_{16}$ … D_1 D_0
Short Real	$10^{\pm 38}$	24 Bits	S E_7 … E_0 F_1 … F_{23} F_0 Implicit
Long Real	$10^{\pm 308}$	53 Bits	S E_{10} … E_0 F_1 … F_{52} F_0 Implicit
Temporary Real	$10^{\pm 4932}$	64 Bits	S E_{14} … E_0 F_0 … F_{63}

Integer: I
Packed BCD: $(-1)^S(D_{17}...D_0)$
Real: $(-1)^S(2^{E-Bias})(F_0 \bullet F_1...)$
bias = 127 for Short Real
1023 for Long Real
16383 for Temp Real

The first three status lines, $\overline{S0}$–$\overline{S2}$, are used with an 8288 bus controller or 82188 Integrated Bus Controller to determine the type of bus cycle being run:

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	
0	X	X	Unused
1	0	0	Unused
1	0	1	Memory Data Read
1	1	0	Memory Data Write
1	1	1	Passive (no bus cycle)

Programming Interface

The 8087 includes the standard 8086, 8088 instruction set for general data manipulation and program control. It also includes 68 numeric instructions for extended precision integer, floating point, trigonometric, logarithmic, and exponential functions. Sample execution times for several 8087 functions are shown in Table 3. Overall performance is up to 100 times that of an 8086 processor for numeric instructions.

Any instruction executed by the 8087 is the combined result of the CPU and 8087 activity. The CPU and the 8087 have specialized functions and registers providing fast concurrent operation. The CPU controls overall program execution while the 8087 uses the coprocessor interface to recognize and perform numeric operations.

Table 2 lists the seven data types the 8087 supports and presents the format for each type. Internally, the 8087 holds all numbers in the temporary real format. Load and store instructions automatically convert operands represented in memory as 16-, 32-, or 64-bit integers, 32- or 64-bit floating point numbers or 18-digit packed BCD numbers into temporary real format and vice versa. The 8087 also provides the capability to control round off, underflow, and overflow errors in each calculation.

Computations in the 8087 use the processor's register stack. These eight 80-bit registers provide the equivalent capacity of 20 32-bit registers. The 8087 register set can be accessed as a stack, with instructions operating on the top one or two stack elements, or as a fixed register set, with instructions operating on explicitly designated registers.

Table 5 lists the 8087's instructions by class. All appear as ESCAPE instructions to the host. Assembly language programs are written in ASM-86, the 8086, 8088 assembly language.

Table 3. Execution Times for Selected 8086/8087 Numeric Instructions and Corresponding 8086 Emulation

Floating Point Instruction	Approximate Execution Time (μs)	
	8086/8087 (8 MHz Clock)	8086 Emulation
Add/Subtract	10.6	1000
Multiply (Single Precision)	11.9	1000
Multiply (Extended Precision)	16.9	1312
Divide	24.4	2000
Compare	–5.6	812
Load (Double Precision)	–6.3	1062
Store (Double Precision)	13.1	750
Square Root	22.5	12250
Tangent	56.3	8125
Exponentiation	62.5	10687

NUMERIC PROCESSOR EXTENSION ARCHITECTURE

As shown in Figure 1, the 8087 is internally divided into two processing elements, the control unit (CU) and the numeric execution unit (NEU). The NEU executes all numeric instructions, while the CU receives and decodes instructions, reads and writes memory operands and executes 8087 control instructions. The two elements are able to operate independently of one another, allowing the CU to maintain synchronization with the CPU while the NEU is busy processing a numeric instruction.

Control Unit

The CU keeps the 8087 operating in synchronization with its host CPU. 8087 instructions are intermixed with CPU instructions in a single instruction stream. The CPU fetches all instructions from memory; by monitoring the status ($\overline{S0}$–$\overline{S2}$, S6) emitted by the CPU, the control unit determines when an instruction is being fetched. The CPU monitors the data bus in parallel with the CPU to obtain instructions that pertain to the 8087.

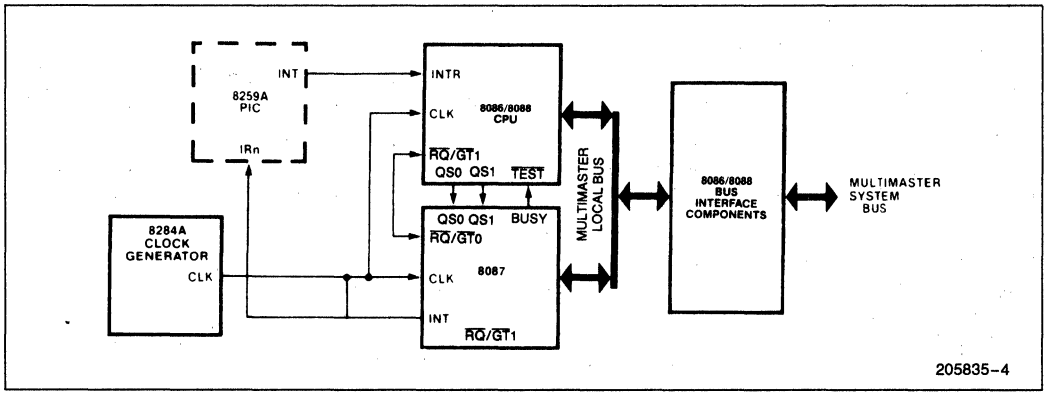


Figure 4. 8086/8087, 8088/8087 System Configuration

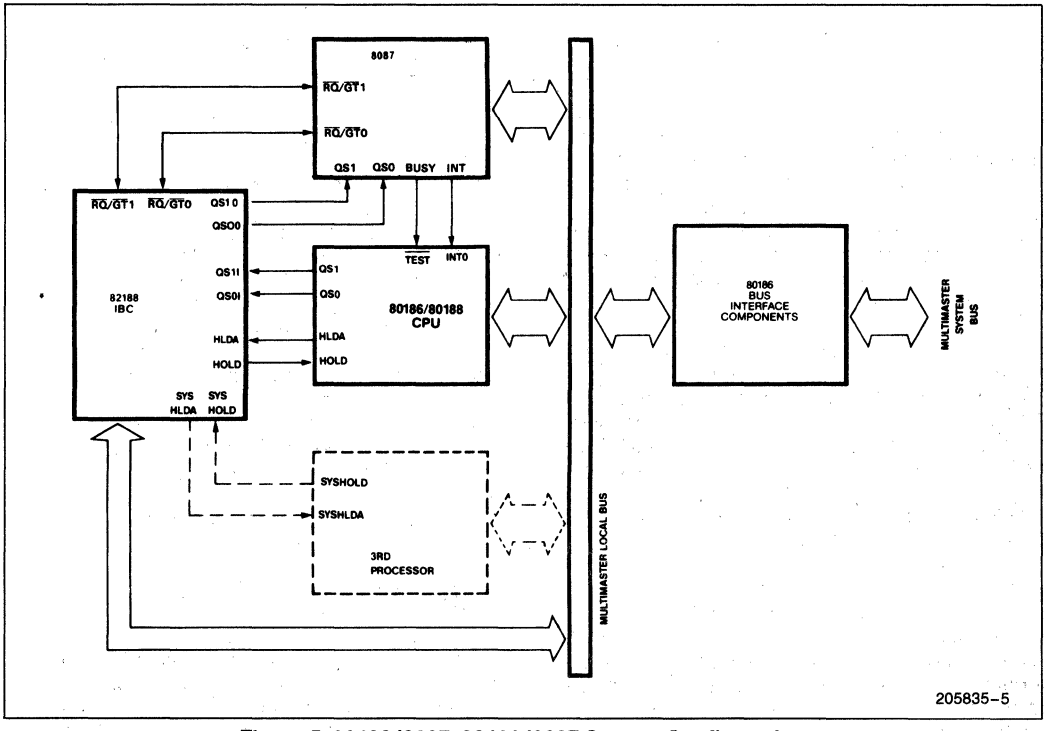


Figure 5. 80186/8087, 80188/8087 System Configuration

The CU maintains an instruction queue that is identical to the queue in the host CPU. The CU automatically determines if the CPU is an 8086/80186 or an 8088/80188 immediately after reset (by monitoring the $\overline{BHE}$/S7 line) and matches its queue length accordingly. By monitoring the CPU's queue status lines (QS0, QS1), the CU obtains and decodes instructions from the queue in synchronization with the CPU.

A numeric instruction appears as an ESCAPE instruction to the CPU. Both the CPU and 8087 decode and execute the ESCAPE instruction together. The 8087 only recognizes the numeric instructions shown in Table 5. The start of a numeric operation is accomplished when the CPU executes the ESCAPE instruction. The instruction may or may not identify a memory operand.

The CPU does, however, distinguish between ESC instructions that reference memory and those that do not. If the instruction refers to a memory operand, the CPU calculates the operand's address using any one of its available addressing modes, and then performs a "dummy read" of the word at that location. (Any location within the 1M byte address space is allowed.) This is a normal read cycle except that the CPU ignores the data it receives. If the ESC instruction does not contain a memory reference (e.g. an 8087 stack operation), the CPU simply proceeds to the next instruction.

An 8087 instruction can have one of three memory reference options: (1) not reference memory; (2) load an operand word from memory into the 8087; or (3) store an operand word from the 8087 into memory. If no memory reference is required, the 8087 simply executes its instruction. If a memory reference is required, the CU uses a "dummy read" cycle initiated by the CPU to capture and save the address that the CPU places on the bus. If the instruction is a load, the CU additionally captures the data word when it becomes available on the local data bus. If data required is longer than one word, the CU immediately obtains the bus from the CPU using the request/grant protocol and reads the rest of the information in consecutive bus cycles. In a store operation, the CU captures and saves the store address as in a load, and ignores the data word that follows in the "dummy read" cycle. When the 8087 is ready to perform the store, the CU obtains the bus from the CPU and writes the operand starting at the specified address.

Numeric Execution Unit

The NEU executes all instructions that involve the register stack; these include arithmetic, logical, transcendental, constant and data transfer instructions. The data path in the NEU is 84 bits wide (68 fractions bits, 15 exponent bits and a sign bit) which allows internal operand transfers to be performed at very high speeds.

When the NEU begins executing an instruction, it activates the 8087 BUSY signal. This signal can be used in conjunction with the CPU WAIT instruction to resynchronize both processors when the NEU has completed its current instruction.

Register Set

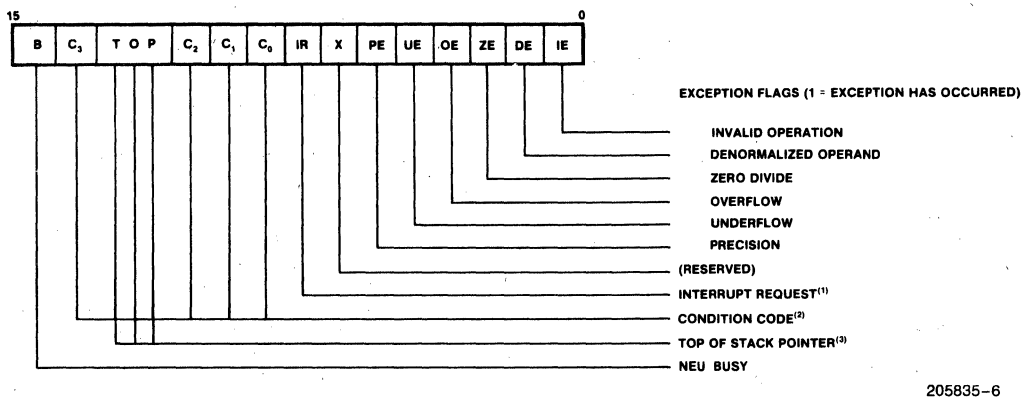
The CPU+8087 register set is shown in Figure 3. Each of the eight data registers in the 8087's register stack is 80 bits and is divided into "fields" corresponding to the 8087's temporary real data type.

At a given point in time the TOP field in the control word identifies the current top-of-stack register. A "push" operation decrements TOP by 1 and loads a value into the new top register. A "pop" operation stores the value from the current top register and then increments TOP by 1. Like CPU stacks in memory, the 8087 register stack grows "down" toward lower-addressed registers.

Instructions may address the data registers either implicitly or explicitly. Many instructions operate on the register at the top of the stack. These instructions implicitly address the register pointed to by the TOP. Other instructions allow the programmer to explicitly specify the register which is to be used. Explicit register addressing is "top-relative."

Status Word

The status word shown in Figure 6 reflects the overall state of the 8087; it may be stored in memory and then inspected by CPU code. The status word is a 16-bit register divided into fields as shown in Figure 6. The busy bit (bit 15) indicates whether the NEU is either executing an instruction or has an interrupt request pending (B=1), or is idle (B=0). Several instructions which store and manipulate the status word are executed exclusively by the CU, and these do not set the busy bit themselves.



NOTES:
1. IR is set if any unmasked exception bit is set, cleared otherwise.
2. See Table 3 for condition code interpretation.
3. Top Values:
   000 = Register 0 is Top of Stack.
   001 = Register 1 is Top of Stack.
   •
   •
   •
   111 = Register 7 is Top of Stack.

Figure 6. 8087 Status Word

The four numeric condition code bits (C_0–C_3) are similar to flags in a CPU: various instructions update these bits to reflect the outcome of the 8087 operations. The effect of these instructions on the condition code bits is summarized in Table 4.

Bits 14–12 of the status word point to the 8087 register that is the current top-of-stack (TOP) as described above.

Bit 7 is the interrupt request bit. This bit is set if any unmasked exception bit is set and cleared otherwise.

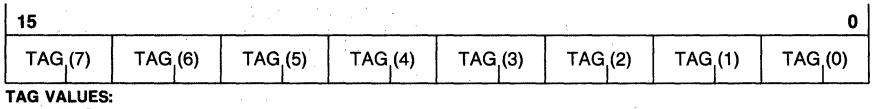
Bits 5–0 are set to indicate that the NEU has detected an exception while executing an instruction.

Tag Word

The tag word marks the content of each register as shown in Figure 7. The principal function of the tag word is to optimize the 8087's performance. The tag word can be used, however, to interpret the contents of 8087 registers.

Instruction and Data Pointers

The instruction and data pointers (see Figure 8) are provided for user-written error handlers. Whenever the 8087 executes a math instruction, the CU saves the instruction address, the operand address (if present) and the instruction opcode. 8087 instructions can store this data into memory



TAG VALUES:
00 = VALID
01 = ZERO
10 = SPECIAL
11 = EMPTY

Figure 7. 8087 Tag Word

Table 4a. Condition Code Interpretation

Instruction Type	C_3	C_2	C_1	C_0	Interpretation
Compare, Test	0	0	X	0	ST > Source or 0 (FTST)
	0	0	X	1	ST < Source or 0 (FTST)
	1	0	X	0	ST = Source or 0 (FTST)
	1	1	X	1	ST is not comparable
Remainder	Q_1	0	Q_0	Q_2	Complete reduction with three low bits of quotient (See Table 4b)
	U	1	U	U	Incomplete Reduction
Examine	0	0	0	0	Valid, positive unnormalized
	0	0	0	1	Invalid, positive, exponent = 0
	0	0	1	0	Valid, negative, unnormalized
	0	0	1	1	Invalid, negative, exponent = 0
	0	1	0	0	Valid, positive, normalized
	0	1	0	1	Infinity, positive
	0	1	1	0	Valid, negative, normalized
	0	1	1	1	Infinity, negative
	1	0	0	0	Zero, positive
	1	0	0	1	Empty
	1	0	1	0	Zero, negative
	1	0	1	1	Empty
	1	1	0	0	Invalid, positive, exponent = 0
	1	1	0	1	Empty
	1	1	1	0	Invalid, negative, exponent = 0
	1	1	1	1	Empty

NOTES:
1. ST = Top of stack
2. X = value is not affected by instruction
3. U = value is undefined following instruction
4. Q_n = Quotient bit n

Table 4b. Condition Code Interpretation after FPREM Instruction As a Function of Divided Value

Dividend Range	Q_2	Q_1	Q_0
Dividend < 2 * Modulus	C_3[1]	C_1[1]	Q_0
Dividend < 4 * Modulus	C_3[1]	Q_1	Q_0
Dividend ≥ 4 * Modulus	Q_2	Q_1	Q_0

NOTE:
1. Previous value of indicated bit, not affected by FPREM instruction execution.

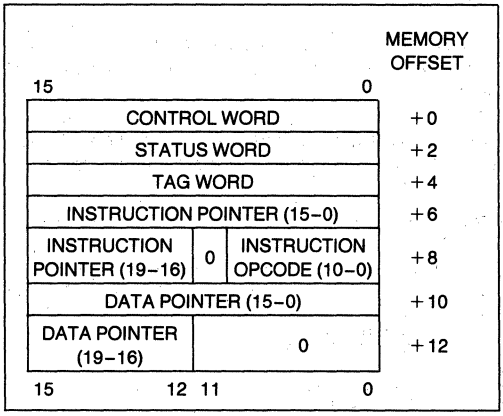


Figure 8. 8087 Instruction and Data Pointer Image in Memory

Control Word

The 8087 provides several processing options which are selected by loading a word from memory into the control word. Figure 9 shows the format and encoding of the fields in the control word.

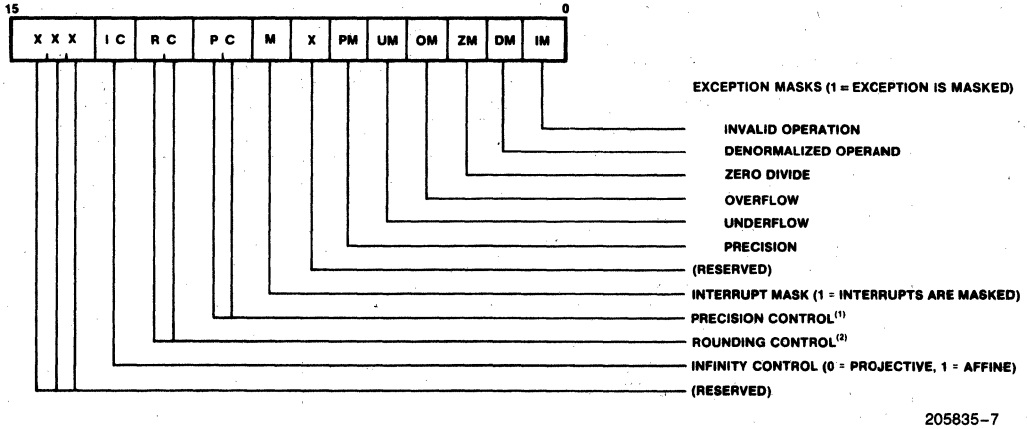
The low order byte of this control word configures 8087 interrupts and exception masking. Bits 5–0 of the control word contain individual masks for each of the six exceptions that the 8087 recognizes and bit 7 contains a general mask bit for all 8087 interrupts. The high order byte of the control word configures the 8087 operating mode including precision, rounding, and infinity controls. The precision control bits (bits 9–8) can be used to set the 8087 internal operating precision at less than the default of temporary real precision. This can be useful in providing compatibility with earlier generation arithmetic processors of smaller precision than the 8087. The rounding control bits (bits 11–10) provide for directed rounding and true chop as well as the unbiased round to nearest mode specified in the proposed IEEE standard. Control over closure of the number space at infinity is also provided (either affine closure, $\pm\infty$, or projective closure, ∞, is treated as unsigned, may be specified).

Exception Handling

The 8087 detects six different exception conditions that can occur during instruction execution. Any or all exceptions will cause an interrupt if unmasked and interrupts are enabled.

If interrupts are disabled the 8087 will simply continue execution regardless of whether the host clears the exception. If a specific exception class is masked and that exception occurs, however, the 8087 will post the exception in the status register and perform an on-chip default exception handling procedure, thereby allowing processing to continue. The exceptions that the 8087 detects are the following:

1. INVALID OPERATION: Stack overflow, stack underflow, indeterminate form (0/0, $\infty - \infty$, etc.) or the use of a Non-Number (NAN) as an operand. An exponent value is reserved and any bit pattern with this value in the exponent field is termed a Non-Number and causes this exception. If this exception is masked, the 8087's default response is to generate a specific NAN called INDEFINITE, or to propagate already existing NANs as the calculation result.



NOTES:

1. Precision Control
   00 = 24 bits
   01 = Reserved
   10 = 53 bits
   11 = 64 bits

2. Rounding Control
   00 = Round to Nearest or Even
   01 = Round Down (toward $-\infty$)
   10 = Round Up (toward $+\infty$)
   11 = Chop (truncate toward zero)

Figure 9. 8087 Control Word

2. OVERFLOW: The result is too large in magnitude to fit the specified format. The 8087 will generate an encoding for infinity if this exception is masked.
3. ZERO DIVISOR: The divisor is zero while the dividend is a non-infinite, non-zero number. Again, the 8087 will generate an encoding for infinity if this exception is masked.
4. UNDERFLOW: The result is non-zero but too small in magnitude to fit in the specified format. If this exception is masked the 8087 will denormalize (shift right) the fraction until the exponent is in range. This process is called gradual underflow.
5. DENORMALIZED OPERAND: At least one of the operands or the result is denormalized; it has the smallest exponent but a non-zero significand. Normal processing continues if this exception is masked off.
6. INEXACT RESULT: If the true result is not exactly representable in the specified format, the result is rounded according to the rounding mode, and this flag is set. If this exception is masked, processing will simply continue.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature .....−65°C to +150°C
Voltage on Any Pin with Respect to Ground .....−1.0V to +7V
Power Dissipation .....3.0 Watt

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±5%

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage (Note 8)		0.45	V	I_{OL} = 2.5 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −400 μA
I_{CC}	Power Supply Current		475	mA	T_A = 25°C
I_{LI}	Input Leakage Current		±10	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{LO}	Output Leakage Current		±10	μA	T_A = 25°C
V_{CL}	Clock Input Low Voltage	−0.5	0.6	V	
V_{CH}	Clock Input High Voltage	3.9	V_{CC} + 1.0	V	
C_{IN}	Capacitance of Inputs		10	pF	fc = 1 MHz
C_{IO}	Capacitance of I/O Buffer (AD0−15, A_{16}−A_{19}, BHE, S2−S0, RQ/GT) and CLK		15	pF	fc = 1 MHz
C_{OUT}	Capacitance of Outputs BUSY INT		10	pF	fc = 1 MHz

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±5%

TIMING REQUIREMENTS

Symbol	Parameter	8087		8087-2		8087-1 (Preliminary: Note 7)		Units	Test Conditions
		Min	Max	Min	Max	Min	Max		
TCLCL	CLK Cycle Period	200	500	125	500	100	500	ns	
TCLCH	CLK Low Time	118		68		53		ns	
TCHCL	CLK High Time	69		44		39		ns	
TCH1CH2	CLK Rise Time		10		10		15	ns	From 1.0V to 3.5V
TCL2CL2	CLK Fall Time		10		10		15	ns	From 3.5V to 1.0V
TDVCL	Data In Setup Time	30		20		15		ns	
TCLDX	Data In Hold Time	10		10		10		ns	
TRYHCH	READY Setup Time	118		68		53		ns	
TCHRYX	READY Hold Time	30		20		5		ns	
TRYLCL	READY Inactive to CLK (Note 6)	−8		−8		−10		ns	
TGVCH	RQ/GT Setup Time (Note 8)	30		15		15		ns	
TCHGX	RQ/GT Hold Time	40		30		20		ns	
TQVCL	QS0–1 Setup Time (Note 8)	30		30		30		ns	
TCLQX	QS0–1 Hold Time	10		10		5		ns	
TSACH	Status Active Setup Time	30		30		30		ns	
TSNCL	Status Inactive Setup Time	30		30		30		ns	
TILIH	Input Rise Time (Except CLK)		20		20		20	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK)		12		12		15	ns	From 2.0V to 0.8V

TIMING RESPONSES

Symbol	Parameter	8087		8087-2		8087-1 (Preliminary: Note 7)		Units	Test Conditions
		Min	Max	Min	Max	Min	Max		
TCLML	Command Active Delay (Notes 1, 2)	10/0	35/70	10/0	35/70	10/0	35/70	ns	C_L = 20–100 pF for all 8087 Outputs (in addition to 8087 self-load)
TCLMH	Command Inactive Delay (Notes 1, 2)	10/0	35/55	10/0	35/55	10/0	35/70	ns	
TRYHSH	Ready Active to Status Passive (Note 5)		110		65		45	ns	
TCHSV	Status Active Delay	10	110	10	60	10	45	ns	
TCLSH	Status Inactive Delay	10	130	10	70	10	55	ns	
TCLAV	Address Valid Delay	10	110	10	60	10	55	ns	
TCLAX	Address Hold Time	10		10		10		ns	

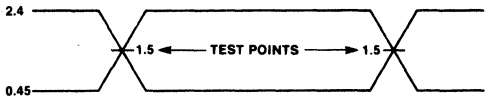
A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±5% (Continued)

TIMING RESPONSES (Continued)

Symbol	Parameter	8087		8087-2		8087-1 (Preliminary: Note 7)		Units	Test Conditions
		Min	Max	Min	Max	Min	Max		
TCLAZ	Address Float Delay	TCLAX	80	TCLAX	50	TCLAX	45	ns	C_L = 20–100 pF for all 8087 Outputs (in addition to 8087 self-load)
TSVLH	Status Valid to ALE High (Notes 1, 2)		15/30		15/30		15/30	ns	
TCLLH	CLK Low to ALE Valid (Notes 1, 2)		15/30		15/30		15/30	ns	
TCHLL	ALE Inactive Delay (Notes 1, 2)		15/30		15/30		15/30	ns	
TCLDV	Data Valid Delay	10	110	10	60	10	50	ns	
TCHDX	Status Hold Time	10		10		10	45	ns	
TCLDOX	Data Hold Time	10		10		10		ns	
TCVNV	Control Active Delay (Notes 1, 3)	5	45	5	45	5	45	ns	
TCVNX	Control Inactive Delay (Notes 1, 3)	10	45	10	45	10	45	ns	
TCHBV	BUSY and INT Valid Delay	10	150	10	85	10	65	ns	
TCHDTL	Direction Control Active Delay (Notes 1, 3)		50		50		50	ns	
TCHDTH	Direction Control Inactive Delay (Notes 1, 3)		30		30		30	ns	
TSVDTV	STATUS to DT/$\overline{R}$ Delay (Notes 1, 4)	0	30	0	30	0	30	ns	
TCLDTV	DT/$\overline{R}$ Active Delay (Notes 1, 4)	0	55	0	55	0	55	ns	
TCHDNV	$\overline{DEN}$ Active Delay (Notes 1, 4)	0	55	0	55	0	55	ns	
TCHDNX	$\overline{DEN}$ Inactive Delay (Notes 1, 4)	5	55	5	55	5	55	ns	
TCLGL	RQ/GT Active Delay (Note 8)	0	85	0	50	0	38	ns	C_L = 40 pF (in addition to 8087 self-load)
TCLGH	RQ/GT Inactive Delay	0	85	0	50	0	45	ns	
TOLOH	Output Rise Time		20		20		15	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time		12		12		12	ns	From 2.0V to 0.8V

NOTES:
1. Signal at 8284A, 8288, or 82188 shown for reference only.
2. 8288 timing/82188 timing.
3. 8288 timing.
4. 82188 timing.
5. Applies only to T_3 and wait states.
6. Applies only to T_2 state (8 ns into T_3).
7. IMPORTANT SYSTEM CONSIDERATION: Some 8087-1 timing parameters are constrained relative to the corresponding 8086-1 specifications. Therefore, 8086-1 systems incorporating the 8087-1 should be designed with the 8087-1 specifications.
8. Changes since last revision.

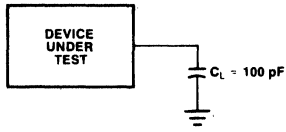
A.C. TESTING INPUT, OUTPUT WAVEFORM



205835-8

A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0".

A.C. TESTING LOAD CIRCUIT

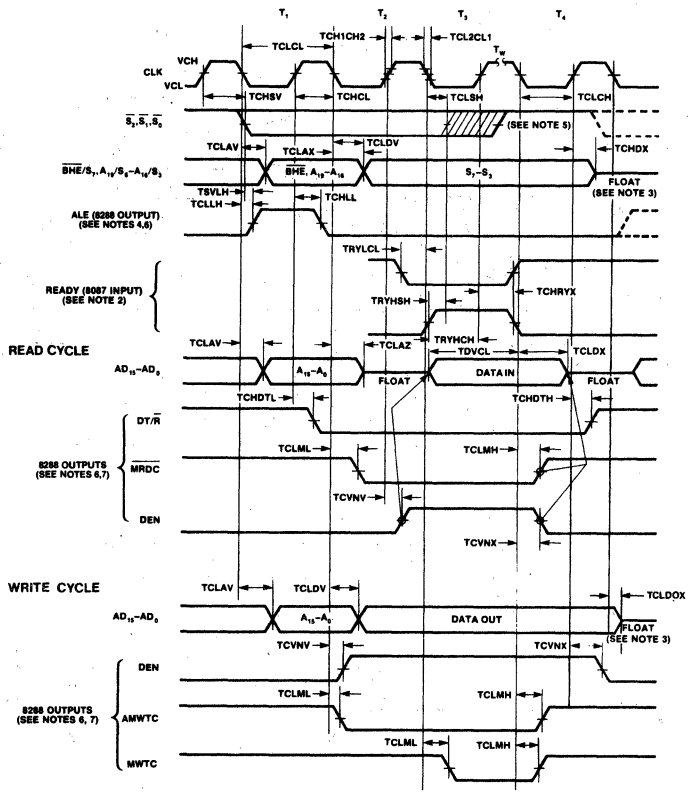


205835-9

C_L Includes Jig Capacitance

WAVEFORMS

MASTER MODE (with 8288 references)



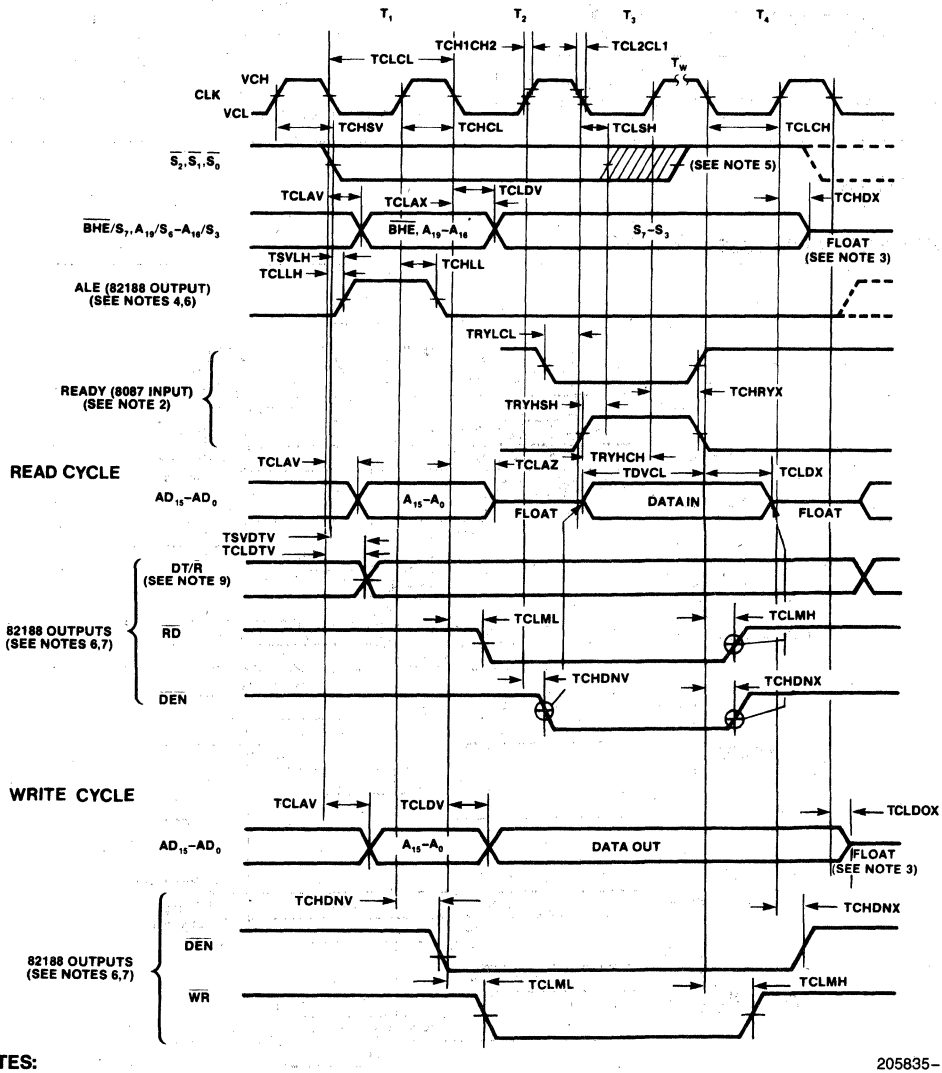
205835-10

NOTES:

1. All signals switch between V_{OL} and V_{OH} unless otherwise specified.
2. READY is sampled near the end of T_2, T_3 and T_W to determine if T_W machine states are to be inserted.
3. The local bus floats only if the 8087 is returning control to the 8086/8088.
4. ALE rises at later of (TSVLH, TCLLH).
5. Status inactive in state just prior to T_4.
6. Signals at 8284A or 8288 are shown for reference only.
7. The issuance of 8288 command and control signals ($\overline{MRDC}$, ($\overline{MWTC}$, $\overline{AMWC}$, and DEN) lags the active high 8288 CEN.
8. All timing measurements are made at 1.5V unless otherwise noted.

WAVEFORMS (Continued)

MASTER MODE (with 82188 references)



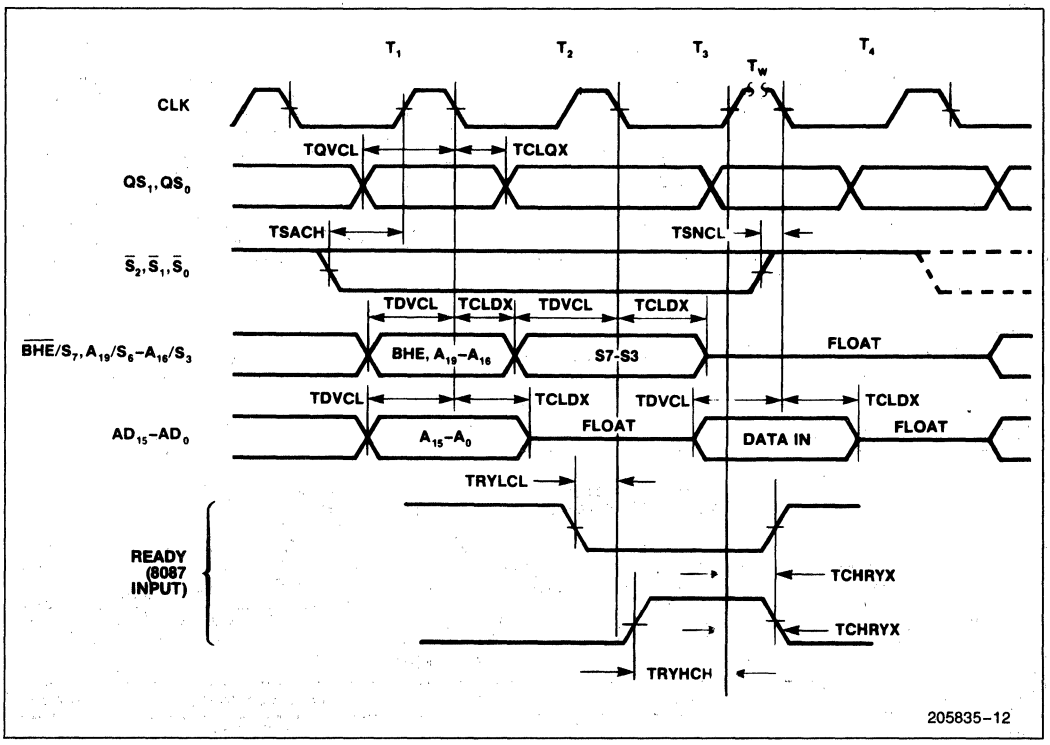
205835–11

NOTES:

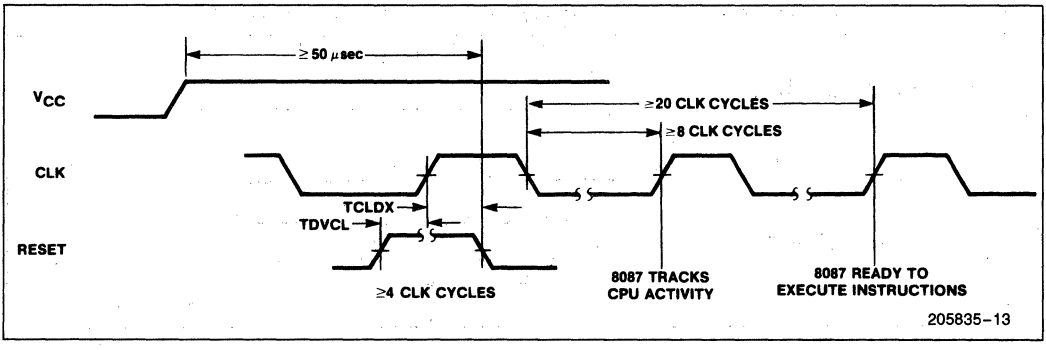
1. All signals switch between V_{OL} and V_{OH} unless otherwise specified.
2. READY is sampled near the end of T_2, T_3 and T_W to determine if T_W machine states are to be inserted.
3. The local bus floats only if the 8087 is returning control to the 80186/80188.
4. ALE rises at later of (TSVLH, TCLLH).
5. Status inactive in state just prior to T_4.
6. Signals at 8284A or 82188 are shown for reference only.
7. The issuance of 8288 command and control signals ($\overline{MRDC}$, ($\overline{MWTC}$, $\overline{AMWC}$, and DEN) lags the active high 8288 CEN.
8. All timing measurements are made at 1.5V unless otherwise noted.
9. DT/$\overline{R}$ becomes valid at the later of (TSVDTV, TCLDTV).

WAVEFORMS (Continued)

PASSIVE MODE

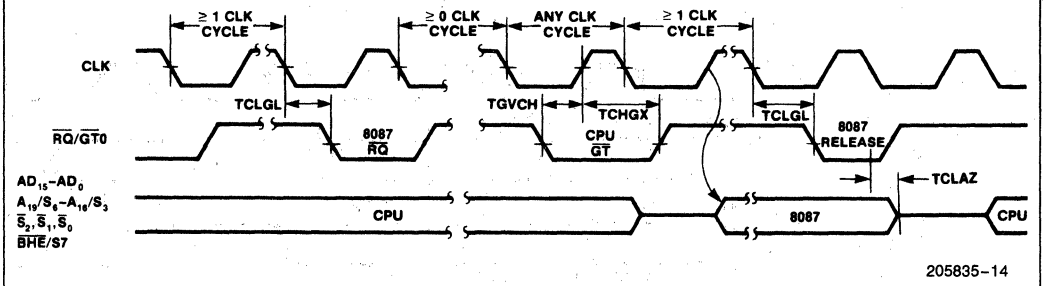


RESET TIMING



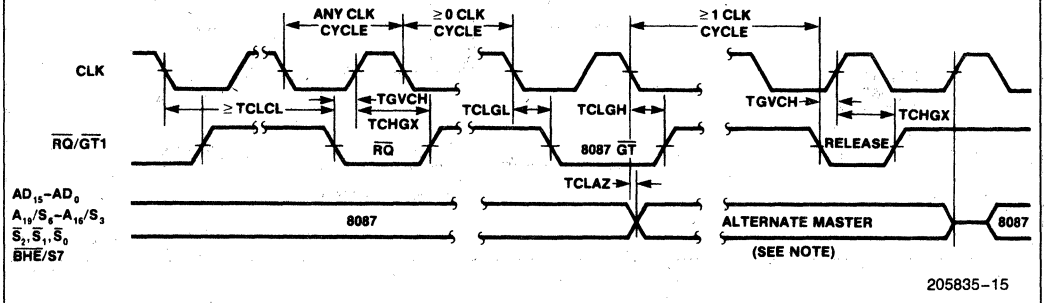
WAVEFORMS (Continued)

REQUEST/GRANT$_0$ TIMING



NOTE:
The CPU provides active pullup of $\overline{RQ}/\overline{GT0}$, see TCLGH spec.

REQUEST/GRANT$_1$ TIMING



NOTE:
Alternate master may not drive the buses outside of the region shown without risking bus contention.

BUSY AND INTERRUPT TIMING

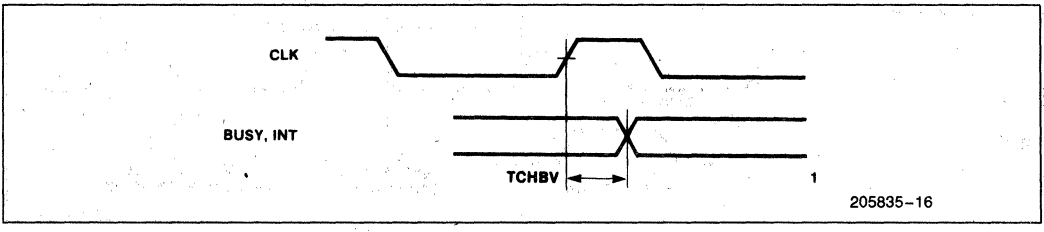


Table 5. 8087 Extensions to the 86/186 Instructions Sets

Data Transfer		Optional 8,16 Bit Displacement	Clock Count Range 32 Bit Real	32 Bit Integer	64 Bit Real	16 Bit Integer
FLD = LOAD	MF =		00	01	10	11
Integer/Real Memory to ST(0)	ESCAPE MF 1 | MOD 0 0 0 R/M	DISP	38–56 + EA	52–60 + EA	40–60 + EA	46–54 + EA
Long Integer Memory to ST(0)	ESCAPE 1 1 1 | MOD 1 0 1 R/M	DISP	60–68 + EA			
Temporary Real Memory to ST(0)	ESCAPE 0 1 1 | MOD 1 0 1 R/M	DISP	53–65 + EA			
BCD Memory to ST(0)	ESCAPE 1 1 1 | MOD 1 0 0 R/M	DISP	290–310 + EA			
ST(i) to ST(0)	ESCAPE 0 0 1 | 1 1 0 0 0 ST(i)		17–22			
FST = STORE						
ST(0) to Integer/Real Memory	ESCAPE MF 1 | MOD 0 1 0 R/M	DISP	84–90 + EA	82–92 + EA	96–104 + EA	80–90 + EA
ST(0) to ST(i)	ESCAPE 1 0 1 | 1 1 0 1 0 ST(i)		15–22			
FSTP = STORE AND POP						
ST(0) to Integer/Real Memory	ESCAPE MF 1 | MOD 0 1 1 R/M	DISP	86–92 + EA	84–94 + EA	98–106 + EA	82–92 + EA
ST(0) to Long Integer Memory	ESCAPE 1 1 1 | MOD 1 1 1 R/M	DISP	94–105 + EA			
ST(0) to Temporary Real Memory	ESCAPE 0 1 1 | MOD 1 1 1 R/M	DISP	52–58 + EA			
ST(0) to BCD Memory	ESCAPE 1 1 1 | MOD 1 1 0 R/M	DISP	520–540 + EA			
ST(0) to ST(i)	ESCAPE 1 0 1 | 1 1 0 1 1 ST(i)		17–24			
FXCH = Exchange ST(i) and ST(0)	ESCAPE 0 0 1 | 1 1 0 0 1 ST(i)		10–15			
Comparison						
FCOM = Compare						
Integer/Real Memory to ST(0)	ESCAPE MF 0 | MOD 0 1 0 R/M	DISP	60–70 + EA	78–91 + EA	65–75 + EA	72–86 + EA
ST(i) to ST (0)	ESCAPE 0 0 0 | 1 1 0 1 0 ST(i)		40–50			
FCOMP = Compare and Pop						
Integer/Real Memory to ST(0)	ESCAPE MF 0 | MOD 0 1 1 R/M	DISP	63–73 + EA	80–93 + EA	67–77 + EA	74–88 + EA
ST(i) to ST(0)	ESCAPE 0 0 0 | 1 1 0 1 1 ST(i)		45–52			
FCOMPP = Compare ST(1) to ST(0) and Pop Twice	ESCAPE 1 1 0 | 1 1 0 1 1 0 0 1		45–55			
FTST = Test ST(0)	ESCAPE 0 0 1 | 1 1 1 0 0 1 0 0		38–48			
FXAM = Examine ST(0)	ESCAPE 0 0 1 | 1 1 1 0 0 1 0 1		12–23			

205835-17

Table 5. 8087 Extensions to the 86/186 Instructions Sets (Continued)

Constants		Optional 8,16 Bit Displacement	Clock Count Range 32 Bit Real	32 Bit Integer	64 Bit Real	16 Bit Integer
	MF =		00	01	10	11
FLDZ = LOAD + 0.0 into ST(0)	ESCAPE 0 0 1 | 1 1 1 0 1 1 1 0		11–17			
FLD1 = LOAD + 1.0 into ST(0)	ESCAPE 0 0 1 | 1 1 1 0 1 0 0 0		15–21			
FLDPI = LOAD π into ST(0)	ESCAPE 0 0 1 | 1 1 1 0 1 0 1 1		16–22			
FLDL2T = LOAD $\log_2 10$ into ST(0)	ESCAPE 0 0 1 | 1 1 1 0 1 0 0 1		16–22			
FLDL2E = LOAD $\log_2 e$ into ST(0)	ESCAPE 0 0 1 | 1 1 1 0 1 0 1 0		15–21			
FLDLG2 = LOAD $\log_{10} 2$ into ST(0)	ESCAPE 0 0 1 | 1 1 1 0 1 1 0 0		18–24			
FLDLN2 = LOAD $\log_e 2$ into ST(0)	ESCAPE 0 0 1 | 1 1 1 0 1 1 0 1		17–23			
Arithmetic						
FADD = Addition						
Integer/Real Memory with ST(0)	ESCAPE MF 0 | MOD 0 0 0 R/M	DISP	90–120 + EA	108–143 + EA	95–125 + EA	102–137 + EA
ST(i) and ST(0)	ESCAPE d P 0 | 1 1 0 0 0 ST(i)		70–100 (Note 1)			
FSUB = Subtraction						
Integer/Real Memory with ST(0)	ESCAPE MF 0 | MOD 1 0 R R/M	DISP	90–120 + EA	108–143 + EA	95–125 + EA	102–137 + EA
ST(i) and ST(0)	ESCAPE d P 0 | 1 1 1 0 R R/M		70–100 (Note 1)			
FMUL = Multiplication						
Integer/Real Memory with ST(0)	ESCAPE MF 0 | MOD 0 0 1 R/M	DISP	110–125 + EA	130–144 + EA	112–168 + EA	124–138 + EA
ST(i) and ST(0)	ESCAPE d P 0 | 1 1 0 0 1 R/M		90–145 (Note 1)			
FDIV = Division						
Integer/Real Memory with ST(0)	ESCAPE MF 0 | MOD 1 1 R R/M	DISP	215–225 + EA	230–243 + EA	220–230 + EA	224–238 + EA
ST(i) and ST(0)	ESCAPE d P 0 | 1 1 1 1 R R/M		193–203 (Note 1)			
FSQRT = Square Root of ST(0)	ESCAPE 0 0 1 | 1 1 1 1 1 0 1 0		180–186			
FSCALE = Scale ST(0) by ST(1)	ESCAPE 0 0 1 | 1 1 1 1 1 1 0 1		32–38			
FPREM = Partial Remainder of ST(0) ÷ ST(1)	ESCAPE 0 0 1 | 1 1 1 1 1 0 0 0		15–190			
FRNDINT = Round ST(0) to Integer	ESCAPE 0 0 1 | 1 1 1 1 1 1 0 0		16–50			

205835-18

NOTE:
1. If P = 1 then add 5 clocks.

Table 5. 8087 Extensions to the 86/186 Instructions Sets (Continued)

		Optional 8,16 Bit Displacement	Clock Count Range
FXTRACT = Extract Components of St(0)	ESCAPE 0 0 1 | 1 1 1 1 0 1 0 0		27–55
FABS = Absolute Value of ST(0)	ESCAPE 0 0 1 | 1 1 1 0 0 0 0 1		10–17
FCHS = Change Sign of ST(0)	ESCAPE 0 0 1 | 1 1 1 0 0 0 0 0		10–17
Transcendental			
FPTAN = Partial Tangent of ST(0)	ESCAPE 0 0 1 | 1 1 1 1 0 0 1 0		30–540
FPATAN = Partial Arctangent of ST(0) ÷ ST(1)	ESCAPE 0 0 1 | 1 1 1 1 0 0 1 1		250–800
F2XM1 = $2^{ST(0)} - 1$	ESCAPE 0 0 1 | 1 1 1 1 0 0 0 0		310–630
FYL2X = ST(1) • Log_2 |ST(0)|	ESCAPE 0 0 1 | 1 1 1 1 0 0 0 1		900–1100
FYL2XP1 = ST(1) • Log_2 [ST(0) + 1]	ESCAPE 0 0 1 | 1 1 1 1 1 0 0 1		700–1000
Processor Control			
FINIT = Initialized 8087	ESCAPE 0 1 1 | 1 1 1 0 0 0 1 1		2–8
FENI = Enable Interrupts	ESCAPE 0 1 1 | 1 1 1 0 0 0 0 0		2–8
FDISI = Disable Interrupts	ESCAPE 0 1 1 | 1 1 1 0 0 0 0 1		2–8
FLDCW = Load Control Word	ESCAPE 0 0 1 | MOD 1 0 1 R/M	DISP	7–14 + EA
FSTCW = Store Control Word	ESCAPE 0 0 1 | MOD 1 1 1 R/M	DISP	12–18 + EA
FSTSW = Store Status Word	ESCAPE 1 0 1 | MOD 1 1 1 R/M	DISP	12–18 + EA
FCLEX = Clear Exceptions	ESCAPE 0 1 1 | 1 1 1 0 0 0 1 0		2–8
FSTENV = Store Environment	ESCAPE 0 0 1 | MOD 1 1 0 R/M	DISP	40–50 + EA
FLDENV = Load Environment	ESCAPE 0 0 1 | MOD 1 0 0 R/M	DISP	35–45 + EA
FSAVE = Save State	ESCAPE 1 0 1 | MOD 1 1 0 R/M	DISP	197–207 + EA
FRSTOR = Restore State	ESCAPE 1 0 1 | MOD 1 0 0 R/M	DISP	197–207 + EA
FINCSTP = Increment Stack Pointer	ESCAPE 0 0 1 | 1 1 1 1 0 1 1 1		6–12
FDECSTP = Decrement Stack Pointer	ESCAPE 0 0 1 | 1 1 1 1 0 1 1 0		6–12

205835-19

Table 5. 8087 Extensions to the 86/186 Instructions Sets (Continued)

		Clock Count Range
FFREE = Free ST(i)	ESCAPE 1 0 1 | 1 1 0 0 0 ST(i)	9–16
FNOP = No Operation	ESCAPE 0 0 1 | 1 1 0 1 0 0 0 0	10–16
FWAIT = CPU Wait for 8087	1 0 0 1 1 0 1 1	3+5n*

205835-20

*n = number of times CPU examines TEST line before 8087 lowers BUSY.

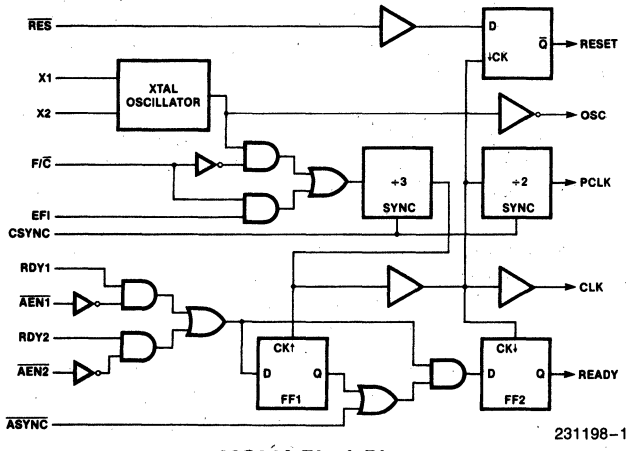
NOTES:
1. if mod = 00 then DISP = 0*, disp-low and disp-high are absent
   if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
   if mod = 10 then DISP = disp-high; disp-low
   if mod = 11 then r/m is treated as an ST(i) field
2. if r/m = 000 then EA = (BX) + (SI) + DISP
   if r/m = 001 then EA = (BX) + (DI) + DISP
   if r/m = 010 then EA = (BP) + (SI) + DISP
   if r/m = 011 then EA = (BP) + (DI) + DISP
   if r/m = 100 then EA = (SI) + DISP
   if r/m = 101 then EA = (DI) + DISP
   if r/m = 110 then EA = (BP) + DISP
   if r/m = 111 then EA = (BX) + DISP
   *except if mod = 000 and r/m = 110 then EA = disp-high; disp-low.
3. MF = Memory Format
   00–32-bit Real
   01–32-bit Integer
   10–64-bit Real
   11–16-bit Integer
4. ST(0) = Current stack top
   ST(i) = i^{th} register below stack top
5. d = Destination
   0—Destination is ST(0)
   1—Destination is ST(i)
6. P = Pop
   0—No pop
   1—Pop ST(0)
7. R = Reverse: When d = 1 reverse the sense of R
   0—Destination (op) Source
   1—Source (op) Destination
8. For **FSQRT:** $-0 \leq ST(0) \leq +\infty$
   For **FSCALE:** $-2^{15} \leq ST(1) < +2^{15}$ and ST(1) integer
   For **F2XM1:** $0 \leq ST(0) \leq 2^{-1}$
   For **FYL2X:** $0 < ST(0) < \infty$
   $-\infty < ST(1) < +\infty$
   For **FYL2XP1:** $0 \leq |ST(0)| < (2 - \sqrt{2})/2$
   $-\infty < ST(1) < \infty$
   For **FPTAN:** $0 \leq ST(0) \leq \pi/4$
   For **FPATAN:** $0 \leq ST(0) < ST(1) < +\infty$

82C84A
CHMOS CLOCK GENERATOR AND DRIVER FOR 80C86, 80C88 PROCESSORS

- **Generates the System Clock for the 80C86, 80C88 Processors:**
  82C84A-5 for 5 MHz
  82C84A for 8 MHz
- **Pin Compatible with Bipolar 8284A***
- **Uses a Crystal or an External Frequency Source**
- **Provides Local READY and MULTIBUS® READY Synchronization**
- **Generates System Reset Output from Schmitt Trigger Input**
- **Capable of Clock Synchronization with other 82C84As**
- **Low Power Consumption**
- **Single 5V Power Supply**
- **TTL Compatible Inputs/Outputs**
- **Available in 18-Lead Plastic DIP**
  (See Packaging Spec., Order #231369)

The Intel 82C84A is a high performance CHMOS clock generator-driver designed to service the requirements of the 80C86/88 and 8086/88. Power consumption is a fraction of that of equivalent bipolar circuits. The chip contains a crystal controlled oscillator, a divide-by-three counter and complete READY synchronization and reset logic. Crystal controlled operation up to 15, 25 MHz utilizes a parallel, fundamental mode crystal and two small load capacitors.

*The Bipolar 8284A requires two load resistors and a resonant crystal.

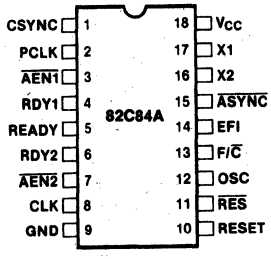


231198-1

82C84A Block Diagram

Control Pin	Logical 1	Logical 0
F/$\overline{C}$	External Clock	Crystal Drive
$\overline{RES}$	Normal	Reset
RDY 1 RDY 2	Bus Ready	Bus not ready
$\overline{AEN\ 1}$ $\overline{AEN\ 2}$	Address Disabled	Address Enabled
$\overline{ASYNC}$	1 Stage Ready Synchronization	2 Stage Ready Synchronization

82C84A Pin Description



231198-2

82C84A 18-Lead DIP Configuration

August 1987
Order Number: 231198-004

Table 1. Pin Description

Symbol	Type	Name and Function
$\overline{AEN1}$, $\overline{AEN2}$	I	**ADDRESS ENABLE:** $\overline{AEN}$ is an active LOW signal. AEN serves to qualify its repective Bus Ready Signal (RDY1 or RDY2). $\overline{AEN1}$ validates RDY1 while $\overline{AEN2}$ validates RDY2. Two AEN signal inputs are useful in system configurations which permit the processor to access two Multi-Master System Busses. In non Multi-Master configurations the $\overline{AEN}$ signal inputs are tied true (LOW).
RDY1, RDY2	I	**BUS READY:** (Transfer Complete). RDY is an active HIGH signal which is an indication from a device located on the system data bus that data has been received, or is available. RDY1 is qualified by $\overline{AEN1}$ while RDY2 is qualified by $\overline{AEN2}$.
$\overline{ASYNC}$	I	**READY SYNCHRONIZATION SELECT:** $\overline{ASYNC}$ is an input which defines the synchronization mode of the READY logic. When $\overline{ASYNC}$ is LOW, two stages of READY synchronization are provided. When $\overline{ASYNC}$ is left open (an internal pull-up is provided) or HIGH a single stage of READY synchronization is provided.
READY	O	**READY:** READY is an active HIGH signal which is the synchronized RDY signal input. READY is cleared after the guaranteed hold time to the processor has been met.
X1, X2	I	**CRYSTAL IN:** X1 and X2 are the pins to which a crystal is attached. The crystal frequency is 3 times the desired processor clock frequency. (If no crystal is attached, then X1 should be tied to V_{CC} or GND and X2 should be left open.)
F/$\overline{C}$	I	**FREQUENCY/CRYSTAL SELECT:** F/$\overline{C}$ is a strapping option. When strapped LOW, F/$\overline{C}$ permits the processor's clock to be generated by the crystal. When F/$\overline{C}$ is strapped HIGH, CLK is generated from the EFI input.
EFI	I	**EXTERNAL FREQUENCY:** When F/$\overline{C}$ is strapped HIGH, CLK is generated from the input frequency appearing on this pin. The input signal is a square wave 3 times the frequency of the desired CLK output. When F/$\overline{C}$ is strapped LOW, EFI should be tied HIGH or LOW.
CLK	O	**PROCESSOR CLOCK:** CLK is the clock output used by the processor and all devices which directly connect to the processor's local bus (i.e., the bipolar support chips and other MOS devices). CLK has an output frequency which is 1/3 of the crystal or EFI input frequency and a 1/3 duty cycle.
PCLK	O	**PERIPHERAL CLOCK:** PCLK is a TTL level peripheral clock signal whose output frequency is 1/2 that of CLK and has a 50% duty cycle.
OSC	O	**OSCILLATOR OUTPUT:** OSC is the TTL level output of the internal oscillator circuitry. Its frequency is equal to that of the crystal.
$\overline{RES}$	I	**RESET IN:** $\overline{RES}$ is an active LOW signal which is used to generate RESET. The 82C84A provides a Schmitt trigger input so that an RC connection can be used to establish the power-up reset of proper duration.

Table 1. Pin Description (Continued)

Symbol	Type	Name and Function
RESET	O	**RESET:** RESET is an active HIGH signal which is used to reset the 80C86/88 family processors. Its timing characteristics are determined by $\overline{RES}$.
CSYNC	I	**CLOCK SYNCHRONIZATION:** CSYNC is an active HIGH signal which allows multiple 82C84A's to be synchronized to provide clocks that are in phase. When CSYNC is HIGH the internal counters are reset. When CSYNC goes LOW the internal counters are allowed to resume counting. CSYNC needs to be externally synchronized to EFI. When using the internal oscillator CSYNC should be hardwired to ground.
GND		**GROUND.**
V_{CC}		**POWER:** +5V supply.

FUNCTIONAL DESCRIPTION

Oscillator

The oscillator circuit of the 82C84A is designed primarily for use with an external parallel resonant, fundamental mode crystal from which the basic operating frequency is derived.

The crystal frequency should be selected at three times the required CPU clock. X1 and X2 are the two crystal input crystal connections. For the most stable operation of the oscillator (OSC) output circuit, two capacitors (C1 = C2) as shown in the waveform figures are recommended. The output of the oscillator is buffered and brought out on OSC so that other system timing signals can be derived from this stable, crystal-controlled source.

Capacitors C1, C2 are chosen such that their combined capacitance:

$$CT = \frac{C1 \bullet C2}{C1 + C2} \quad \text{(Including stray capacitance)}$$

matches the load capacitance as specified by the crystal manufacturer. This insures operation within the frequency tolerance specified by the crystal manufacturer.

Clock Generator

The clock generator consists of a synchronous divide-by-three counter with a special clear input that inhibits the counting. This clear input (CSYNC) allows the output clock to be synchronized with an external event (such as another 82C84A clock). It is necessary to synchronize the CSYNC input to the EFI clock external to the 82C84A. This is accomplished with two Schottky flip-flops. The counter output is a 33% duty cycle clock at one-third the input frequency.

The F/$\overline{C}$ input is a strapping pin that selects either the crystal oscillator or the EFI input as the clock for the ÷3 counter. If the EFI input is selected as the clock source, the oscillator section can be used independently for another clock source. Output is taken from OSC.

Clock Outputs

The CLK ouput is a 33% duty cycle MOS clock driver designed to drive the 80C86/88 processors directly. PCLK is a TTL level peripheral clock signal whose output frequency is ½ that of CLK. PCLK has a 50% duty cycle.

Reset Logic

The reset logic provides a Schmitt trigger input ($\overline{RES}$) and a synchronizing flip-flop to generate the reset timing. The reset signal is synchronized to the falling edge of CLK. A simple RC network can be used to provide power-on reset by utilizing this function of the 82C84A.

READY Synchronization

Two READY inputs (RDY1, RDY2) are provided to accommodate two Multi-Master system busses. Each input has a qualifier ($\overline{AEN1}$ and $\overline{AEN2}$, respectively). The $\overline{AEN}$ signals validate their respective RDY signals. If a Multi-Master system is not being used the $\overline{AEN}$ pin should be tied LOW.

Synchronization is required for all asynchronous active-going edges of either RDY input to guarantee that the RDY setup and hold times are met. Inactive-going edges of RDY in normally ready systems do not require synchronization but must satisfy RDY setup and hold as a matter of proper system design.

The $\overline{\text{ASYNC}}$ input defines two modes of READY synchronization operation.

When $\overline{\text{ASYNC}}$ is LOW, two stages of synchronization are provided for active READY input signals. Positive-going asynchronous READY inputs will first be synchronized to flip-flop one at the rising edge of CLK and then synchronized to flip-flop two at the next falling edge of CLK, after which time the READY output will go active (HIGH). Negative-going asynchronous READY inputs will be synchronized directly to flip-flop two at the falling edge of CLK, after which time the READY output will go inactive. This mode of operation is intended for use by asynchronous (normally not ready) devices in the system which cannot be guaranteed by design to meet the required RDY setup timing, T_{R1VCL}, on each bus cycle.

When $\overline{\text{ASYNC}}$ is HIGH, the first READY flip-flop is bypassed in the READY synchronization logic. READY inputs are synchronized by flip-flop two on the falling edge of CLK before they are presented to the processor. This mode is available for synchronous devices that can be guaranteed to meet the required RDY setup time.

$\overline{\text{ASYNC}}$ can be changed on every bus cycle to select the appropriate mode of synchronization for each device in the system.

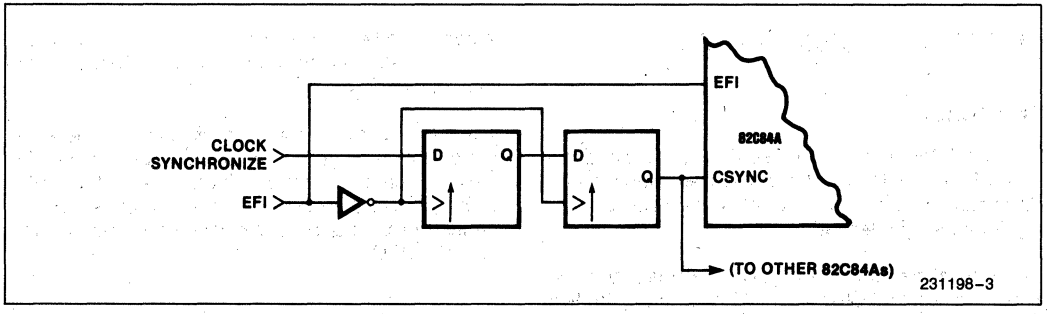


Figure 3. CSYNC Synchronization

ABSOLUTE MAXIMUM RATINGS*

Supply Voltage ..... −0.5V to 7.0V
Input Voltage Applied ..... −0.5V to V_{CC} +0.5V
Output Voltage Applied ..... −0.5V to V_{CC} +0.5V
Storage Temperature ..... −65°C to +150°C
Ambient Temp. Under Bias ..... 0°C to +70°C
Power Dissipation ..... 1.0 Watt

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS (T_A = 0°C to +70°C, V_{CC} = 5V ±10%)

Symbol	Parameter		Min	Max	Units	Test Conditions
I_{CC}	Operating Supply Current	82C84A		10	mA	25 MHz xtal, C_L = 0
		82C84A-5		10	mA	15 MHz xtal, C_L = 0
I_{CCS}	Stand By Supply Current (Note 1)			100	μA	
I_{LI}	Input Leakage Current (Note 2)	$\overline{\text{ASYNC}}$ Only		10	μA	$\overline{\text{ASYNC}}$ = V_{CC}
				−130	μA	$\overline{\text{ASYNC}}$ = GND
		All Other Pins		±1.0	μA	0V ≤ V_{IN} ≤ V_{CC}

D.C. CHARACTERISTICS (Continued)

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input LOW Voltage		0.8	V	
V_{IH}	Input HIGH Voltage	2.2	V_{CC} + 0.5	V	
V_{IHR}	Reset Input HIGH Voltage	0.6 V_{CC}		V	
V_{OL}	Output LOW Voltage		0.4	V	CLK: I_{OL} = 4 mA Others: I_{OL} = 2.5 mA
V_{OH}	Output HIGH Voltage	V_{CC} − 0.4		V V	CLK: I_{OH} = −4 mA Others: I_{OH} = −2.5 mA
V_{IHR}-V_{ILR}	$\overline{RES}$ Input Hysteresis: 82C84A 82C84A-5	0.15 0.25		V V	
C_{IN}	Input Capacitance		7	pF	freq = 1 MHz

NOTES:
1. V_{IH}, F/$\overline{C}$, X1 ≥ V_{CC} − 0.2V; EFI = V_{CC} or GND; ASYNC = V_{CC} or OPEN; X2 = OPEN; V_{IL} ≤ 0.2V.
2. An internal pull-up resistor is implemented on the $\overline{ASYNC}$ input.

A.C. CHARACTERISTICS (T_A = 0°C to +70°C, V_{CC} = 5V ±10%)

TIMING REQUIREMENTS

Symbol	Parameter	82C84A		82C84A-5		Units	Test Conditions
		Min	Max	Min	Max		
t_{EHEL}	External Frequency HIGH Time	13		20		ns	90%–90% V_{IN}
t_{ELEH}	External Frequency LOW Time	13		20		ns	10%–10% V_{IN}
t_{ELEL}	EFI Period	40		66		ns	(Note 1)
	XTAL Frequency	2.4	25	6.0	15	MHz	
t_{R1VCL}	RDY1, RDY2 Active Setup to CLK	35		35		ns	$\overline{ASYNC}$ = HIGH
t_{R1VCH}	RDY1, RDY2 Active Setup to CLK	35		35		ns	$\overline{ASYNC}$ = LOW
t_{R1VCL}	RDY1, RDY2 Inactive Setup to CLK	35		35		ns	
t_{CLR1X}	RDY1, RDY2 Hold to CLK	0		0		ns	
t_{AYVCL}	$\overline{ASYNC}$ Setup to CLK	50		50		ns	
t_{CLAYX}	$\overline{ASYNC}$ Hold to CLK	0		0		ns	
t_{A1VR1V}	$\overline{AEN1}$, $\overline{AEN2}$ Setup to RDY1, RDY2	15		15		ns	
t_{CLA1X}	$\overline{AEN1}$, $\overline{AEN2}$ Hold to CLK	0		0		ns	
t_{YHEH}	CSYNC Setup to EFI	20		20		ns	
t_{EHYL}	CSYNC Hold to EFI	10		20		ns	
t_{YHYL}	CSYNC Width	2 • t_{ELEL}		2 • t_{ELEL}		ns	
t_{I1HCL}	$\overline{RES}$ Setup to CLK	65		65		ns	(Note 2)
t_{CLI1H}	$\overline{RES}$ Hold to CLK	20		20		ns	(Note 2)
t_{ILIH}	Input Rise Time		15		15	ns	(Note 1)
t_{IHIL}	Input Fall Time		15		15	ns	(Note 1)

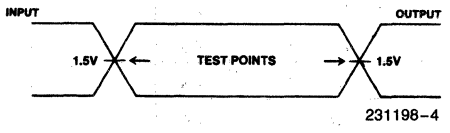
A.C. CHARACTERISTICS (Continued)

TIMING RESPONSES

Symbol	Parameter	Min 82C84A	Min 82C84A-5	Max	Units	Test Conditions
t_{CLCL}	CLK Cycle Period	125	200		ns	
t_{CHCL}	CLK HIGH Time	$(1/3\ t_{CLCL})+2$	$(1/3\ t_{CLCL})+2$		ns	
t_{CLCH}	CLK LOW Time	$(2/3\ t_{CLCL})-15$	$(2/3\ t_{CLCL})-15$		ns	
t_{CH1CH2} t_{CL2CL1}	CLK Rise or Fall Time			10	ns	1.0V to 3.5V
t_{PHPL}	PCLK HIGH Time	$t_{CLCL}-20$	$t_{CLCL}-20$		ns	
t_{PLPH}	PCLK LOW Time	$t_{CLCL}-20$	$t_{CLCL}-20$		ns	
t_{RYLCL}	Ready Inactive to CLK (Note 4)	−8	−8		ns	
t_{RYHCH}	Ready Active to CLK (Note 3)	$(2/3\ t_{CLCL})-15$	$(2/3\ t_{CLCL})-15$		ns	
t_{CLIL}	CLK to Reset Delay			40	ns	
t_{CLPH}	CLK to PCLK HIGH DELAY			22	ns	
t_{CLPL}	CLK to PCLK LOW Delay			22	ns	
t_{OLCH}	OSC to CLK HIGH Delay	−5	−5	22	ns	
t_{OLCL}	OSC to CLK LOW Delay	2	2	35	ns	
t_{OLOH}	Output Rise Time (except CLK)			15	ns	From 0.8V to 2.0V
t_{OHOL}	Output Fall Time (except CLK)			15	ns	From 2.0V to 0.8V

NOTES:
1. Transition between V_{IL}(max) − 0.4V and V_{IH}(min) + 0.4V.
2. Setup and hold necessary only to guarantee recognition at next clock.
3. Applies only to T3 and TW states.
4. Applies only to T2 states.

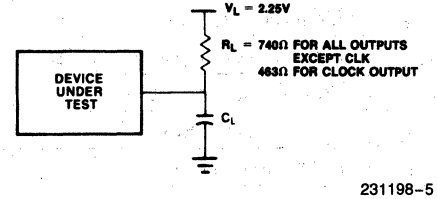
A.C. TESTING INPUT, OUTPUT WAVEFORM



231198-4

A.C. TESTING: ALL INPUT SIGNALS MUST SWITCH BETWEEN 0.45V AND 2.4V. T_{RISE} AND T_{FALL} MUST BE ≤ 15 ns.
ALL TIMING MEASUREMENTS ARE MADE AT 1.5V.

A.C. TESTING LOAD CIRCUIT

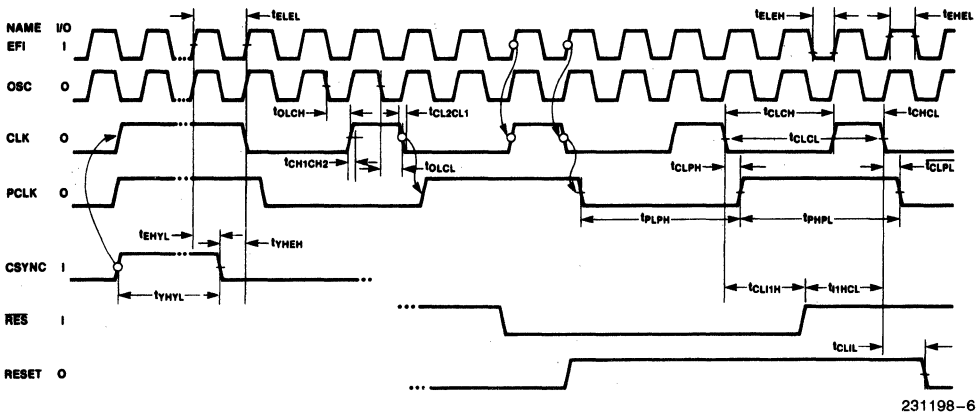


231198-5

C_L = 100 pF FOR CLK
C_L = 30 pF FOR READY
C_L INCLUDES PROBE AND JIG CAPACITANCE

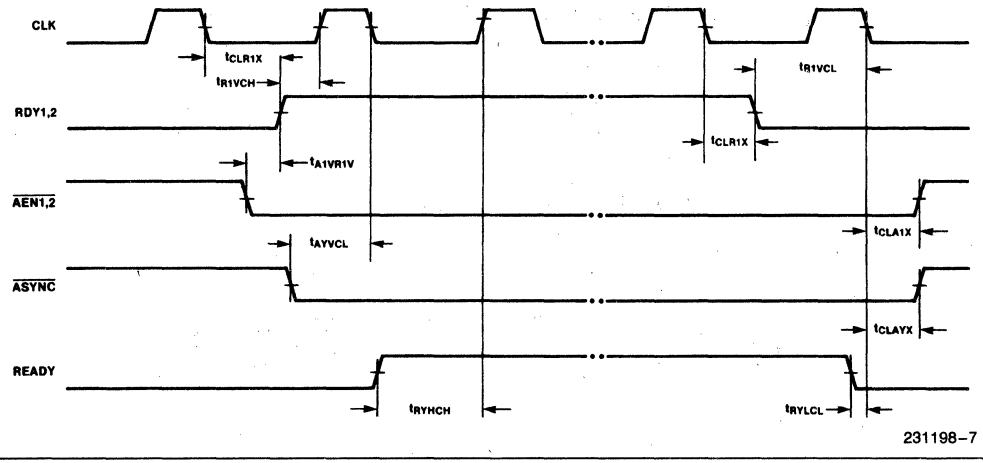
WAVEFORMS

CLOCKS AND RESET SIGNALS



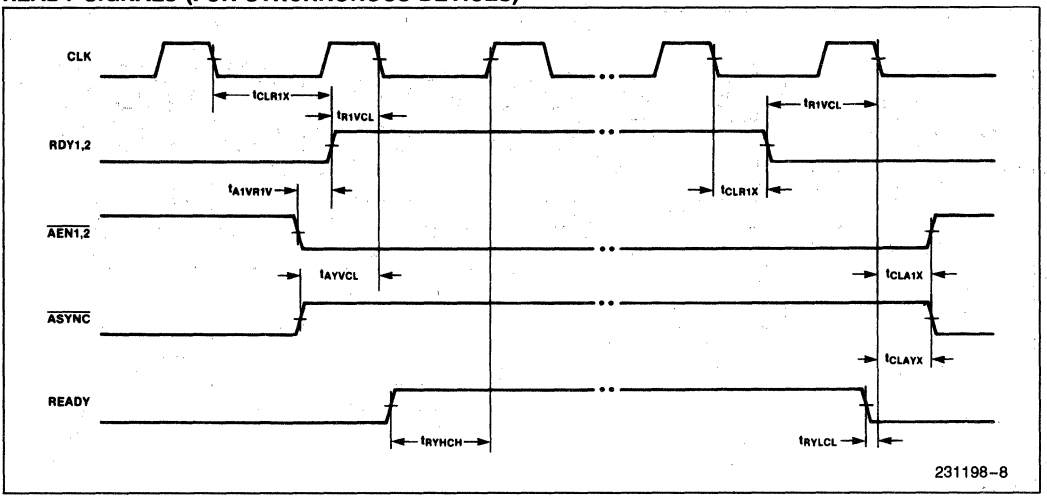
NOTE:
ALL TIMING MEASUREMENTS ARE MADE AT 1.5V, UNLESS OTHERWISE NOTED.

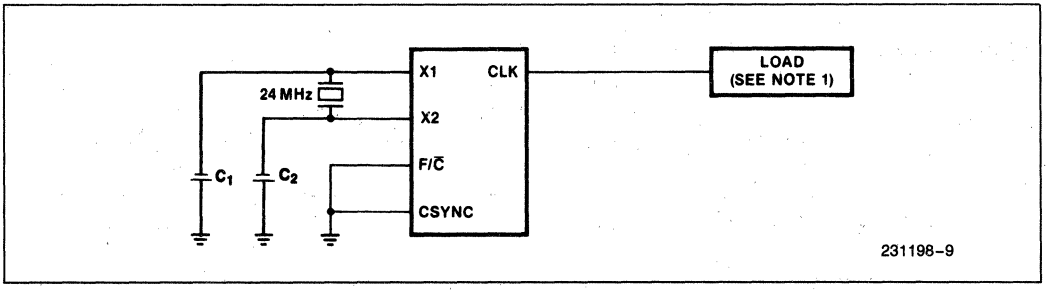
READY SIGNALS (FOR ASYNCHRONOUS DEVICES)



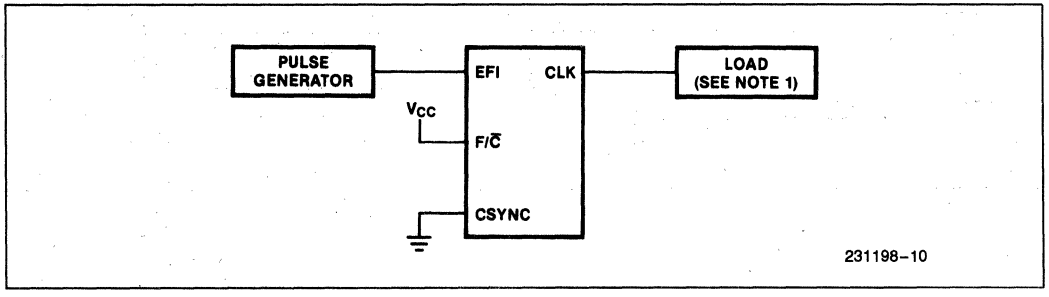
WAVEFORMS (Continued)

READY SIGNALS (FOR SYNCHRONOUS DEVICES)

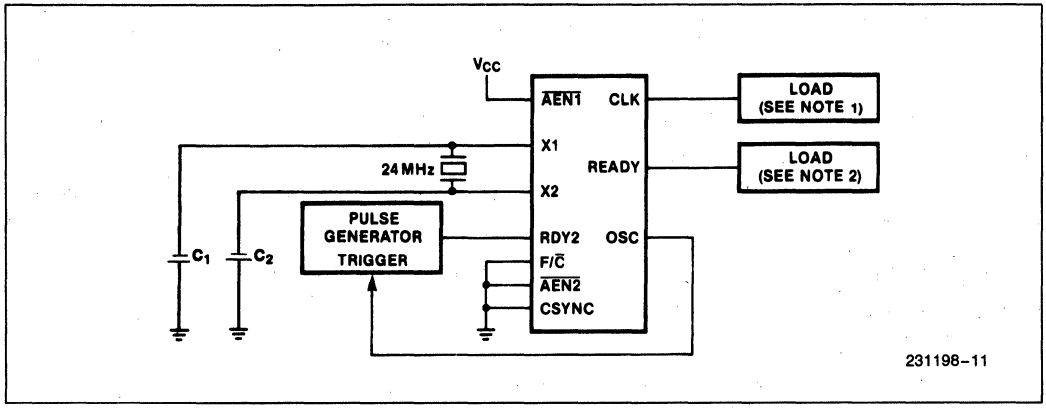




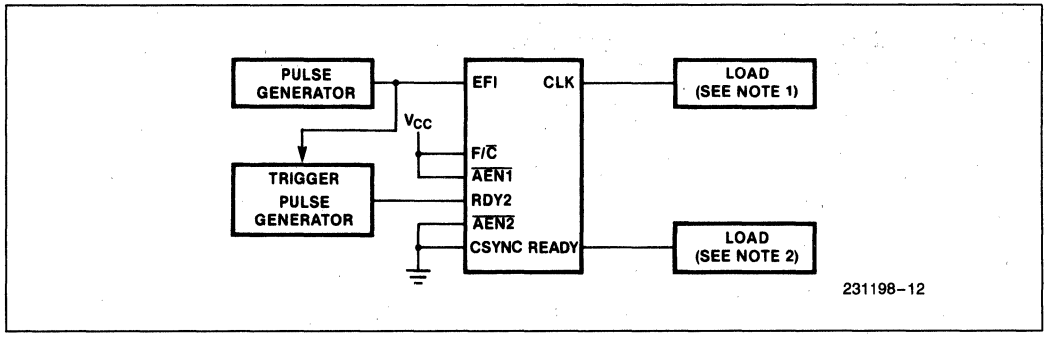
Clock High and Low Time (Using X1, X2)



Clock High and Low Time (Using EFI)



Ready to Clock (Using X1, X2)



Ready to Clock (Using EFI)

NOTES:
1. C_L = 100 pF
2. C_L = 30 pF

DATA SHEET REVISION REVIEW

The following list represents key differences between this and the October 1986 (order no. 231198-003) data sheet. Please review this summary.

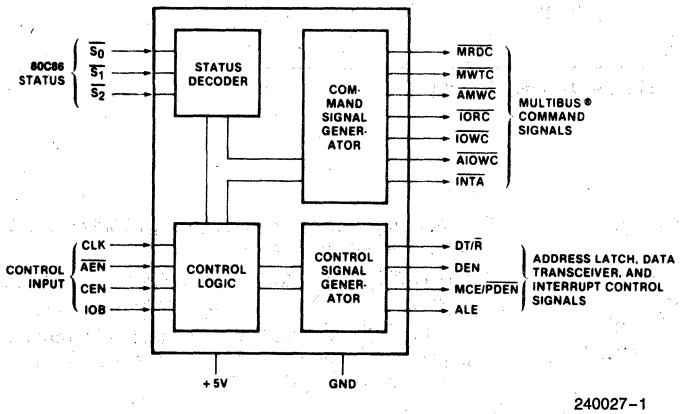
1. A diagram of the PLCC package was deleted.
2. Test conditions for I_{CCS} (stand by supply current) were clarified in Note 1.
3. t_{CLI1H} ($\overline{RES}$ Hold to CLK) for 82C84A changed from 10 ns to 20 ns to reflect current testing.
4. For the "Clocks and Reset Signals" diagram, all timing measurements voltages changed to 1.5V from 0.8V and 2.0V.

82C88
CHMOS BUS CONTROLLER

- **Pin Compatible with Bipolar 8288**
- **Provides Support for 8086/88, 80C86/88**
- **Low Power Operation**
  - **— I_{CCS} = 100 μA**
  - **— I_{CC} = 10 mA**
- **Provides Advanced Commands for Multi-Master Busses**
- **3-State Command Output Drivers**
- **High Drive Capability**
- **Configurable for Use with an I/O Bus**
- **Single 5V Power Supply**
- **8 MHz Operation**
  - **— 82C88-2**

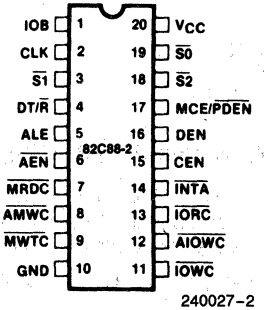
The Intel 82C88-2 is a high performance CHMOS version of the 8288 bipolar bus controller. The 82C88-2 provides command and control timing generation for 8086 architecture* systems. Static CHMOS circuit design ensures low operating power. The 82C88-2 high output drive capability eliminates the need for additional bus drivers.

***NOTE:**
In this data sheet, all references to 8086 or 8086 architecture include: 8086/88 and 80C86/88.



240027-1

Figure 1. Block Diagram



240027-2

Figure 2a. 82C88-2 20-Lead DIP Configuration

September 1988
Order Number: 240027-002

Table 1. Pin Description

Symbol	Type	Name and Function
V_{CC}		**POWER:** +5V supply.
GND		**GROUND.**
$\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$	I	**STATUS INPUT PINS:** These pins are the status input pins from the 8086 or 8088 processors. The 82C88-2 decodes these inputs to generate command and control signals at the appropriate time. When these pins are not in use (passive) they are all HIGH. (See chart under Command and Control Logic.) Internal pull-up resistors hold these lines HIGH when no other driving source is present.
CLK	I	**CLOCK:** This is a clock signal from the 82C84A clock generator and serves to establish when command and control signals are generated.
ALE	O	**ADDRESS LATCH ENABLE:** This signal serves to strobe an address into the address latches. This signal is active HIGH and latching occurs on the falling (HIGH to LOW) transition. ALE is intended for use with transparent D type latches.
DEN	O	**DATA ENABLE:** This signal serves to enable data transceivers onto either the local or system data bus. This signal is active HIGH.
DT/$\overline{R}$	O	**DATA TRANSMIT/RECEIVE:** This signal establishes the direction of data flow through the transceivers. A HIGH on this line indicates Transmit (write to I/O or memory) and a LOW indicates Receive (Read).
$\overline{AEN}$	I	**ADDRESS ENABLE:** $\overline{AEN}$ enables command outputs of the 82C88-2 Bus Controller at least T_{AELCV} after it becomes active (LOW). $\overline{AEN}$ going inactive immediately 3-states the command output drivers. $\overline{AEN}$ does not affect the I/O command lines if the 82C88-2 is in the I/O Bus mode (IOB tied HIGH).
CEN	I	**COMMAND ENABLE:** When this signal is LOW all 82C88-2 command outputs and the DEN and $\overline{PDEN}$ control outputs are forced to their inactive state. When this signal is HIGH, these same outputs are enabled.
IOB	I	**INPUT/OUTPUT BUS MODE:** When the IOB is strapped HIGH the 82C88-2 functions in the I/O Bus mode. When it is strapped LOW, the 82C88-2 functions in the System Bus mode. (See sections on I/O Bus and System Bus modes).
$\overline{AIOWC}$	O	**ADVANCED I/O WRITE COMMAND:** The $\overline{AIOWC}$ issues an I/O Write Command earlier in the machine cycle to give I/O devices an early indication of a write instruction. Its timing is the same as a read command signal. $\overline{AIOWC}$ is active LOW.
$\overline{IOWC}$	O	**I/O WRITE COMMAND:** This command line instructs an I/O device to read the data on the data bus. This signal is active LOW.
$\overline{IORC}$	O	**I/O READ COMMAND:** This command line instructs an I/O device to drive its data onto the data bus. This signal is active LOW.
$\overline{AMWC}$	O	**ADVANCED MEMORY WRITE COMMAND:** The $\overline{AMWC}$ issues a memory write command earlier in the machine cycle to give memory devices an early indication of a write instruction. Its timing is the same as read command signal. $\overline{AMWC}$ is active LOW.
$\overline{MWTC}$	O	**MEMORY WRITE COMMAND:** This command line instructs the memory to record the data present on the data bus. This signal is active LOW.
$\overline{MRDC}$	O	**MEMORY READ COMMAND:** This command line instructs the memory to drive its data onto the data bus. This signal is active LOW.
$\overline{INTA}$	O	**INTERRUPT ACKNOWLEDGE:** This command line tells an interrupting device that its interrupt has been acknowledged and that it should drive vectoring information onto the data bus. This signal is active LOW.
MCE/$\overline{PDEN}$	O	This is a dual function pin. **MCE (IOB IS TIED LOW):** Master Cascade Enable occurs during an interrupt sequence and serves to read a Cascade Address from a master PIC (Priority Interrupt Controller) onto the data bus. The MCE signal is active HIGH. **$\overline{PDEN}$ (IOB IS TIED HIGH):** Peripheral Data Enable enables the data bus transceiver for the I/O bus that DEN performs for the system bus. $\overline{PDEN}$ is active LOW.

FUNCTIONAL DESCRIPTION

Command and Control Logic

The command logic decodes the three 8086 or 8088 CPU status lines ($\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$) to determine what command is to be issued.

This chart shows the meaning of each status "word".

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Processor State	82C88-2 Command
0	0	0	Interrupt Acknowledge	$\overline{INTA}$
0	0	1	Read I/O Port	$\overline{IORC}$
0	1	0	Write I/O Port	$\overline{IOWC}$, $\overline{AIOWC}$
0	1	1	Halt	None
1	0	0	Code Access	$\overline{MRDC}$
1	0	1	Read Memory	$\overline{MRDC}$
1	1	0	Write Memory	$\overline{MWTC}$, $\overline{AMWC}$
1	1	1	Passive	None

The command is issued in one of two ways dependent on the mode of the 82C88-2 Bus Controller.

I/O Bus Mode — The 82C88-2 is in the I/O Bus mode if the IOB pin is strapped HIGH. In the I/O Bus mode all I/O command lines ($\overline{IORC}$, $\overline{IOWC}$, $\overline{AIOWC}$, $\overline{INTA}$) are always enabled (i.e., not dependent on $\overline{AEN}$). When an I/O command is initiated by the processor, the 82C88-2 immediately activates the command lines, using $\overline{PDEN}$ and DT/$\overline{R}$ to control the I/O bus transceiver. The I/O command lines should not be used to control the system bus in this configuration because no arbitration is present. This mode allows one 82C88-2 Bus Controller to handle two external busses. No waiting is involved when the CPU wants to gain access to the I/O bus. Normal memory access requires a "Bus Ready" signal ($\overline{AEN}$ LOW) before it will proceed. It is advantageous to use the IOB mode if I/O or peripherals dedicated to one processor exist in a multi-processor system.

System Bus Mode — The 82C88-2 in the System Bus mode if the IOB pin is strapped LOW. In this mode no command is issued until T_{AELCV} after the $\overline{AEN}$ Line is activated (LOW). This mode assumes bus arbitration logic will inform the bus controller (on the $\overline{AEN}$ line) when the bus is free for use. Both memory and I/O commands wait for bus arbitration. This mode is used when only one bus exists. Here, both I/O and memory are shared by more than one processor.

COMMAND OUTPUTS

The advanced write commands are made available to initiate write procedures early in the machine cycle. This signal can be used to prevent the processor from entering an unnecessary wait state.

The command outputs are:

- $\overline{MRDC}$ — Memory Read Command
- $\overline{MWTC}$ — Memory Write Command
- $\overline{IORC}$ — I/O Read Command
- $\overline{IOWC}$ — I/O Write Command
- $\overline{AMWC}$ — Advanced Memory Write Command
- $\overline{AIOWC}$ — Advanced I/O Write Command
- $\overline{INTA}$ — Interrupt Acknowledge

$\overline{INTA}$ (Interrupt Acknowledge) acts as an I/O read during an interrupt cycle. Its purpose is to inform an interrupting device that its interrupt is being acknowledged and that it should place vectoring information onto the data bus.

CONTROL OUTPUTS

The control outputs of the 82C88-2 are Data Enable (DEN), Data Transmit/Receive (DT/$\overline{R}$) and Master Cascade Enable/Peripheral Data Enable (MCE/$\overline{PDEN}$). The DEN signal determines when the external bus should be enabled onto the local bus and the DT/$\overline{R}$ determines the direction of data transfer. These two signals usually go to the chip select and direction pins of a transceiver.

The MCE/$\overline{PDEN}$ pin changes function with the two modes of the 82C88-2. When the 82C88-2 is in the IOB mode (IOB HIGH) the $\overline{PDEN}$ signal serves as a dedicated data enable signal for the I/O or Peripheral System bus.

INTERRUPT ACKNOWLEDGE AND MCE

The MCE signal is used during an interrupt acknowledge cycle if the 82C88-2 is in the System Bus mode (IOB LOW). During any interrupt sequence there are two interrupt acknowledge cycles that occur back to back. During the first interrupt cycle no data or address transfers take place. Logic should be provided to mask off MCE during this cycle. Just before the second cycle begins the MCE signal gates a master Priority Interrupt Controller's (PIC) cascade address onto the processor's local bus where ALE (Address Latch Enable) strobes it into the address latches. On the leading edge of the second interrupt cycle the addressed slave PIC gates an interrupt vector onto the system data bus where it is read by the processor.

If the system contains only one PIC, the MCE signal is not used. In this case the second interrupt Ac-

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias .....0°C to 70°C
Storage Temperature .....−55°C to +150°C
Supply Voltage
with Respect to GND .....−0.5V to +7.0V
All Input Voltages
with Respect to GND .....−0.5V to V_{CC} +0.5V
All Output Voltages
with Respect to GND .....−0.5V to V_{CC} +0.5V
Power Dissipation 0.7W

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS (V_{CC} = 5V ±10%, T_A = 0°C to 70°C)

Symbol	Parameter	Min	Max	Unit	Test Conditions
I_{CC}	Operating Supply Current		10	mA	C_L = 0 pF t_{CLCL} = 200 ns
I_{CCS}	Standby Supply Current		100	μA	Outputs Unloaded (Note 1)
V_{IH}	Input High Voltage	2.2	V_{CC} + 0.3	V	
V_{IL}	Input Low Voltage	−0.3	0.8	V	
V_{CH}	V_{IH} for Clock, $\overline{S0}$, $\overline{S1}$, $\overline{S2}$	3.0	V_{CC} + 0.3	V	
V_{CL}	V_{IL} for Clock, $\overline{S0}$, $\overline{S1}$, $\overline{S2}$	−0.3	0.8	V	
I_{LI}	Input Leakage Current		±10	μA	0V ≤ V_{IN} ≤ V_{CC} (Note 2)
I_{LO}	Output Leakage Current		±10	μA	0V ≤ V_{OUT} ≤ V_{CC}
I_{LIS}	Status Input Current	−100	10	μA	0V ≤ V_{IN} ≤ V_{CC}
V_{OL}	Output Low Voltage: Command Outputs Control Outputs		 0.5 0.45	V	 I_{OL} = 20 mA I_{OL} = 8 mA
V_{OH}	Output High Voltage: Command Outputs Control Outputs	 3.7 3.7		V	 I_{OH} = −8 mA I_{OH} = −4 mA
C_{IN}	Input Capacitance		7	pF	Freq. = 1 MHz Unmeasured pins at GND
C_{OUT}	Output Capacitance		15	pF	Freq. = 1 MHz Unmeasured pins at GND

NOTES:
1. I_{CCS} test conditions are: Status inputs @ V_{CC}, other inputs @ V_{CC} or GND, Outputs open.
2. Except $\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$.

knowledge signal gates the interrupt vector onto the processor bus.

ADDRESS LATCH ENABLE AND HALT

Address Latch Enable (ALE) occurs during each machine cycle and serves to strobe the current address into the address latches. ALE also serves to strobe the status ($\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$) into a latch for halt state decoding.

COMMAND ENABLE

The Command Enable (CEN) input acts as a command qualifier for the 82C88-2. If the CEN pin is high the 82C88-2 functions normally. If the CEN pin is pulled LOW, all command lines are held in their inactive state (not 3-state). This feature can be used to implement memory partitioning and to eliminate address conflicts between system bus devices and resident bus devices.

A.C. CHARACTERISTICS (V_{CC} = 5V ±10%, T_A = 0°C to 70°C)*

82C88-2 TIMING REQUIREMENTS

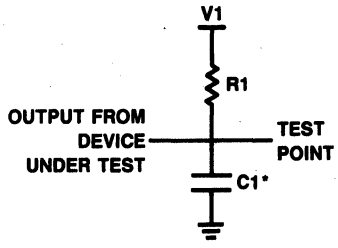
Symbol	Parameter	Min	Max	Units	Test Conditions
fc	CLK Frequency		8	MHz	
TCLCL	CLK Cycle Period	125		ns	
TCLCH	CLK Low Time	66		ns	
TCHCL	CLK High Time	40		ns	
TSVCH	Status Active Setup Time	35		ns	
TCHSV	Status Inactive Hold Time	10		ns	
TSHCL	Status Inactive Setup Time	35		ns	
TCLSH	Status Active Hold Time	10		ns	

82C88-2 TIMING RESPONSES

Symbol	Parameter	Min	Max	Units	Test Conditions**
TCVNV	Control Active Delay	5	45	ns	a
TCVNX	Control Inactive Delay	5	45	ns	a
TCLLH	ALE Active Delay (from CLK)		25	ns	a
TCLMCH	MCE Active Delay (from CLK)		25	ns	a
TMHNL	Command to DEN Delay	TCLCH-5		ns	DEN: a Command: b
TSVLH	ALE Active Delay (from Status)		25	ns	a
TSVMCH	MCE Active Delay (from Status)		30	ns	a
TCHLL	ALE Inactive Delay	4	25	ns	a
TCLML	Command Active Delay	5	35	ns	b
TCLMH	Command Inactive Delay	5	45	ns	b
TCHDTL	Direction Control Active Delay		50	ns	a
TCHDTH	Direction Control Inactive Delay		30	ns	a
TAELCH	Command Enable Time		40	ns	c
TAEHCZ	Command Disable Time		40	ns	d
TAELCV	Enable Delay Time	100	250	ns	b
TAEVNV	$\overline{AEN}$ to DEN		35	ns	a
TCEVNV	CEN to DEN, $\overline{PDEN}$		35	ns	a
TCELRH	CEN to Command		TCLML + 10	ns	b
TOLOH	Output, Rise Time		15	ns	a, b. From 0.8V to 2.2V
TOHOL	Output, Fall Time		15	ns	a, b. From 2.2V to 0.8V

**See Test Condition Definition Table.

TEST LOAD CIRCUITS—3-STATE COMMAND OUTPUT TEST LOAD



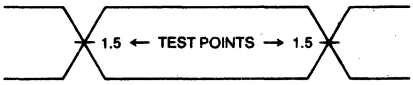
240027-3

*Includes stray and jig capacitance

Test Condition Definition Table

Test Condition	V1, v	R1, Ω	C1, pf
a	2.13	220	80
b	2.29	91	150
c	1.5	187	150
d	1.5	187	50

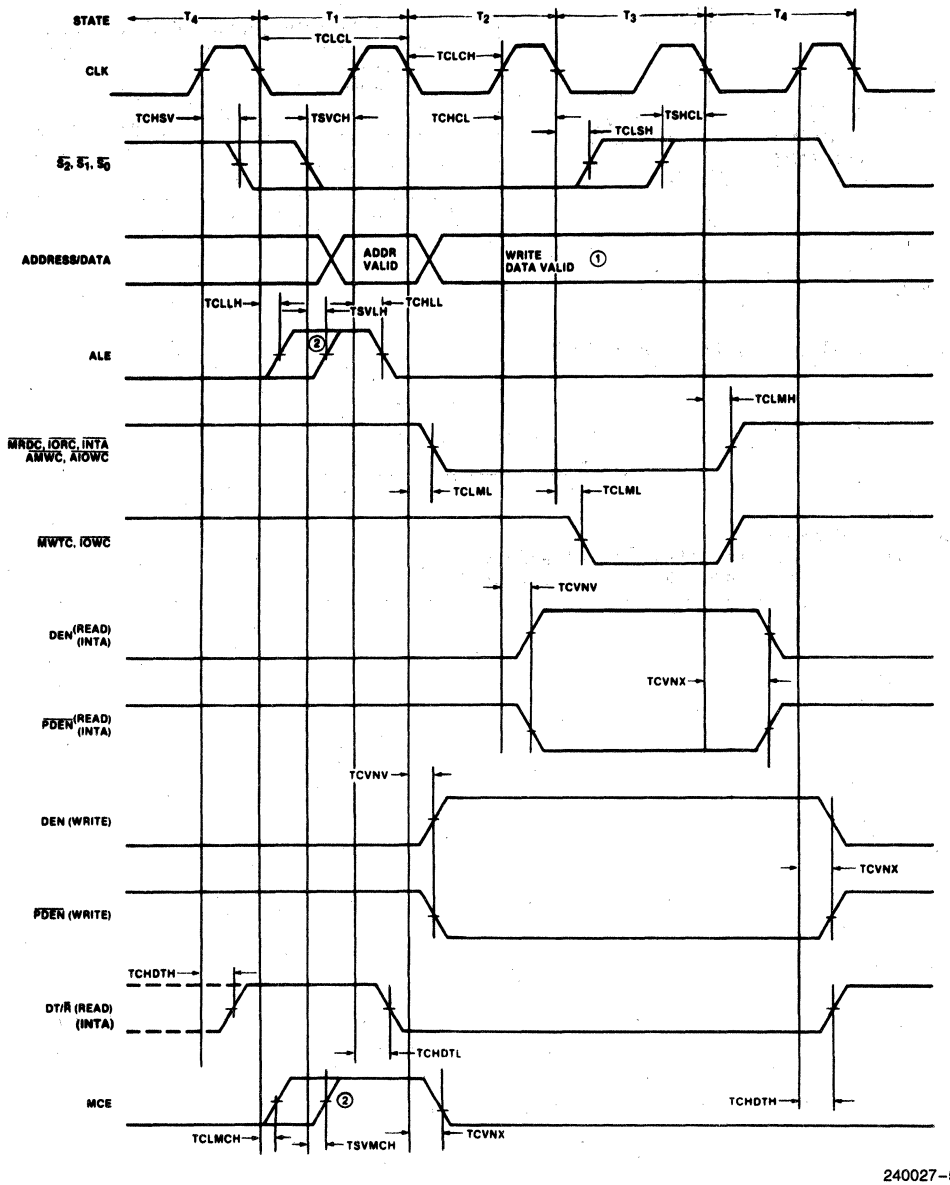
A.C. TESTING INPUT, OUTPUT WAVEFORM



240027-4

A.C. Testing:
Input rise and fall times are:
5 ns ±2 ns between 0.8V and 2.2V for $\overline{AEN}$, CEN and IOB
8 ns ±2 ns between 0.8V and 3.0V for $\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$ and CLK.

WAVEFORMS

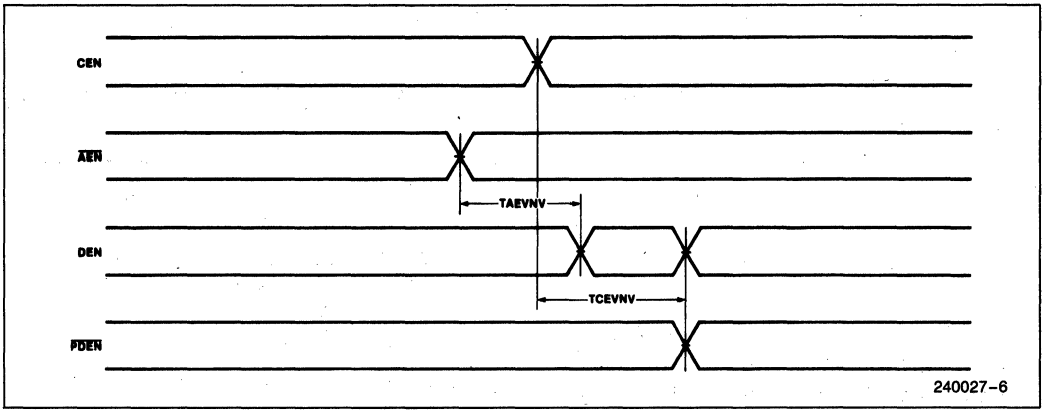


240027-5

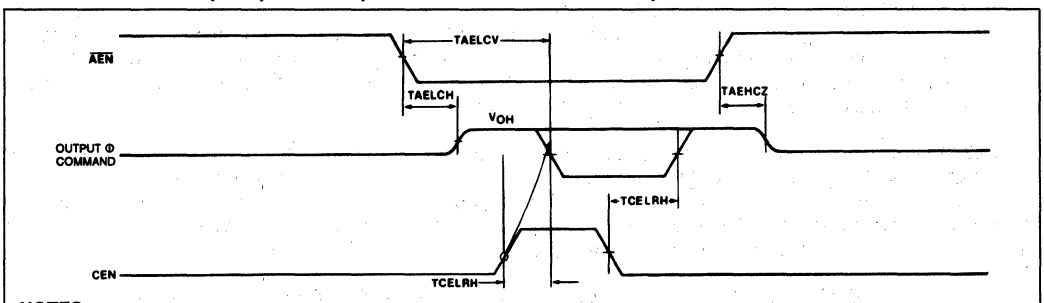
NOTES:
1. Address/Data Bus is shown only for reference purposes.
2. Leading edge of ALE and MCE is determined by the falling edge of CLK or Status going active, whichever occurs last.

WAVEFORMS (Continued)

DEN, PDEN QUALIFICATION TIMING



ADDRESS ENABLE (AEN) TIMING (3-STATE ENABLE/DISABLE)



NOTES:
CEN must be low or valid prior to T2 to prevent the command from being generated.
1. All Command outputs when in system bus mode, $\overline{MRDC}$, $\overline{MWTC}$, and $\overline{AMWC}$ when in I/O bus mode.

Data Sheet Revision Review

The following represents key differences between this and the previous 82C88. (Order No. 231199-004) data sheet. Please review.

1. Thls component has been redesigned from 5 MHz to 8 MHz for higher performance.

8237A
HIGH PERFORMANCE PROGRAMMABLE DMA CONTROLLER
(8237A, 8237A-4, 8237A-5)

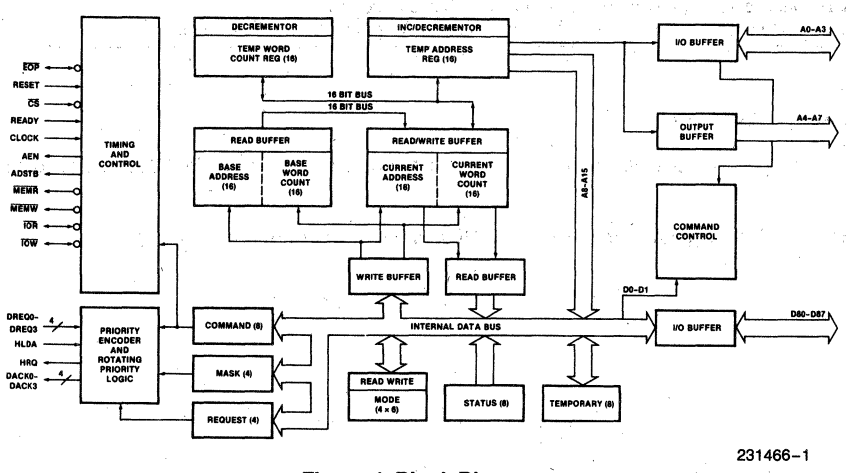
- **Enable/Disable Control of Individual DMA Requests**
- **Four Independent DMA Channels**
- **Independent Autoinitialization of All Channels**
- **Memory-to-Memory Transfers**
- **Memory Block Initialization**
- **Address Increment or Decrement**
- **High Performance: Transfers up to 1.6M Bytes/Second with 5 MHz 8237A-5**
- **Directly Expandable to Any Number of Channels**
- **End of Process Input for Terminating Transfers**
- **Software DMA Requests**
- **Independent Polarity Control for DREQ and DACK Signals**
- **Available in EXPRESS — Standard Temperature Range**
- **Available in 40-Lead Cerdip and Plastic Packages**

(See Packaging Spec, Order #231369)

The 8237A Multimode Direct Memory Access (DMA) Controller is a peripheral interface circuit for microprocessor systems. It is designed to improve system performance by allowing external devices to directly transfer information from the system memory. Memory-to-memory transfer capability is also provided. The 8237A offers a wide variety of programmable control features to enhance data throughput and system optimization and to allow dynamic reconfiguration under program control.

The 8237A is designed to be used in conjunction with an external 8-bit address latch. It contains four independent channels and may be expanded to any number of channels by cascading additional controller chips. The three basic transfer modes allow programmability of the types of DMA service by the user. Each channel can be individually programmed to Autoinitialize to its original condition following an End of Process ($\overline{EOP}$). Each channel has a full 64K address and word count capability.

The 8273A-4 and 8237A-5 are 4 MHz and 5 MHz versions of the standard 3 MHz 8237A respectively.



231466-1

Figure 1. Block Diagram

231466-2

Figure 2. Pin Configuration

October 1987
Order Number: 231466-003

Table 1. Pin Description

Symbol	Type	Name and Function
V_{CC}		**POWER:** +5V supply.
V_{SS}		**GROUND:** Ground.
CLK	I	**CLOCK INPUT:** Clock Input controls the internal operations of the 8237A and its rate of data transfers. The input may be driven at up to 3 MHz for the standard 8237A and up to 5 MHz for the 8237A-5.
$\overline{CS}$	I	**CHIP SELECT:** Chip Select is an active low input used to select the 8237A as an I/O device during the Idle cycle. This allows CPU communication on the data bus.
RESET	I	**RESET:** Reset is an active high input which clears the Command, Status, Request and Temporary registers. It also clears the first/last flip/flop and sets the Mask register. Following a Reset the device is in the Idle cycle.
READY	I	**READY:** Ready is an input used to extend the memory read and write pulses from the 8237A to accommodate slow memories or I/O peripheral devices. Ready must not make transitions during its specified setup/hold time.
HLDA	I	**HOLD ACKNOWLEDGE:** The active high Hold Acknowledge from the CPU indicates that it has relinquished control of the system busses.
DREQ0–DREQ3	I	**DMA REQUEST:** The DMA Request lines are individual asynchronous channel request inputs used by peripheral circuits to obtain DMA service. In fixed Priority, DREQ0 has the highest priority and DREQ3 has the lowest priority. A request is generated by activating the DREQ line of a channel. DACK will acknowledge the recognition of DREQ signal. Polarity of DREQ is programmable. Reset initializes these lines to active high. DREQ must be maintained until the corresponding DACK goes active.
DB0–DB7	I/O	**DATA BUS:** The Data Bus lines are bidirectional three-state signals connected to the system data bus. The outputs are enabled in the Program condition during the I/O Read to output the contents of an Address register, a Status register, the Temporary register or a Word Count register to the CPU. The outputs are disabled and the inputs are read during an I/O Write cycle when the CPU is programming the 8237A control registers. During DMA cycles the most significant 8 bits of the address are output onto the data bus to be strobed into an external latch by ADSTB. In memory-to-memory operations, data from the memory comes into the 8237A on the data bus during the read-from-memory transfer. In the write-to-memory transfer, the data bus outputs place the data into the new memory location.
$\overline{IOR}$	I/O	**I/O READ:** I/O Read is a bidirectional active low three-state line. In the Idle cycle, it is an input control signal used by the CPU to read the control registers. In the Active cycle, it is an output control signal used by the 8237A to access data from a peripheral during a DMA Write transfer.
$\overline{IOW}$	I/O	**I/O WRITE:** I/O Write is a bidirectional active low three-state line. In the Idle cycle, it is an input control signal used by the CPU to load information into the 8237A. In the Active cycle, it is an output control signal used by the 8237A to load data to the peripheral during a DMA Read transfer.

Table 1. Pin Description (Continued)

Symbol	Type	Name and Function
$\overline{EOP}$	I/O	**END OF PROCESS:** End of Process is an active low bidirectional signal. Information concerning the completion of DMA services is available at the bidirectional $\overline{EOP}$ pin. The 8237A allows an external signal to terminate an active DMA service. This is accomplished by pulling the $\overline{EOP}$ input low with an external $\overline{EOP}$ signal. The 8237A also generates a pulse when the terminal count (TC) for any channel is reached. This generates an $\overline{EOP}$ signal which is output through the $\overline{EOP}$ line. The reception of $\overline{EOP}$, either internal or external, will cause the 8237A to terminate the service, reset the request, and, if Autoinitialize is enabled, to write the base registers to the current registers of that channel. The mask bit and TC bit in the status word will be set for the currently active channel by $\overline{EOP}$ unless the channel is programmed for Autoinitialize. In that case, the mask bit remains unchanged. During memory-to-memory transfers, $\overline{EOP}$ will be output when the TC for channel 1 occurs. $\overline{EOP}$ should be tied high with a pull-up resistor if it is not used to prevent erroneous end of process inputs.
A0–A3	I/O	**ADDRESS:** The four least significant address lines are bidirectional three-state signals. In the Idle cycle they are inputs and are used by the CPU to address the register to be loaded or read. In the Active cycle they are outputs and provide the lower 4 bits of the output address.
A4–A7	O	**ADDRESS:** The four most significant address lines are three-state outputs and provide 4 bits of address. These lines are enabled only during the DMA service.
HRQ	O	**HOLD REQUEST:** This is the Hold Request to the CPU and is used to request control of the system bus. If the corresponding mask bit is clear, the presence of any valid DREQ causes 8237A to issue the HRQ.
DACK0–DACK3	O	**DMA ACKNOWLEDGE:** DMA Acknowledge is used to notify the individual peripherals when one has been granted a DMA cycle. The sense of these lines is programmable. Reset initializes them to active low.
AEN	O	**ADDRESS ENABLE:** Address Enable enables the 8-bit latch containing the upper 8 address bits onto the system address bus. AEN can also be used to disable other system bus drivers during DMA transfers. AEN is active HIGH.
ADSTB	O	**ADDRESS STROBE:** The active high, Address Strobe is used to strobe the upper address byte into an external latch.
$\overline{MEMR}$	O	**MEMORY READ:** The Memory Read signal is an active low three-state output used to access data from the selected memory location during a DMA Read or a memory-to-memory transfer.
$\overline{MEMW}$	O	**MEMORY WRITE:** The Memory Write is an active low three-state output used to write data to the selected memory location during a DMA Write or a memory-to-memory transfer.
PIN5	I	**PIN5:** This pin should always be at a logic HIGH level. An internal pull-up resistor will establish a logic high when the pin is left floating. It is recommended however, that PIN5 be connected to V_{CC}.

FUNCTIONAL DESCRIPTION

The 8237A block diagram includes the major logic blocks and all of the internal registers. The data interconnection paths are also shown. Not shown are the various control signals between the blocks. The 8237A contains 344 bits of internal memory in the form of registers. Figure 3 lists these registers by name and shows the size of each. A detailed description of the registers and their functions can be found under Register Description.

Name	Size	Number
Base Address Registers	16 bits	4
Base Word Count Registers	16 bits	4
Current Address Registers	16 bits	4
Current Word Count Registers	16 bits	4
Temporary Address Register	16 bits	1
Temporary Word Count Register	16 bits	1
Status Register	8 bits	1
Command Register	8 bits	1
Temporary Register	8 bits	1
Mode Registers	6 bits	4
Mask Register	4 bits	1
Request Register	4 bits	1

Figure 3. 8237A Internal Registers

The 8237A contains three basic blocks of control logic. The Timing Control block generates internal timing and external control signals for the 8237A. The Program Command Control block decodes the various commands given to the 8237A by the microprocessor prior to servicing a DMA Request. It also decodes the Mode Control word used to select the type of DMA during the servicing. The Priority Encoder block resolves priority contention between DMA channels requesting service simultaneously.

The Timing Control block derives internal timing from the clock input. In 8237A systems, this input will usually be the $\phi 2$ TTL clock from an 8224 or CLK from an 8085AH or 8284A. 33% duty cycle clock generators, however, may not meet the clock high time requirement of the 8237A of the same frequency. For example, 82C84A-5 CLK output violates the clock high time requirement of 8237A-5. In this case 82C84A CLK can simply be inverted to meet 8237A-5 clock high and low time requirements. For 8085AH-2 systems above 3.9 MHz, the 8085 CLK(OUT) does not satisfy 8237A-5 clock LOW and HIGH time requirements. In this case, an external clock should be used to drive the 8237A-5.

DMA OPERATION

The 8237A is designed to operate in two major cycles. These are called Idle and Active cycles. Each device cycle is made up of a number of states. The 8237A can assume seven separate states, each composed of one full clock period. State I (SI) is the inactive state. It is entered when the 8237A has no valid DMA requests pending. While in SI, the DMA controller is inactive but may be in the Program Condition, being programmed by the processor. State S0 (S0) is the first state of a DMA service. The 8237A has requested a hold but the processor has not yet returned an acknowledge. The 8237A may still be programmed until it receives HLDA from the CPU. An acknowledge from the CPU will signal that DMA transfers may begin. S1, S2, S3 and S4 are the working states of the DMA service. If more time is needed to complete a transfer than is available with normal timing, wait states (SW) can be inserted between S2 or S3 and S4 by the use of the Ready line on the 8237A. Note that the data is transferred directly from the I/O device to memory (or vice versa) with $\overline{IOR}$ and $\overline{MEMW}$ (or $\overline{MEMR}$ and $\overline{IOW}$) being active at the same time. The data is not read into or driven out of the 8237A in I/O-to-memory or memory-to-I/O DMA transfers.

Memory-to-memory transfers require a read-from and a write-to-memory to complete each transfer. The states, which resemble the normal working states, use two digit numbers for identification. Eight states are required for a single transfer. The first four states (S11, S12, S13, S14) are used for the read-from-memory half and the last four states (S21, S22, S23, S24) for the write-to-memory half of the transfer.

IDLE CYCLE

When no channel is requesting service, the 8237A will enter the Idle cycle and perform "SI" states. In this cycle the 8237A will sample the DREQ lines every clock cycle to determine if any channel is requesting a DMA service. The device will also sample $\overline{CS}$, looking for an attempt by the microprocessor to write or read the internal registers of the 8237A. When $\overline{CS}$ is low and HLDA is low, the 8237A enters the Program Condition. The CPU can now establish, change or inspect the internal definition of the part by reading from or writing to the internal registers. Address lines A0–A3 are inputs to the device and select which registers will be read or written. The $\overline{IOR}$ and $\overline{IOW}$ lines are used to select and time reads or writes. Due to the number and size of the internal registers, an internal flip-flop is used to generate an additional bit of address. This bit is used to determine the upper or lower byte of the 16-bit Address and Word Count registers. The flip-flop is reset by Master Clear or Reset. A separate software command can also reset this flip-flop.

Special software commands can be executed by the 8237A in the Program Condition. These commands are decoded as sets of addresses with the $\overline{CS}$ and $\overline{IOW}$. The commands do not make use of the data bus. Instructions include Clear First/Last Flip-Flop and Master Clear.

ACTIVE CYCLE

When the 8237A is in the Idle cycle and a non-masked channel requests a DMA service, the device will output an HRQ to the microprocessor and enter the Active cycle. It is in this cycle that the DMA service will take place, in one of four modes:

Single Transfer Mode—In Single Transfer mode the device is programmed to make one transfer only. The word count will be decremented and the address decremented or incremented following each transfer. When the word count "rolls over" from zero to FFFFH, a Terminal Count (TC) will cause an Autoinitialize if the channel has been programmed to do so.

DREQ must be held active until DACK becomes active in order to be recognized. If DREQ is held active throughout the single transfer, HRQ will go inactive and release the bus to the system. It will again go active and, upon receipt of a new HLDA, another single transfer will be performed. In 8080A, 8085AH, 8088, or 8086 system, this will ensure one full machine cycle execution between DMA transfers. Details of timing between the 8237A and other bus control protocols will depend upon the characteristics of the microprocessor involved.

Block Transfer Mode—In Block Transfer mode the device is activated by DREQ to continue making transfers during the service until a TC, caused by word count going to FFFFH, or an external End of Process ($\overline{EOP}$) is encountered. DREQ need only be held active until DACK becomes active. Again, an Autoinitialization will occur at the end of the service if the channel has been programmed for it.

Demand Transfer Mode—In Demand Transfer mode the device is programmed to continue making transfers until a TC or external $\overline{EOP}$ is encountered or until DREQ goes inactive. Thus transfers may continue until the I/O device has exhausted its data capacity. After the I/O device has had a chance to catch up, the DMA service is re-established by means of a DREQ. During the time between services when the microprocessor is allowed to operate, the intermediate values of address and word count are stored in the 8237A Current Address and Current Word Count registers. Only an $\overline{EOP}$ can cause an Autoinitialize at the end of the service. $\overline{EOP}$ is generated either by TC or by an external signal. DREQ has to be low before S4 to prevent another Transfer.

Cascade Mode—This mode is used to cascade more than one 8237A together for simple system expansion. The HRQ and HLDA signals from the additional 8237A are connected to the DREQ and DACK signals of a channel of the initial 8237A. This allows the DMA requests of the additional device to propagate through the priority network circuitry of the preceding device. The priority chain is preserved and the new device must wait for its turn to acknowledge requests. Since the cascade channel of the initial 8237A is used only for prioritizing the additional device, it does not output any address or control

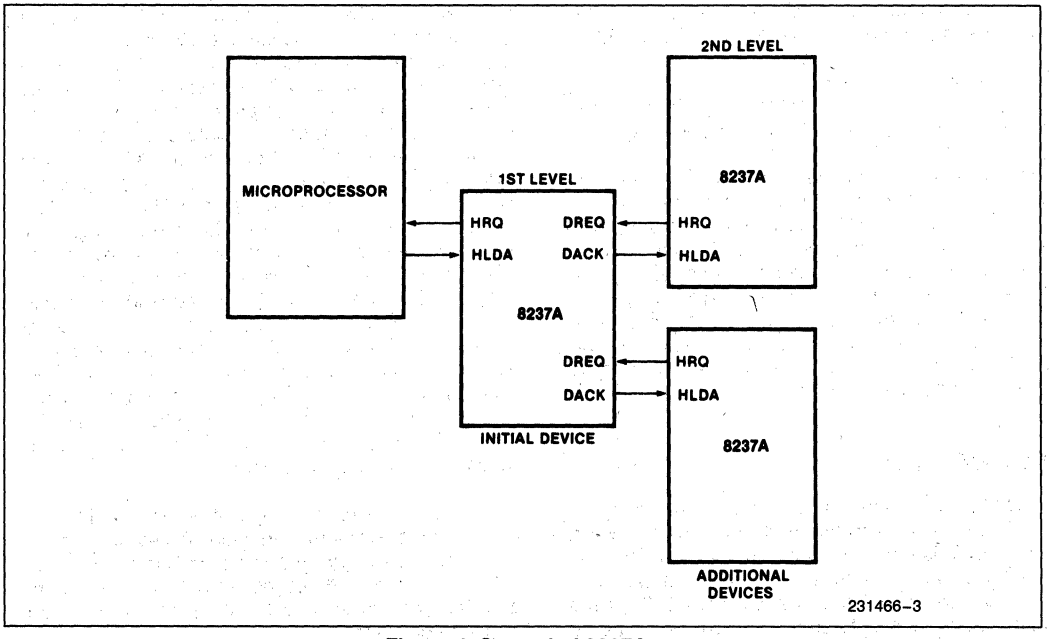


Figure 4. Cascaded 8237As

signals of its own. These could conflict with the outputs of the active channel in the added device. The 8237A will respond to DREQ and DACK but all other outputs except HRQ will be disabled. The ready input is ignored.

Figure 4 shows two additional devices cascaded into an initial device using two of the previous channels. This forms a two level DMA system. More 8237As could be added at the second level by using the remaining channels of the first level. Additional devices can also be added by cascading into the channels of the second level device, forming a third level.

TRANSFER TYPES

Each of the three active transfer modes can perform three different types of transfers. These are Read, Write and Verify. Write transfers move data from an I/O device to the memory by activating $\overline{MEMW}$ and $\overline{IOR}$. Read transfers move data from memory to an I/O device by activating $\overline{MEMR}$ and $\overline{IOW}$. Verify transfers are pseudo transfers. The 8237A operates as in Read or Write transfers generating addresses, and responding to EOP, etc. However, the memory and I/O control lines all remain inactive. The ready input is ignored in verify mode.

Memory-to-Memory—To perform block moves of data from one memory address space to another with a minimum of program effort and time, the 8237A includes a memory-to-memory transfer feature. Programming a bit in the Command register selects channels 0 and 1 to operate as memory-to-memory transfer channels. The transfer is initiated by setting the software DREQ for channel 0. The 8237A requests a DMA service in the normal manner. After HLDA is true, the device, using four state transfers in Block Transfer mode, reads data from the memory. The channel 0 Current Address register is the source for the address used and is decremented or incremented in the normal manner. The data byte read from the memory is stored in the 8237A internal Temporary register. Channel 1 then performs a four-state transfer of the data from the Temporary register to memory using the address in its Current Address register and incrementing or decrementing it in the normal manner. The channel 1 current Word Count is decremented. When the word count of channel 1 goes to FFFFH, a TC is generated causing an $\overline{EOP}$ output terminating the service.

Channel 0 may be programmed to retain the same address for all transfers. This allows a single word to be written to a block of memory.

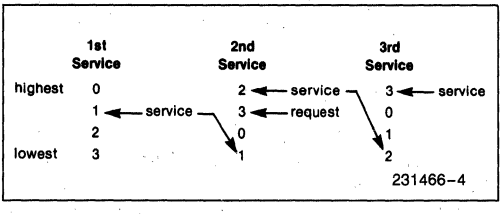
The 8237A will respond to external $\overline{EOP}$ signals during memory-to-memory transfers. Data comparators in block search schemes may use this input to terminate the service when a match is found. The timing of memory-to-memory transfers is found in Figure 12. Memory-to-memory operations can be detected as an active AEN with no DACK outputs.

Autoinitialize—By programming a bit in the Mode register, a channel may be set up as an Autoinitialize channel. During Autoinitialize initialization, the original values of the Current Address and Current Word Count registers are automatically restored from the Base Address and Base Word count registers of that channel following $\overline{EOP}$. The base registers are loaded simultaneously with the current registers by the microprocessor and remain unchanged throughout the DMA service. The mask bit is not altered when the channel is in Autoinitialize. Following Autoinitialize the channel is ready to perform another DMA service, without CPU intervention, as soon as a valid DREQ is detected. In order to Autoinitialize both channels in a memory-to-memory transfer, both word counts should be programmed identically. If interrupted externally, $\overline{EOP}$ pulses should be applied in both bus cycles.

Priority—The 8237A has two types of priority encoding available as software selectable options. The first is Fixed Priority which fixes the channels in priority order based upon the descending value of their number. The channel with the lowest priority is 3 followed by 2, 1 and the highest priority channel, 0. After the recognition of any one channel for service, the other channels are prevented from interfering with that service until it is completed.

After completion of a service, HRQ will go inactive and the 8237A will wait for HLDA to go low before activating HRQ to service another channel.

The second scheme is Rotating Priority. The last channel to get service becomes the lowest priority channel with the others rotating accordingly.



With Rotating Priority in a single chip DMA system, any device requesting service is guaranteed to be recognized after no more than three higher priority services have occurred. This prevents any one channel from monopolizing the system.

Compressed Timing—In order to achieve even greater throughput where system characteristics permit, the 8237A can compress the transfer time to two clock cycles. From Figure 11 it can be seen that state S3 is used to extend the access time of the read pulse. By removing state S3, the read pulse width is made equal to the write pulse width and a transfer consists only of state S2 to change the address and state S4 to perform the read/write. S1 states will still occur when A8–A15 need updating (see Address Generation). Timing for compressed transfers is found in Figure 14.

Address Generation—In order to reduce pin count, the 8237A multiplexes the eight higher order address bits on the data lines. State S1 is used to output the higher order address bits to an external latch from which they may be placed on the address bus. The falling edge of Address Strobe (ADSTB) is used to load these bits from the data lines to the latch. Address Enable (AEN) is used to enable the bits onto the address bus through a three-state enable. The lower order address bits are output by the 8237A directly. Lines A0–A7 should be connected to the address bus. Figure 11 shows the time relationships between CLK, AEN, ADSTB, DB0–DB7 and A0–A7.

During Block and Demand Transfer mode services, which include multiple transfers, the addresses generated will be sequential. For many transfers the data held in the external address latch will remain the same. This data need only change when a carry or borrow from A7 to A8 takes place in the normal sequence of addresses. To save time and speed transfers, the 8237A executes S1 states only when updating of A8–A15 in the latch is necessary. This means for long services, S1 states and Address Strobes may occur only once every 256 transfers, a savings of 255 clock cycles for each 256 transfers.

REGISTER DESCRIPTION

Current Address Register—Each channel has a 16-bit Current Address register. This register holds the value of the address used during DMA transfers. The address is automatically incremented or decremented after each transfer and the intermediate values of the address are stored in the Current Address register during the transfer. This register is written or read by the microprocessor in successive 8-bit bytes. It may also be reinitialized by an Autoinitialize back to its original value. Autoinitialize takes place only after an $\overline{EOP}$.

Current Word Register—Each channel has a 16-bit Current Word Count register. This register determines the number of transfers to be performed. The actual number of transfers will be one more than the number programmed in the Current Word Count register (i.e., programming a count of 100 will result in 101 transfers). The word count is decremented after each transfer. The intermediate value of the word count is stored in the register during the transfer. When the value in the register goes from zero to FFFFH, a TC will be generated. This register is loaded or read in successive 8-bit bytes by the microprocessor in the Program Condition. Following the end of a DMA service it may also be reinitialized by an Autoinitialization back to its original value. Autoinitialize can occur only when an $\overline{EOP}$ occurs. If it is not Autoinitialized, this register will have a count of FFFFH after TC.

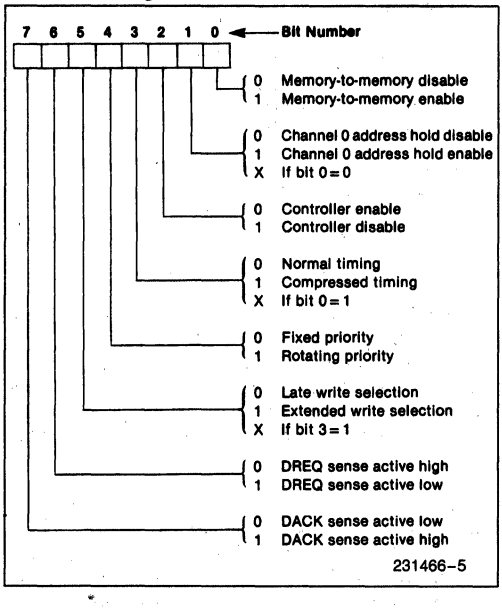
Base Address and Base Word Count Registers—Each channel has a pair of Base Address and Base Word Count registers. These 16-bit registers store the original value of their associated current registers. During Autoinitialize these values are used to restore the current registers to their original values. The base registers are written simultaneously with their corresponding current register in 8-bit bytes in the Program Condition by the microprocessor. These registers cannot be read by the microprocessor.

Command Register—This 8-bit register controls the operation of the 8237A. It is programmed by the microprocessor in the Program Condition and is cleared by Reset or a Master Clear instruction. The following table lists the function of the command bits. See Figure 6 for address coding.

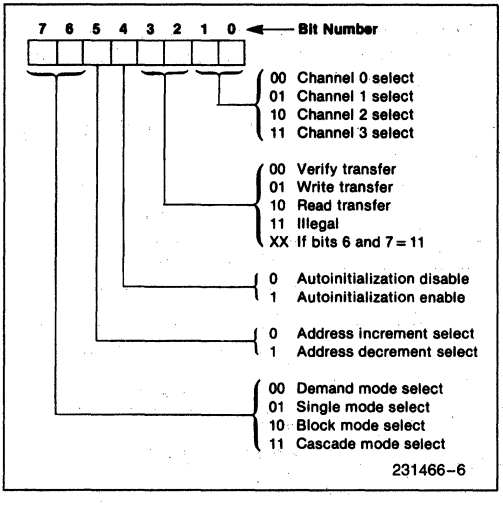
Mode Register—Each channel has a 6-bit Mode register associated with it. When the register is being written to by the microprocessor in the Program Condition, bits 0 and 1 determine which channel Mode register is to be written.

Request Register—The 8237A can respond to requests for DMA service which are initiated by software as well as by a DREQ. Each channel has a request bit associated with it in the 4-bit Request register. These are non-maskable and subject to prioritization by the Priority Encoder network. Each register bit is set or reset separately under software control or is cleared upon generation of a TC or external $\overline{EOP}$. The entire register is cleared by a Reset. To set or reset a bit, the software loads the proper form of the data word. See Figure 5 for register address coding. In order to make a software request, the channel must be in Block Mode.

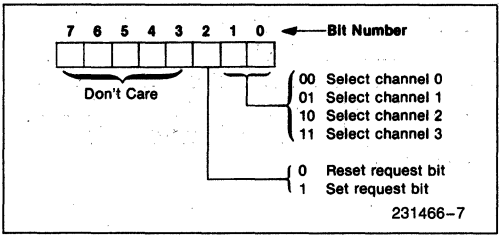
Command Register



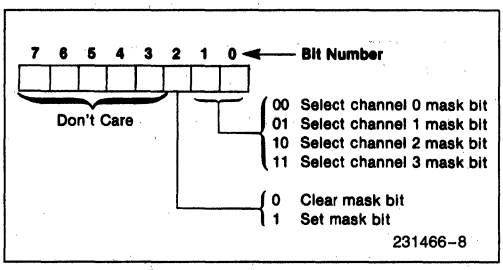
Mode Register



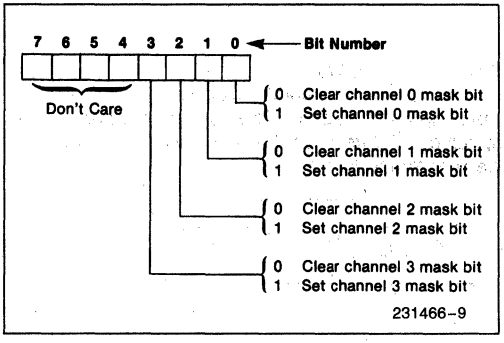
Request Register



Mask Register—Each channel has associated with it a mask bit which can be set to disable the incoming DREQ. Each mask bit is set when its associated channel produces an $\overline{EOP}$ if the channel is not programmed for Autoinitialize. Each bit of the 4-bit Mask register may also be set or cleared separately under software control. The entire register is also set by a Reset. This disables all DMA requests until a clear Mask register instruction allows them to occur. The instruction to separately set or clear the mask bits is similar in form to that used with the Request register. See Figure 5 for instruction addressing.



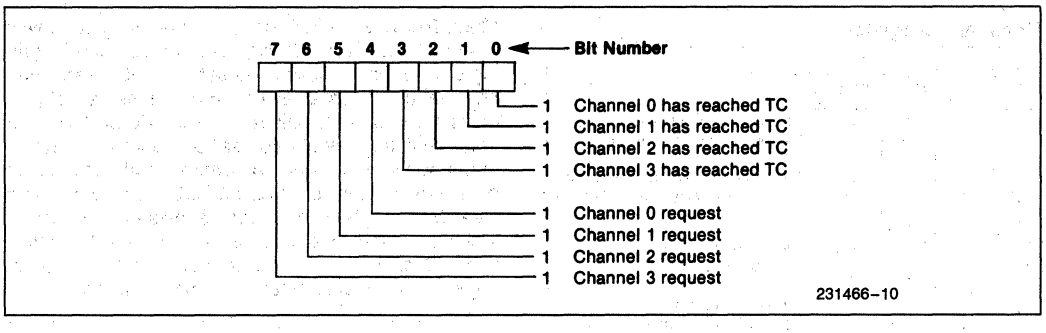
All four bits of the Mask register may also be written with a single command.



Register	Operation	Signals						
		$\overline{CS}$	$\overline{IOR}$	$\overline{IOW}$	A3	A2	A1	A0
Command	Write	0	1	0	1	0	0	0
Mode	Write	0	1	0	1	0	1	1
Request	Write	0	1	0	1	0	0	1
Mask	Set/Reset	0	1	0	1	0	1	0
Mask	Write	0	1	0	1	1	1	1
Temporary	Read	0	0	1	1	1	0	1
Status	Read	0	0	1	1	0	0	0

Figure 5. Definition of Register Codes

Status Register—The Status register is available to be read out of the 8237A by the microprocessor. It contains information about the status of the devices at this point. This information includes which channels have reached a terminal count and which chan-



nels have pending DMA requests. Bits 0–3 are set every time a TC is reached by that channel or an external $\overline{EOP}$ is applied. These bits are cleared upon Reset and on each Status Read. Bits 4–7 are set whenever their corresponding channel is requesting service.

Temporary Register—The Temporary register is used to hold data during memory-to-memory transfers. Following the completion of the transfers, the last word moved can be read by the microprocessor in the Program Condition. The Temporary register always contains the last byte transferred in the previous memory-to-memory operation, unless cleared by a Reset.

Software Commands—These are additional special software commands which can be executed in the Program Condition. They do not depend on any specific bit pattern on the data bus. The three software commands are:

Clear First/Last Flip-Flop: This command must be executed prior to writing or reading new address or word count information to the 8237A. This initializes the flip-flop to a known state so that subsequent accesses to register contents by the microprocessor will address upper and lower bytes in the correct sequence.

Master Clear: This software instruction has the same effect as the hardware Reset. The Command, Status, Request, Temporary, and Internal First/Last Flip-Flop registers are cleared and the Mask register is set. The 8237A will enter the Idle cycle.

Clear Mask Register: This command clears the mask bits of all four channels, enabling them to accept DMA requests.

Figure 6 lists the address codes for the software commands.

Signals						Operation
A3	A2	A1	A0	$\overline{IOR}$	$\overline{IOW}$	
1	0	0	0	0	1	Read Status Register
1	0	0	0	1	0	Write Command Register
1	0	0	1	0	1	Illegal
1	0	0	1	1	0	Write Request Register
1	0	1	0	0	1	Illegal
1	0	1	0	1	0	Write Single Mask Register Bit
1	0	1	1	0	1	Illegal
1	0	1	1	1	0	Write Mode Register
1	1	0	0	0	1	Illegal
1	1	0	0	1	0	Clear Byte Pointer Flip/Flop
1	1	0	1	0	1	Read Temporary Register
1	1	0	1	1	0	Master Clear
1	1	1	0	0	1	Illegal
1	1	1	0	1	0	Clear Mask Register
1	1	1	1	0	1	Illegal
1	1	1	1	1	0	Write All Mask Register Bits

Figure 6. Software Command Codes

Channel	Register	Operation	Signals							Internal Flip-Flop	Data Bus DB0–DB7
			$\overline{CS}$	$\overline{IOR}$	$\overline{IOW}$	A3	A2	A1	A0		
0	Base and Current Address	Write	0	1	0	0	0	0	0	0	A0–A7
			0	1	0	0	0	0	0	1	A8–A15
	Current Address	Read	0	0	1	0	0	0	0	0	A0–A7
			0	0	1	0	0	0	0	1	A8–A15
	Base and Current Word Count	Write	0	1	0	0	0	0	1	0	W0–W7
			0	1	0	0	0	0	1	1	W8–W15
	Current Word Count	Read	0	0	1	0	0	0	1	0	W0–W7
			0	0	1	0	0	0	1	1	W8–W15
1	Base and Current Address	Write	0	1	0	0	0	1	0	0	A0–A7
			0	1	0	0	0	1	0	1	A8–A15
	Current Address	Read	0	0	1	0	0	1	0	0	A0–A7
			0	0	1	0	0	1	0	1	A8–A15
	Base and Current Word Count	Write	0	1	0	0	0	1	1	0	W0–W7
			0	1	0	0	0	1	1	1	W8–W15
	Current Word Count	Read	0	0	1	0	0	1	1	0	W0–W7
			0	0	1	0	0	1	1	1	W8–W15
2	Base and Current Address	Write	0	1	0	0	1	0	0	0	A0–A7
			0	1	0	0	1	0	0	1	A8–A15
	Current Address	Read	0	0	1	0	1	0	0	0	A0–A7
			0	0	1	0	1	0	0	1	A8–A15
	Base and Current Word Count	Write	0	1	0	0	1	0	1	0	W0–W7
			0	1	0	0	1	0	1	1	W8–W15
	Current Word Count	Read	0	0	1	0	1	0	1	0	W0–W7
			0	0	1	0	1	0	1	1	W8–W15
3	Base and Current Address	Write	0	1	0	0	1	1	0	0	A0–A7
			0	1	0	0	1	1	0	1	A8–A15
	Current Address	Read	0	0	1	0	1	1	0	0	A0–A7
			0	0	1	0	1	1	0	1	A8–A15
	Base and Current Word Count	Write	0	1	0	0	1	1	1	0	W0–W7
			0	1	0	0	1	1	1	1	W8–W15
	Current Word Count	Read	0	0	1	0	1	1	1	0	W0–W7
			0	0	1	0	1	1	1	1	W8–W15

Figure 7. Word Count and Address Register Command Codes

PROGRAMMING

The 8237A will accept programming from the host processor any time that HLDA is inactive; this is true even if HRQ is active. The responsibility of the host is to assure that programming and HLDA are mutually exclusive. Note that a problem can occur if a DMA request occurs, on an unmasked channel while the 8237A is being programmed. For instance, the CPU may be starting to reprogram the two byte Address register of channel 1 when channel 1 receives a DMA request. If the 8237A is enabled (bit 2 in the command register is 0) and channel 1 is unmasked, a DMA service will occur after only one byte of the Address register has been reprogrammed. This can be avoided by disabling the controller (setting bit 2 in the command register) or masking the channel before programming any other registers. Once the programming is complete, the controller can be enabled/unmasked.

After power-up it is suggested that all internal locations, especially the Mode registers, be loaded with some valid value. This should be done even if some channels are unused. An invalid mode may force all control signals to go active at the same time.

APPLICATION INFORMATION (Note 1)

Figure 8 shows a convenient method for configuring a DMA system with the 8237A controller and an 8080A/8085AH microprocessor system. The multimode DMA controller issues a HRQ to the processor whenever there is at least one valid DMA request from a peripheral device. When the processor replies with a HLDA signal, the 8237A takes control of the address bus, the data bus and the control bus. The address for the first transfer operation comes out in two bytes—the least significant 8 bits on the eight address outputs and the most significant 8 bits on the data bus. The contents of the data bus are then latched into an 8-bit latch to complete the full 16 bits of the address bus. The 8282 is a high speed, 8-bit, three-state latch in a 20-pin package. After the initial transfer takes place, the latch is updated only after a carry or borrow is generated in the least significant address byte. Four DMA channels are provided when one 8237A is used.

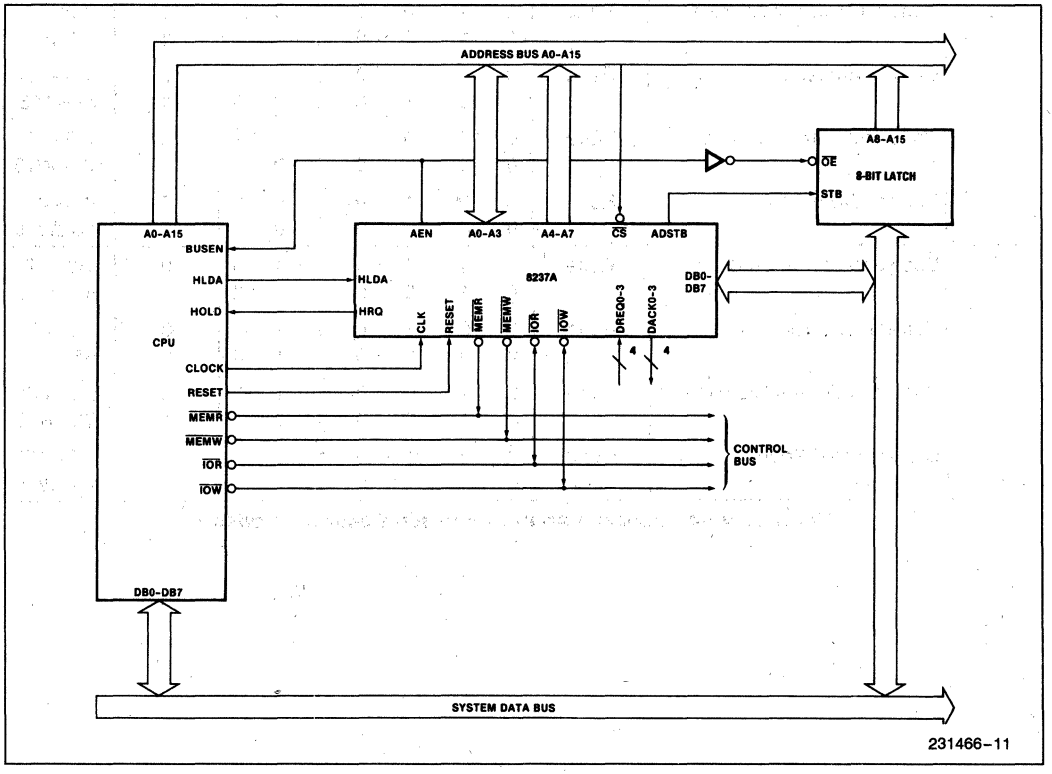


Figure 8. 8237A System Interface

NOTE:
1. See Application Note AP-67 for 8086 design information.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature under Bias .....0°C to 70°C
Case Temperature .....0°C to +75°C
Storage Temperature .....−65°C to +150°C
Voltage on Any Pin with
Respect to Ground.....−0.5V to +7V
Power Dissipation.....1.5 Watt

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, T_{CASE} = 0°C to 75°C, V_{CC} = +5.0V ±5%, GND = 0V

Symbol	Parameter	Min	Typ (Note 1)	Max	Unit	Test Conditions
V_{OH}	Output High Voltage	2.4			V	I_{OH} = −200 μA
		3.3			V	I_{OH} = −100 μA (HRQ Only)
V_{OL}	Output LOW Voltage			0.40	V	I_{OL} = 3.2 mA
V_{IH}	Input HIGH Voltage	2.0		V_{CC} + 0.5	V	
V_{IL}	Input LOW Voltage	−0.5		0.8	V	
I_{LI}	Input Load Current			±10	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{LO}	Output Leakage Current			±10	μA	0.45V ≤ V_{OUT} ≤ V_{CC}
I_{CC}	V_{CC} Supply Current		110	130	mA	T_A = +25°C
			130	150	mA	T_A = 0°C
C_O	Output Capacitance		4	8	pF	fc = 1.0 MHz, Inputs = 0V
C_I	Input Capacitance		8	15	pF	
C_{IO}	I/O Capacitance		10	18	pF	

NOTE:
1. Typical values are for T_A = 25°C, nominal supply voltage and nominal processing parameters.

A.C. CHARACTERISTICS—DMA (MASTER) MODE

T_A = 0°C to 70°C, T_{CASE} = 0°C to 75°C, V_{CC} = +5V ±5%, GND = 0V

Symbol	Parameter	8237A		8237A-4		8237A-5		Unit
		Min	Max	Min	Max	Min	Max	
TAEL	AEN HIGH from CLK LOW (S1) Delay Time		300		225		200	ns
TAET	AEN LOW from CLK HIGH (SI) Delay Time		200		150		130	ns
TAFAB	ADR Active to Float Delay from CLK HIGH		150		120		90	ns
TAFC	$\overline{READ}$ or $\overline{WRITE}$ Float from CLK HIGH		150		120		120	ns
TAFDB	DB Active to Float Delay from CLK HIGH		250		190		170	ns
TAHR	ADR from $\overline{READ}$ HIGH Hold Time	TCY-100		TCY-100		TCY-100		ns
TAHS	DB from ADSTB LOW Hold Time	40		40		30		ns
TAHW	ADR from $\overline{WRITE}$ HIGH Hold Time	TCY-50		TCY-50		TCY-50		ns
TAK	DACK Valid from CLK LOW Delay Time (Note 1)		250		220		170	ns
	$\overline{EOP}$ HIGH from CLK HIGH Delay Time (Note 2)		250		190		170	ns
	$\overline{EOP}$ LOW from CLK HIGH Delay Time		250		190		170	ns
TASM	ADR Stable from CLK HIGH		250		190		170	ns
TASS	DB to ADSTB LOW Setup Time	100		100		100		ns
TCH	Clock High Time (Transitions ≤ 10 ns)	120		100		80		ns
TCL	Clock LOW Time (Transitions ≤ 10 ns)	150		110		68		ns
TCY	CLK Cycle Time	320		250		200		ns
TDCL	CLK HIGH to $\overline{READ}$ or $\overline{WRITE}$ LOW Delay (Note 3)		270		200		190	ns
TDCTR	$\overline{READ}$ HIGH from CLK HIGH (S4) Delay Time (Note 3)		270		210		190	ns
TDCTW	$\overline{WRITE}$ HIGH from CLK HIGH (S4) Delay Time (Note 3)		200		150		130	ns
TDQ1	HRQ Valid from CLK HIGH Delay Time (Note 4)		160		120		120	ns
TDQ2			250		190		120	ns
TEPS	$\overline{EOP}$ LOW from CLK LOW Setup Time	60		45		40		ns
TEPW	$\overline{EOP}$ Pulse Width	300		225		220		ns
TFAAB	ADR Float to Active Delay from CLK HIGH		250		190		170	ns
TFAC	$\overline{READ}$ or $\overline{WRITE}$ Active from CLK HIGH		200		150		150	ns
TFADB	DB Float to Active Delay from CLK HIGH		300		225		200	ns
THS	HLDA Valid to CLK HIGH Setup Time	100		75		75		ns
TIDH	Input Data from $\overline{MEMR}$ HIGH Hold Time	0		0		0		ns
TIDS	Input Data to $\overline{MEMR}$ HIGH Setup Time	250		190		170		ns
TODH	Output Data from $\overline{MEMW}$ HIGH Hold Time	20		20		10		ns
TODV	Output Data Valid to $\overline{MEMW}$ HIGH	200		125		125		ns
TQS	DREQ to CLK LOW (SI, S4) Setup Time (Note 1)	0		0		0		ns
TRH	CLK to READY LOW Hold Time	20		20		20		ns
TRS	READY to CLK LOW Setup Time	100		60		60		ns
TSTL	ADSTB HIGH from CLK HIGH Delay Time		200		150		130	ns
TSTT	ADSTB LOW from CLK HIGH Delay Time		140		110		90	ns

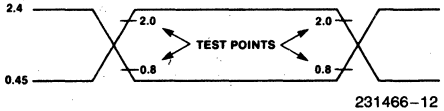
A.C. CHARACTERISTICS—PERIPHERAL (SLAVE) MODE

T_A = 0°C to 70°C, T_{CASE} = 0°C to 75°C, V_{CC} = +5V ±5%, GND = 0V

Symbol	Parameter	8237A		8237A-4		8237A-5		Unit
		Min	Max	Min	Max	Min	Max	
TAR	ADR Valid or $\overline{CS}$ LOW to $\overline{READ}$ LOW	50		50		50		ns
TAW	ADR Valid to $\overline{WRITE}$ HIGH Setup Time	200		150		130		ns
TCW	CS LOW to $\overline{WRITE}$ HIGH Setup Time	200		150		130		ns
TDW	Data Valid to $\overline{WRITE}$ HIGH Setup Time	200		150		130		ns
TRA	ADR or CS Hold from $\overline{READ}$ HIGH	0		0		0		ns
TRDE	Data Access from $\overline{READ}$ LOW (Note 5)		200		200		140	ns
TRDF	DB Float Delay from $\overline{READ}$ HIGH	20	100	20	100	0	70	ns
TRSTD	Power Supply HIGH to RESET LOW Setup Time	500		500		500		ns
TRSTS	RESET to First $\overline{IOWR}$	2TCY		2TCY		2TCY		ns
TRSTW	RESET Pulse Width	300		300		300		ns
TRW	$\overline{READ}$ Width	300		250		200		ns
TWA	ADR from $\overline{WRITE}$ HIGH Hold Time	20		20		20		ns
TWC	CS HIGH from $\overline{WRITE}$ HIGH Hold Time	20		20		20		ns
TWD	Data from $\overline{WRITE}$ HIGH Hold Time	30		30		30		ns
TWWS	Write Width	200		200		160		ns
TWR	End of Write to End of Read in DMA Transfer	0		0		0		ns

NOTES:
1. DREQ and DACK signals may be active high or active low. Timing diagrams assume the active high mode.
2. $\overline{EOP}$ is an open collector output. This parameter assumes the presence of a 2.2K pullup to V_{CC}.
3. The net $\overline{IOW}$ or $\overline{MEMW}$ Pulse width for normal write will be TCY−100 ns and for extended write will be 2TCY−100 ns. The net $\overline{IOR}$ or $\overline{MEMR}$ pulse width for normal read will be 2TCY−50 ns and for compressed read will be TCY−50 ns.
4. TDQ is specified for two different output HIGH levels. TDQ1 is measured at 2.0V. TDQ2 is measured at 3.3V. The value for TDQ2 assumes an external 3.3 KΩ pull-up resistor connected from HRQ to V_{CC}.
5. Output Loading on the Data Bus is 1 TTL Gate plus 100 pF capacitance.

A.C. TESTING INPUT/OUTPUT WAVEFORM

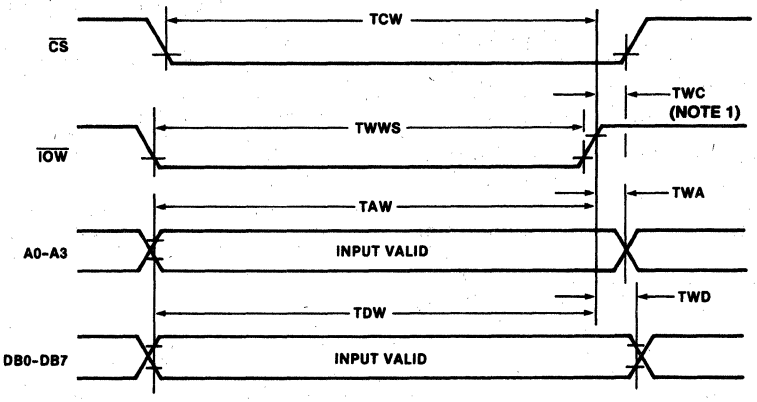


231466-12

A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0." Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0." Input timing parameters assume transition times of 20 ns or less. Waveform measurement points for both input and output signals are 2.0V for HIGH and 0.8V for LOW, unless otherwise noted.

WAVEFORMS

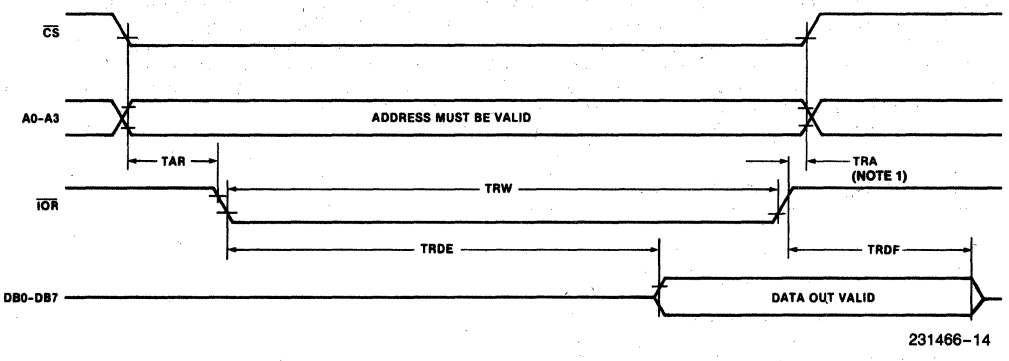
SLAVE MODE WRITE TIMING



NOTE:
1. Successive read and/or write operations by the external processor to program or examine the controller must be timed to allow at least 600 ns for the 8237A, at least 500 ns for the 8237A-4, and at least 400 ns for the 8237A-5 as recovery time between active read or write pulses. The same recovery time is needed between an active read or write pulse followed by a DMA transfer.

Figure 9. Slave Mode Write

SLAVE MODE READ TIMING

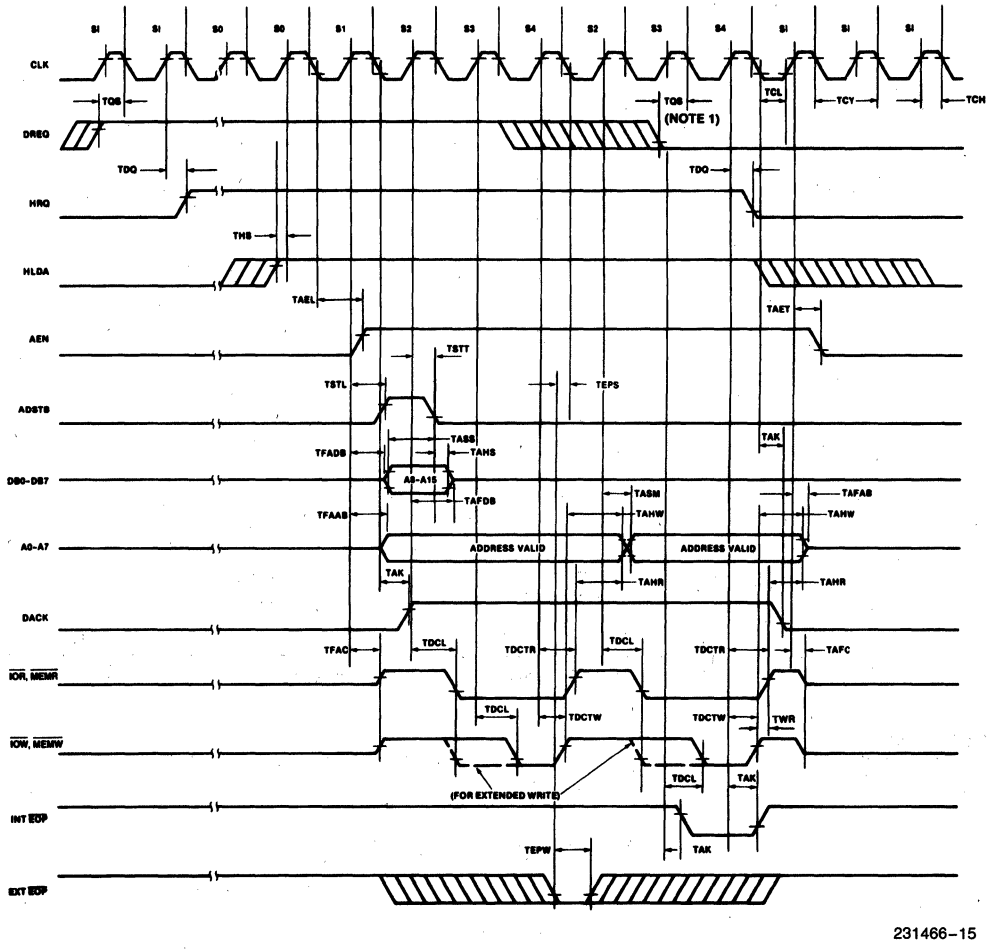


NOTE:
1. Successive read and/or write operations by the external processor to program or examine the controller must be timed to allow at least 600 ns for the 8237A, at least 500 ns for the 8237A-4, and at least 400 ns for the 8237A-5 as recovery time between active read or write pulses. The same recovery time is needed between an active read or write pulse followed by a DMA transfer.

Figure 10. Slave Mode Read

WAVEFORMS (Continued)

DMA TRANSFER TIMING



231466-15

NOTE:
1. DREQ should be held active until DACK is returned.

Figure 11. DMA Transfer

WAVEFORMS (Continued)

MEMORY-TO-MEMORY TRANSFER TIMING

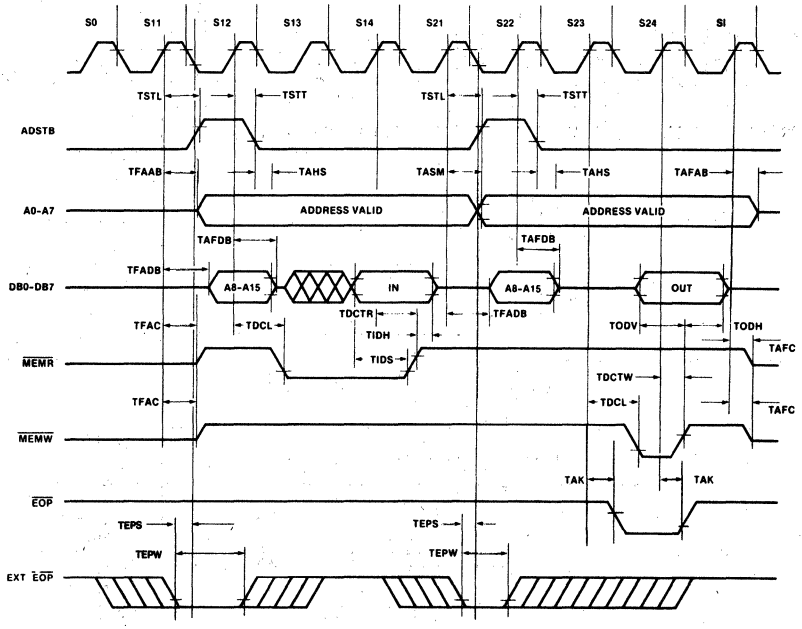


Figure 12. Memory-to-Memory Transfer

READY TIMING

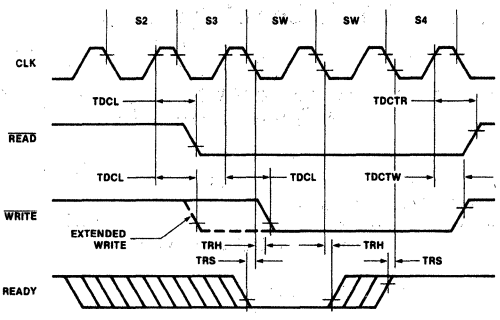


Figure 13. Ready

WAVEFORMS (Continued)

COMPRESSED TRANSFER TIMING

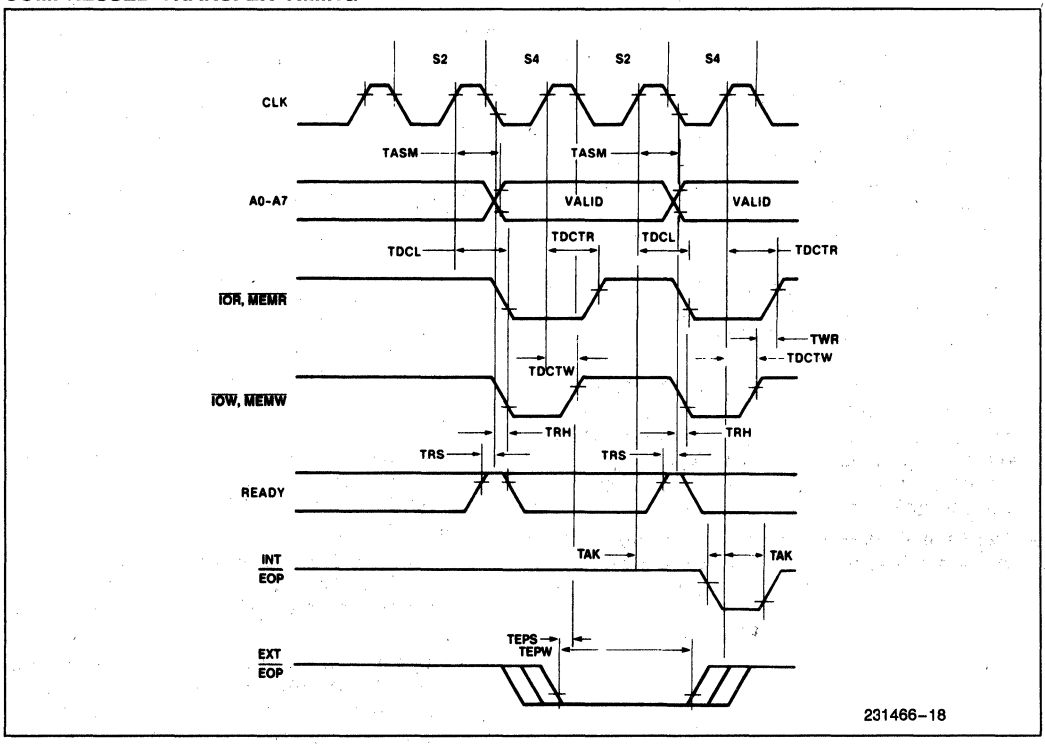


Figure 14. Compressed Transfer

RESET TIMING

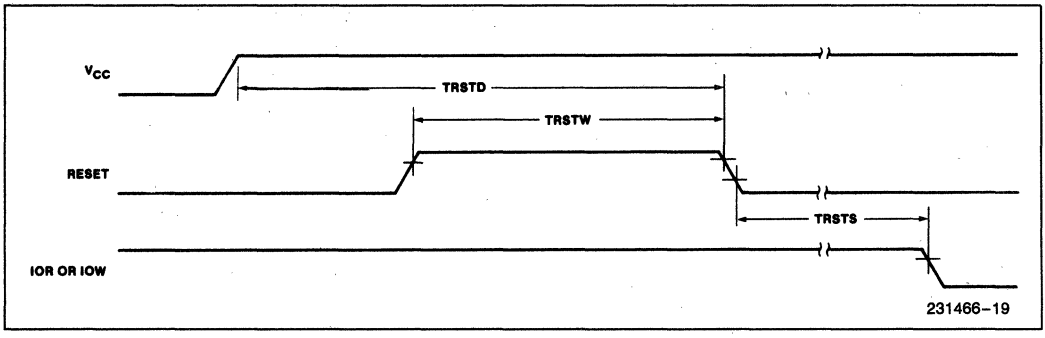


Figure 15. Reset

DESIGN CONSIDERATIONS

1. ***Cascading from channel zero.*** When using multiple 8237s, always start cascading with channel zero. Channel zero of the 8237 will operate incorrectly if one or more of channels 1, 2, or 3 are used in the cascade mode while channel zero is used in a mode other than cascade.
2. ***Do not treat the DREQ signal as an asynchronous input while the channel is in the "demand" or "cascade" modes.*** If DREQ becomes inactive at any time during state S4, an illegal state may occur causing the 8237 to operate improperly.
3. ***HRQ must remain active until HLDA becomes active.*** If HRQ goes inactive before HLDA is received the 8237 can enter an illegal state causing it to operate improperly.
4. ***Make sure the MEMR# line has 50 pF loading capacitance on it.*** When doing memory to memory transfers, the 8237 requires at least 50 pF loading capacitance on the MEMR# signal for proper operation. In most cases board capacitance is sufficient.
5. ***Treat the READY input as a synchronous input.*** If a transition occurs during the setup/hold window, erratic operation may result.

DATA SHEET REVISION REVIEW

The following list represents key differences between this and the -002 data sheet. Please review this summary carefully.

1. Major cleanup on the "NOTE" sections of this data sheet.
   a. Pin 5 no longer references a note. It is now included in the pin description area under the name "PIN5".
   b. The note placed in the "typical" section of the D.C. Characteristics table is now referenced to a note section included with that table.
   c. Notes in the A.C. Characteristics table have been renumbered and are included in a notes section for the A.C. Characteristics.
   d. The note that was previously referenced in the A.C. TESTING INPUT/OUTPUT WAVEFORM diagram has been replaced with the actual note.
   e. The note that was previously referenced in the SLAVE MODE WRITE TIMING diagram has been included in a "NOTE" section with the diagram.
   f. The note that was previously referenced in the SLAVE MODE READ TIMING diagram has been included in a "NOTE" section with the diagram.
   g. The note that was previously referenced in the DMA TRANSFER TIMING diagram has been included in a "NOTE" section with the diagram.
2. A "Design Considerations" section was added to alert designers to certain design aspects of the 8237.
3. The timing parameters TAR for the 8237A-4 and 8237A-5 have been changed from 50 ns to 0 ns.

82C37A-5
CHMOS HIGH PERFORMANCE PROGRAMMABLE DMA CONTROLLER

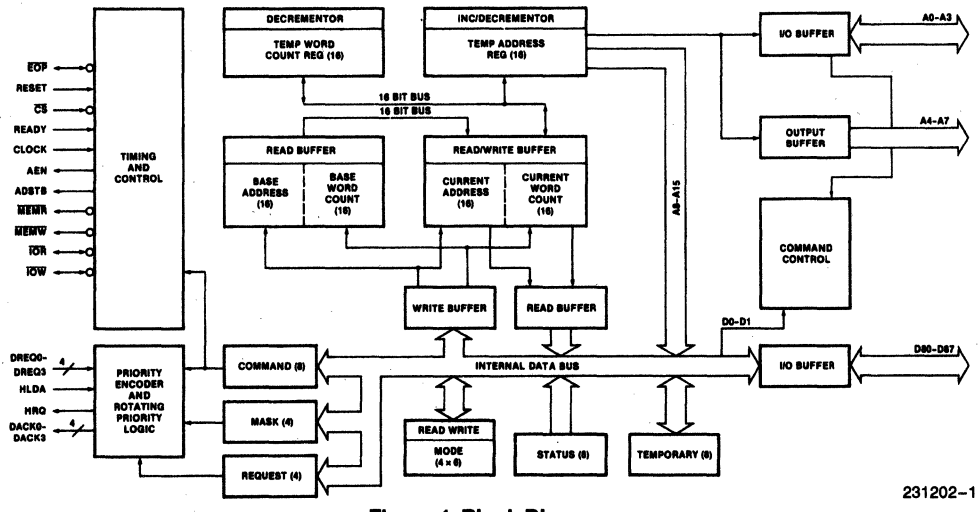
- Pin Compatible with NMOS 8237A-5
- Enable/Disable Control of Individual DMA Requests
- Fully Static Design with Frequency Range from DC to 5 MHz
- Low Power Operation
- Four Independent DMA Channels
- Independent Autoinitialization of all Channels
- Memory-to-Memory Transfers
- Memory Block Initialization
- Address Increment or Decrement
- High performance: 5 MHz Speed Transfers up to 1.6 MBytes/Second
- Directly Expandable to any Number of Channels
- End of Process Input for Terminating Transfers
- Software DMA Requests
- Independent Polarity Control for DREQ and DACK Signals
- Available in 40-Lead Plastic DIP

The Intel 82C37A-5 Multimode Direct Memory Access (DMA) Controller is a CHMOS peripheral interface circuit for microprocessor systems. It is designed to improve system performance by allowing external devices to directly transfer information from the system memory. Memory-to-memory transfer capability is also provided. The 82C37A-5 offers a wide variety of programmable control features to enhance data throughput and system optimization and to allow dynamic reconfiguration under program control.

The 82C37A-5 is designed to be used in conjunction with an external 8-bit address register. It contains four independent channels and may be expanded to any number of channels by cascading additional controller chips.

The three basic transfer modes allow programmability of the types of DMA service by the user. Each channel can be individually programmed to Autoinitialize to its original condition following an End of Process ($\overline{EOP}$).

Each channel has a full 64K address and word count capability.



231202-1

Figure 1. Block Diagram

September 1988
Order Number: 231202-005

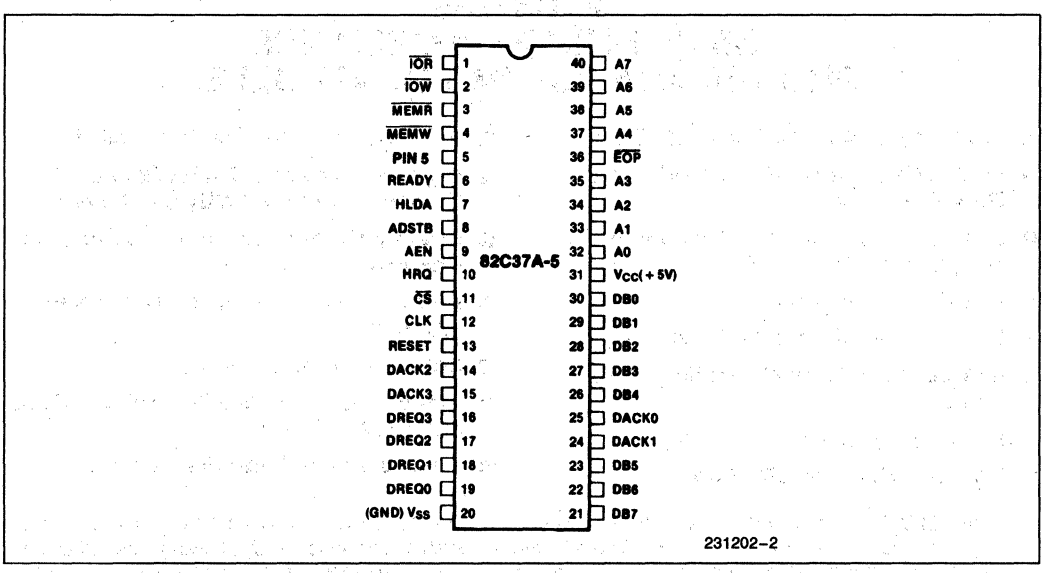


Figure 2. 82C37A-5
40-Lead DIP Configuration

Table 1. Pin Description

Symbol	Type	Name and Function
V_{CC}		**POWER:** +5 volt supply.
V_{SS}		**GROUND:** Ground.
CLK	I	**CLOCK INPUT:** Clock Input controls the internal operations of the 82C37A-5 and its rate of data transfers. The input may be driven at up to 5 MHz for the 82C37A-5.
$\overline{CS}$	I	**CHIP SELECT:** Chip Select is an active low input used to select the 82C37A-5 as an I/O device during the Idle cycle. This allows CPU communication on the data bus.
RESET	I	**RESET:** Reset is an active high input which clears the Command, Status, Request and Temporary registers. It also clears the first/last flip-flop and sets the Mask register. Following a Reset the device is in the Idle cycle.
READY	I	**READY:** Ready is an input used to extend the memory read and write pulses from the 82C37A-5 to accommodate slow memories or I/O peripheral devices. Ready must not make transitions during its specified setup/hold time.
HLDA	I	**HOLD ACKNOWLEDGE:** The active high Hold Acknowledge from the CPU indicates that it has relinquished control of the system busses.
DREQ0–DREQ3	I	**DMA REQUEST:** The DMA Request lines are individual asynchronous channel request inputs used by peripheral circuits to obtain DMA service. In fixed Priority, DREQ0 has the highest priority and DREQ3 has the lowest priority. A request is generated by activating the DREQ line of a channel. DACK will acknowledge the recognition of DREQ signal. Polarity of DREQ is programmable. Reset initializes these lines to active high. DREQ must be maintained until the corresponding DACK goes active.
DB0–DB7	I/O	**DATA BUS:** The Data Bus lines are bidirectional three-state signals connected to the system data bus. The outputs are enabled in the Program condition during the I/O Read to output the contents of an Address register, a Status register, the Temporary register or a Word Count register to the CPU. The outputs are disabled and the inputs are read during an I/O Write cycle when the CPU is programming the 82C37A-5 control registers. During DMA cycles the most significant 8 bits of the address are output onto the data bus to be strobed into an external latch by ADSTB. In memory-to-memory operations, data from the memory comes into the 82C37A-5 on the data bus during the read-from-memory transfer. In the write-to-memory transfer, the data bus outputs place the data into the new memory location.
$\overline{IOR}$	I/O	**I/O READ:** I/O Read is a bidirectional active low three-state line. In the Idle cycle, it is an input control signal used by the CPU to read the control registers. In the Active cycle, it is an output control signal used by the 82C37A-5 to access data from a peripheral during a DMA Write transfer.
$\overline{IOW}$	I/O	**I/O WRITE:** I/O Write is a bidirectional active low three-state line. In the Idle cycle, it is an input control signal used by the CPU to load information into the 82C37A-5. In the Active cycle, it is an output control signal used by the 82C37A-5 to load data to the peripheral during a DMA Read transfer.

Table 1. Pin Description (Continued)

Symbol	Type	Name and Function
$\overline{EOP}$	I/O	**END OF PROCESS:** End of Process is an active low bidirectional signal. Information concerning the completion of DMA services is available at the bidirectional $\overline{EOP}$ pin. The 82C37A-5 allows an external signal to terminate an active DMA service. This is accomplished by pulling the $\overline{EOP}$ input low with an external $\overline{EOP}$ signal. The 82C37A-5 also generates a pulse when the terminal count (TC) for any channel is reached. This generates an $\overline{EOP}$ signal which is output through the $\overline{EOP}$ Line. The reception of $\overline{EOP}$, either internal or external, will cause the 82C37A-5 to terminate the service, reset the request, and, if Autoinitialize is enabled, to write the base registers to the current registers of that channel. The mask bit and TC bit in the status word will be set for the currently active channel by $\overline{EOP}$ unless the channel is programmed for Autoinitialize. In that case, the mask bit remains unchanged. During memory-to-memory transfers, $\overline{EOP}$ will be output when the TC for channel 1 occurs. $\overline{EOP}$ should be tied high with a pull-up resistor if it is not used to prevent erroneous end of process inputs.
A0–A3	I/O	**ADDRESS:** The four least significant address lines are bidirectional three-state signals. In the Idle cycle they are inputs and are used by the CPU to address the register to be loaded or read. In the Active cycle they are outputs and provide the lower 4 bits of the output address.
A4–A7	O	**ADDRESS:** The four most significant address lines are three-state outputs and provide 4 bits of address. These lines are enabled only during the DMA service.
HRQ	O	**HOLD REQUEST:** This is the Hold Request to the CPU and is used to request control of the system bus. If the corresponding mask bit is clear, the presence of any valid DREQ causes 82C37A-5 to issue the HRQ. After HRQ goes active at least one clock cycle (TCY) must occur before HLDA goes active.
DACK0–DACK3	O	**DMA ACKNOWLEDGE:** DMA Acknowledge is used to notify the individual peripherals when one has been granted a DMA cycle. The sense of these lines is programmable. Reset initializes them to active low.
AEN	O	**ADDRESS ENABLE:** Address Enable enables the 8-bit latch containing the upper 8 address bits onto the system address bus. AEN can also be used to disable other system bus drivers during DMA transfers. AEN is active HIGH.
ADSTB	O	**ADDRESS STROBE:** The active high, Address Strobe is used to strobe the upper address byte into an external latch.
$\overline{MEMR}$	O	**MEMORY READ:** The Memory Read signal is an active low three-state output used to access data from the selected memory location during a DMA Read or a memory-to-memory transfer.
$\overline{MEMW}$	O	**MEMORY WRITE:** The Memory Write is an active low three-state output used to write data to the selected memory location during a DMA Write or a memory-to-memory transfer.
PIN5	I	**PIN5:** This pin should always be at a logic HIGH level. An internal pull-up resistor will establish a logic HIGH when the pin is left floating. It is recommended, however, that PIN5 be connected to V_{CC}.

FUNCTIONAL DESCRIPTION

The 82C37A-5 block diagram includes the major logic blocks and all of the internal registers. The data interconnection paths are also shown. Not shown are the various control signals between the blocks. The 82C37A-5 contains 344 bits of internal memory in the form of registers. Figure 3 lists these registers by name and shows the size of each. A detailed description of the registers and their functions can be found under Register Description.

Name	Size	Number
Base Address Registers	16 bits	4
Base Word Count Registers	16 bits	4
Current Address Registers	16 bits	4
Current Word Count Registers	16 bits	4
Temporary Address Register	16 bits	1
Temporary Word Count Register	16 bits	1
Status Register	8 bits	1
Command Register	8 bits	1
Temporary Register	8 bits	1
Mode Registers	6 bits	4
Mask Register	4 bits	1
Request Register	4 bits	1

Figure 3. 82C37A-5 Internal Registers

The 82C37A-5 contains three basic blocks of control logic. The Timing Control block generates internal timing and external control signals for the 82C37A-5. The Program Command Control block decodes the various commands given to the 82C37A-5 by the microprocessor prior to servicing a DMA Request. It also decodes the Mode Control word used to select the type of DMA during the servicing. The Priority Encoder block resolves priority contention between DMA channels requesting service simultaneously.

DMA Operation

The 82C37A-5 is designed to operate in two major cycles. These are called Idle and Active cycles. Each device cycle is made up of a number of states. The 82C37A-5 can assume seven separate states, each composed of one full clock period. State 1 (S1) is the inactive state. It is entered when the 82C37A-5 has no valid DMA requests pending. While in S1, the DMA controller is inactive but may be in the Program Condition, being programmed by the processor. State 0 (S0) is the first state of a DMA service. The 82C37A-5 has requested a hold but the processor has not yet returned an acknowledge. The 82C37A-5 may still be programmed until it receives HLDA from the CPU. An acknowledge from the CPU will signal that DMA transfers may begin. S1, S2, S3 and S4 are the working states of the DMA service. If more time is needed to complete a transfer than is available with normal timing, wait states (SW) can be inserted between S2 or S3 and S4 by the use of the Ready line on the 82C37A-5. Note that the data is transferred directly from the I/O device to memory (or vice versa) with $\overline{IOR}$ and $\overline{MEMW}$ (or $\overline{MEMR}$ and $\overline{IOW}$) being active at the same time. The data is not read into or driven out of the 82C37A-5 in I/O-to-memory or memory-to-I/O DMA transfers.

Memory-to-memory transfers require a read-from and a write-to-memory to complete each transfer. The states, which resemble the normal working states, use two digit numbers for identification. Eight states are required for a single transfer. The first four states (S11, S12, S13, S14) are used for the read-from-memory half and the last four states (S21, S22, S23, S24) for the write-to-memory half of the transfer.

IDLE CYCLE

When no channel is requesting service, the 82C37A-5 will enter the Idle cycle and perform "S1" states. In this cycle the 82C37A-5 will sample the DREQ lines every clock cycle to determine if any channel is requesting a DMA service. The device will also sample $\overline{CS}$, looking for an attempt by the microprocessor to write or read the internal registers of the 82C37A-5. When $\overline{CS}$ is low and HLDA is low, the 82C37A-5 enters the Program Condition. The CPU can now establish, change or inspect the internal definition of the part by reading from or writing to the internal registers. Address lines A0–A3 are inputs to the device and select which registers will be read or written. The $\overline{IOR}$ and $\overline{IOW}$ lines are used to select and time reads or writes. Due to the number and size of the internal registers, an internal flip-flop is used to generate an additional bit of address. This bit is used to determine the upper or lower byte of the 16-bit Address and Word Count registers. The flip-flop is reset by Master Clear or Reset. A separate software command can also reset this flip-flop.

Special software commands can be executed by the 82C37A-5 in the Program Condition. These commands are decoded as sets of addresses with the $\overline{CS}$ and $\overline{IOW}$. The commands do not make use of the data bus. Instructions include Clear First/Last Flip-Flop and Master Clear.

ACTIVE CYCLE

When the 82C37A-5 is in the Idle cycle and a non-masked channel requests a DMA service, the device

will output an HRQ to the microprocessor and enter the Active cycle. It is in this cycle that the DMA service will take place, in one of four modes:

Single Transfer Mode — In Single Transfer mode the device is programmed to make one transfer only. The word count will be decremented and the address decremented or incremented following each transfer. When the word count "rolls over" from zero to FFFFH, a Terminal Count (TC) will cause an Autoinitialize if the channel has been programmed to do so.

DREQ must be held active until DACK becomes active in order to be recognized. If DREQ is held active throughout the single transfer, HRQ will go inactive and release the bus to the system. It will again go active and, upon receipt of a new HLDA, another single transfer will be performed, in 8080A, 8085AH, 80C88, or 80C86 system this will ensure one full machine cycle execution between DMA transfers. Details of timing between the 82C37A-5 and other bus control protocols will depend upon the characteristics of the microprocessor involved.

Block Transfer Mode — In Block Transfer mode the device is activated by DREQ to continue making transfers during the service until a TC, caused by word count going to FFFFH, or an external End of Process (EOP) is encountered. DREQ need only be held active until DACK becomes active. Again, an Autoinitialization will occur at the end of the service if the channel has been programmed for it.

Demand Transfer Mode — In Demand Transfer mode the device is programmed to continue making transfers until a TC or external $\overline{EOP}$ is encountered or until DREQ goes inactive. Thus transfers may continue until the I/O device has exhausted its data capacity. After the I/O device has had a chance to catch up, the DMA service is re-established by means of a DREQ. During the time between services when the microprocessor is allowed to operate, the intermediate values of address and word count are stored in the 82C37A-5 Current Address and Current Word Count registers. Only an $\overline{EOP}$ can cause an Autoinitialize at the end of the service. $\overline{EOP}$ is generated either by TC or by an external signal.

Cascade Mode — This mode is used to cascade more than one 82C37A-5 together for simple system expansion. The HRQ and HLDA signals from the additional 82C37A-5 are connected to the DREQ and DACK signals of a channel of the initial 82C37A-5. This allows the DMA requests of the additional device to propagate through the priority network circuitry of the preceding device. The priority chain is preserved and the new device must wait for its turn to acknowledge requests. Since the cascade channel of the initial 82C37A-5 is used only for prioritizing the additional device, it does not output any address or control signals of its own. These could conflict with the outputs of the active channel in the added device. The 82C37A-5 will respond to DREQ and DACK but all other outputs except HRQ will be disabled. The ready input is ignored.

Figure 4 shows two additional devices cascaded into an initial device using two of the previous channels. This forms a two level DMA system. More 82C37A-5s could be added at the second level by using the remaining channels of the first level. Additional devices can also be added by cascading into the channels of the second level devices, forming a third level.

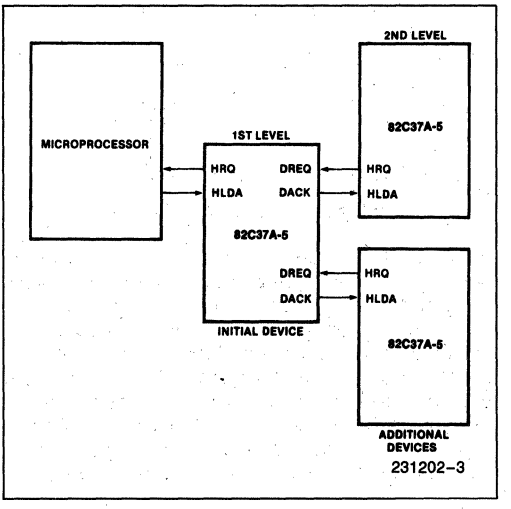


Figure 4. Cascaded 82C37A-5s

TRANSFER TYPES

Each of the three active transfer modes can perform three different types of transfers. These are Read, Write and Verify. Write transfers move data from and I/O device to the memory by activating $\overline{MEMW}$ and $\overline{IOR}$. Read transfers move data from memory to an I/O device by activating $\overline{MEMR}$ and $\overline{IOW}$. Verify transfers are pseudo transfers. The 82C37A-5 operates as in Read or Write transfers generating addresses, and responding to EOP, etc. However, the memory and I/O control lines all remain inactive. The ready input is ignored in verify mode.

Memory-to-Memory — To perform block moves of data from one memory address space to another with a minimum of program effort and time, the 82C37A-5 includes a memory-to-memory transfer feature. Programming a bit in the Command register selects channels 0 to 1 to operate as memory-to-memory transfer channels. The transfer is initiated by setting the software DREQ for channel 0. The

82C37A-5 requests a DMA service in the normal manner. After HLDA is true, the device, using four state transfers in Block Transfer mode, reads data from the memory. The channel 0 Current Address register is the source for the address used and is decremented or incremented in the normal manner. The data byte read from the memory is stored in the 82C37A-5 internal Temporary register. Channel 1 then performs a four-state transfer of the data from the Temporary register to memory using the address in its Current Address register and incrementing or decrementing it in the normal manner. The channel 1 current Word Count is decremented. When the word count of channel 1 goes to FFFFH, a TC is generated causing an $\overline{EOP}$ output terminating the service.

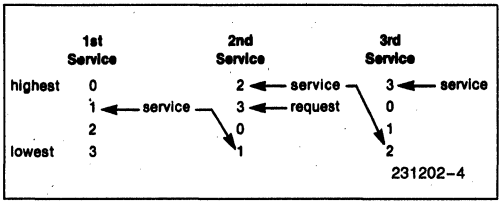
Channel 0 may be programmed to retain the same address for all transfers. This allows a single word to be written to a block of memory.

The 82C37A-5 will respond to external $\overline{EOP}$ signals during memory-to-memory transfers. Data comparators in block search schemes may use this input to terminate the service when a match is found. The timing of memory-to-memory transfers is found in Figure 12. Memory-to-memory operations can be detected as an active AEN with no DACK outputs.

Autoinitialize — By programming a bit in the Mode register, a channel may be set up as an Autoinitialize channel. During Autoinitialize initialization, the original values of the Current Address and Current Word Count registers are automatically restored from the Base Address and Base Word count registers of that channel following $\overline{EOP}$. The base registers are loaded simultaneously with the current registers by the microprocessor and remain unchanged throughout the DMA service. The mask bit is not altered when the channel is in Autoinitialize. Following Autoinitialize the channel is ready to perform another DMA service, without CPU intervention, as soon as a valid DREQ is detected. In order to Autoinitialize both channels in a memory-to-memory transfer, both word counts should be programmed identically. If interrupted externally, $\overline{EOP}$ pulses should be applied in both bus cycles.

Priority — The 82C37A-5 has two types of priority encoding available as software selectable options. The first is Fixed Priority which fixes the channels in priority order based upon the descending value of their number. The channel with the lowest priority is 3 followed by 2, 1 and the highest priority channel, 0. After the recognition of any one channel for service, the other channels are prevented from interfering with that service until it is completed.

The second scheme is Rotating Priority. The last channel to get service becomes the lowest priority channel with the others rotating accordingly.



With Rotating Priority in a single chip DMA system, any device requesting service is guaranteed to be recognized after no more than three higher priority services have occurred. This prevents any one channel from monopolizing the system.

Compressed Timing — In order to achieve even greater throughput where system characteristics permit, the 82C37A-5 can compress the transfer time to two clock cycles. From Figure 11 it can be seen that state S3 is used to extend the access time of the read pulse. By removing state S3, the read pulse width is made equal to the write pulse width and a transfer consists only of state S2 to change the address and state S4 to perform the read/write. S1 states will still occur when A8–A15 need updating (see Address Generation). Timing for compressed transfers is found in Figure 14.

Address Generation — In order to reduce pin count, the 82C37A-5 multiplexes the eight higher order address bits on the data lines. State S1 is used to output the higher order address bits to an external latch from which they may be placed on the address bus. The falling edge of Address Strobe (ADSTB) is used to load these bits from the data lines to the latch. Address Enable (AEN) is used to enable the bits onto the address bus through a three-state enable. The lower order address bits are output by the 82C37A-5 directly. Lines A0–A7 should be connected to the address bus. Figure 11 shows the time relationships between CLK, AEN, ADSTB, DB0–DB7 and A0–A7.

During Block and Demand Transfer mode services, which include multiple transfers, the addresses generated will be sequential. For many transfers the data held in the external address latch will remain the same. This data need only change when a carry or borrow from A7 to A8 takes place in the normal sequence of addresses. To save time and speed transfers, the 82C37A-5 executes S1 states only when updating of A8–A15 in the latch is necessary. This means for long services, S1 states and Address Strobes may occur only once every 256 transfers, a savings of 255 clock cycles for each 256 transfers.

REGISTER DESCRIPTION

Current Address Register — Each channel has a 16-bit Current Address register. This register holds

the value of the address used during DMA transfers. The address is automatically incremented or decremented after each transfer and the intermediate values of the address are stored in the Current Address register during the transfer. This register is written or read by the microprocessor in successive 8-bit bytes. It may also be reinitialized by an Autoinitialize back to its original value. Autoinitialize takes place only after an $\overline{EOP}$.

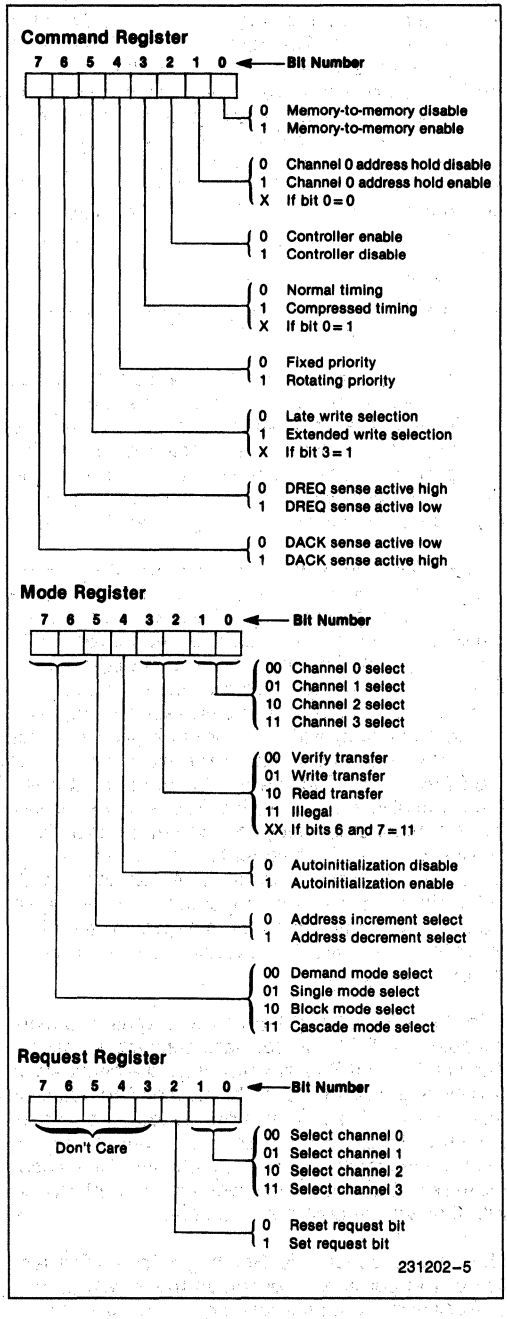
Current Word Register — Each channel has a 16-bit Current Word Count register. This register determines the number of transfers to be performed. The actual number of transfers will be one more than the number programmed in the Current Word Count register (i.e., programming a count of 100 will result in 101 transfers). The word count is decremented after each transfer. The intermediate value of the word count is stored in the register during the transfer. When the value in the register goes from zero to FFFFH, a TC will be generated. This register is loaded or read in successive 8-bit bytes by the microprocessor in the Program Condition. Following the end of a DMA service it may also be reinitialized by an Autoinitialization back to its original value. Autoinitialize can occur only when an $\overline{EOP}$ occurs. If it is not Autoinitialized, this register will have a count of FFFFH after TC.

Base Address and Base Word Count Registers — Each channel has a pair of Base Address and Base Word Count registers. These 16-bit registers store the original value of their associated current registers. During Autoinitialize these values are used to restore the current registers to their original values. The base registers are written simultaneously with their corresponding current register in 8-bit bytes in the Program Condition by the microprocessor. These registers cannot be read by the microprocessor.

Command Register — This 8-bit register controls the operation of the 82C37A-5. It is programmed by the microprocessor in the Program Condition and is cleared by Reset or a Master Clear instruction. The following table lists the function of the command bits. See Figure 6 for address coding.

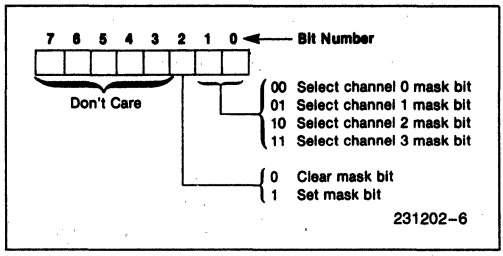
Mode Register — Each channel has a 6-bit Mode register associated with it. When the register is being written to by the microprocessor in the Program Condition, bits 0 and 1 determine which channel Mode register is to be written.

Request Register — The 82C37A-5 can respond to requests for DMA service which are initiated by software as well as by a DREQ. Each channel has a request bit associated with it in the 4-bit Request register. These are non-maskable and subject to prioritization by the Priority Encoder network. Each register bit is set or reset separately under software control or is cleared upon generation of a TC or external $\overline{EOP}$. The entire register is cleared by a Reset. To set or reset a bit, the software loads the proper form of the data word. See Figure 5 for register ad-

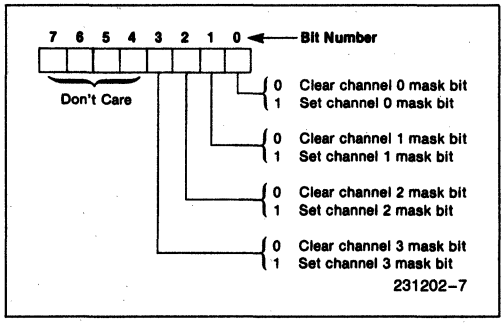


dress coding. In order to make a software request, the channel must be in Block Mode.

Mask Register — Each channel has associated with it a mask bit which can be set to disable the incoming DREQ. Each mask bit is set when its associated channel produces an $\overline{EOP}$ if the channel is not programmed for Autoinitialize. Each bit of the 4-bit Mask register may also be set or cleared separately under software control. The entire register is also set by a Reset. This disables all DMA requests until a clear Mask register instruction allows them to occur. The instruction to separately set or clear the mask bits is similar in form to that used with the Request register. See Figure 5 for instruction addressing.



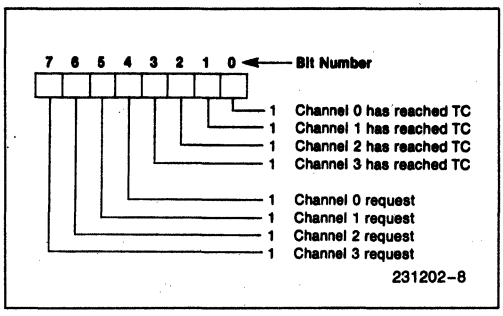
All four bits of the Mask register may also be written with a single command.



Register	Operation	Signals						
		$\overline{CS}$	$\overline{IOR}$	$\overline{IOW}$	A3	A2	A1	A0
Command	Write	0	1	0	1	0	0	0
Mode	Write	0	1	0	1	0	1	1
Request	Write	0	1	0	1	0	0	1
Mask	Set/Reset	0	1	0	1	0	1	0
Mask	Write	0	1	0	1	1	1	1
Temporary	Read	0	0	1	1	1	0	1
Status	Read	0	0	1	1	0	0	0

Figure 5. Definition of Register Codes

Status Register — The Status register is available to be read out of the 82C37A-5 by the microprocessor. It contains information about the status of the devices at this point. This information includes which channels have reached a terminal count and which channels have pending DMA requests. Bits 0–3 are set every time a TC is reached by that channel or an external $\overline{EOP}$ is applied. These bits are cleared upon Reset and on each Status Read. Bits 4–7 are set whenever their corresponding channel is requesting service.



Temporary Register — The Temporary register is used to hold data during memory-to-memory transfers. Following the completion of the transfers, the last word moved can be read by the microprocessor in the Program Condition. The Temporary register always contains the last byte transferred in the previous memory-to-memory operation, unless cleared by a Reset.

Software Commands — These are additional special software commands which can be executed in the Program Condition. They do not depend on any specific bit pattern on the data bus. The three software commands are:

Clear First/Last Flip-Flop: This command is executed prior to writing or reading new address or word count information to the 82C37A-5. This initializes the flip-flop to a known state so that subsequent accesses to register contents by the microprocessor will address upper and lower bytes in the correct sequence.

Master Clear: This software instruction has the same effect as the hardware Reset. The Command, Status, Request, Temporary, and Internal First/Last Flip-Flop registers are cleared and the Mask register is set. The 82C37A-5 will enter the Idle cycle.

Clear Mask Register: This command clears the mask bits of all four channels, enabling them to accept DMA requests.

Figure 6 lists the address codes for the software commands:

Signals						Operation
A3	A2	A1	A0	$\overline{IOR}$	$\overline{IOW}$	
1	0	0	0	0	1	Read Status Register
1	0	0	0	1	0	Write Command Register
1	0	0	1	0	1	Illegal
1	0	0	1	1	0	Write Request Register
1	0	1	0	0	1	Illegal
1	0	1	0	1	0	Write Single Mask Register Bit
1	0	1	1	0	1	Illegal
1	0	1	1	1	0	Write Mode Register
1	1	0	0	0	1	Illegal
1	1	0	0	1	0	Clear Byte Pointer Flip-Flop
1	1	0	1	0	1	Read Temporary Register
1	1	0	1	1	0	Master Clear
1	1	1	0	0	1	Illegal
1	1	1	0	1	0	Clear Mask Register
1	1	1	1	0	1	Illegal
1	1	1	1	1	0	Write All Mask Register Bits

Figure 6. Software Command Codes

PROGRAMMING

The 82C37A-5 will accept programming from the host processor any time that HLDA is inactive; this is true even if HRQ is active. The responsibility of the host is to assure that programming and HLDA are mutually exclusive. Note that a problem can occur if a DMA request occurs, on an unmasked channel while the 82C37A-5 is being programmed. For instance, the CPU may be starting to reprogram the two byte Address register of channel 1 when channel 1 receives a DMA request. If the 82C37A-5 is enabled (bit 2 in the command register is 0) and channel 1 is unmasked, a DMA service will occur after only one byte of the Address register has been reprogrammed. This can be avoided by disabling the controller (setting bit 2 in the command register) or masking the channel before programming any other registers. Once the programming is complete, the controller can be enabled/unmasked.

Channel	Register	Operation	Signals							Internal Flip-Flop	Data Bus DB0–DB7
			$\overline{CS}$	$\overline{IOR}$	$\overline{IOW}$	A3	A2	A1	A0		
0	Base and Current Address	Write	0	1	0	0	0	0	0	0	A0–A7
			0	1	0	0	0	0	0	1	A8–A15
	Current Address	Read	0	0	1	0	0	0	0	0	A0–A7
			0	0	1	0	0	0	0	1	A8–A15
	Base and Current Word Count	Write	0	1	0	0	0	0	1	0	W0–W7
			0	1	0	0	0	0	1	1	W8–W15
	Current Word Count	Read	0	0	1	0	0	0	1	0	W0–W7
			0	0	1	0	0	0	1	1	W8–W15
1	Base and Current Address	Write	0	1	0	0	0	1	0	0	A0–A7
			0	1	0	0	0	1	0	1	A8–A15
	Current Address	Read	0	0	1	0	0	1	0	0	A0–A7
			0	0	1	0	0	1	0	1	A8–A15
	Base and Current Word Count	Write	0	1	0	0	0	1	1	0	W0–W7
			0	1	0	0	0	1	1	1	W8–W15
	Current Word Count	Read	0	0	1	0	0	1	1	0	W0–W7
			0	0	1	0	0	1	1	1	W8–W15
2	Base and Current Address	Write	0	1	0	0	1	0	0	0	A0–A7
			0	1	0	0	1	0	0	1	A8–A15
	Current Address	Read	0	0	1	0	1	0	0	0	A0–A7
			0	0	1	0	1	0	0	1	A8–A15
	Base and Current Word Count	Write	0	1	0	0	1	0	1	0	W0–W7
			0	1	0	0	1	0	1	1	W8–W15
	Current Word Count	Read	0	0	1	0	1	0	1	0	W0–W7
			0	0	1	0	1	0	1	1	W8–W15
3	Base and Current Address	Write	0	1	0	0	1	1	0	0	A0–A7
			0	1	0	0	1	1	0	1	A8–A15
	Current Address	Read	0	0	1	0	1	1	0	0	A0–A7
			0	0	1	0	1	1	0	1	A8–A15
	Base and Current Word Count	Write	0	1	0	0	1	1	1	0	W0–W7
			0	1	0	0	1	1	1	1	W8–W15
	Current Word Count	Read	0	0	1	0	1	1	1	0	W0–W7
			0	0	1	0	1	1	1	1	W8–W15

Figure 7. Word Count and Address Register Command Codes

After power-up it is suggested that all internal locations, especially the Mode registers, be loaded with some valid value. This should be done even if some channels are unused.

APPLICATION INFORMATION

Figure 8 shows a convenient method for configuring a DMA system with the 82C37A-5 controller and an 8080A/8085AH microprocessor system. The multimode DMA controller issues a HRQ to the processor whenever there is at least one valid DMA request from a peripheral device. When the processor replies with a HLDA signal, the 82C37A-5 takes control of the address bus, the data bus and the control bus. The address for the first transfer operation comes out in two bytes — the least significant 8 bits on the eight address outputs and the most significant 8 bits on the data bus. The contents of the data bus are then latched into the 8-bit latch to complete the full 16 bits of the address bus. After the initial transfer takes place, the latch is updated only after a carry or borrow is generated in the least significant address byte. Four DMA channels are provided when one 82C37A-5 is used.

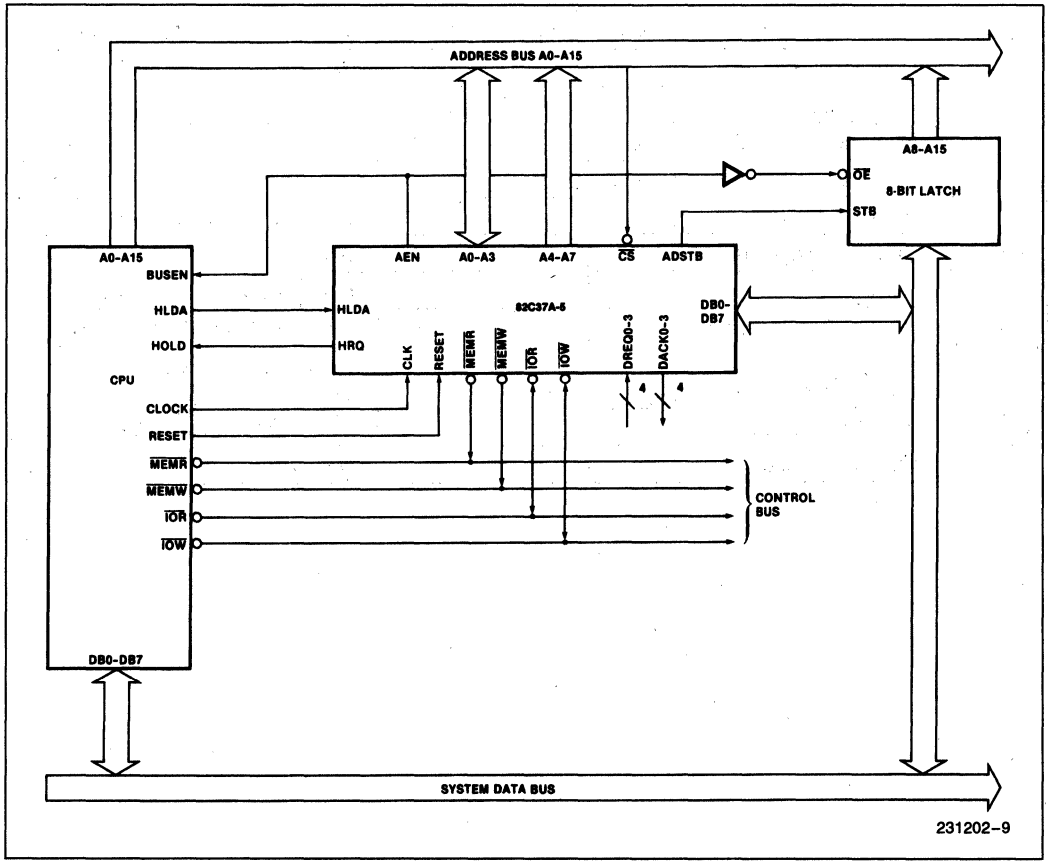


Figure 8. 82C37A-5 System Interface

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature under Bias .....0°C to 70°C
Case Temperature .....0°C to +75°C
Storage Temperature .....−55°C to +150°C
Voltage on Any Pin with Respect to Ground.....−0.5V to +7V
Power Dissipation.....1.0 Watt

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, T_{CASE} = 0°C to 75°C, V_{CC} = +5.0V ±5%, GND = 0V

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
V_{OH}	Output High Voltage	3.7			V	I_{OH} = −1.0 mA
V_{OL}	Output LOW Voltage			0.40	V	I_{OL} = 3.2 mA
V_{IH}	Input HIGH Voltage	2.2		V_{CC} + 0.5	V	
V_{IL}	Input LOW Voltage	−0.5		0.8	V	
I_{LI}	Input Load Current			±10	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{LO}	Output Leakage Current			±10	μA	0V ≤ V_{OUT} ≤ V_{CC}
I_{CC}	V_{CC} Supply Current			10	mA	(Note 1)
I_{CCS}	Standby Supply Current			10	μA	HLDA = 0V, V_{IL} = 0V, V_{IH} = V_{CC}
C_O	Output Capacitance		4	8	pF	fc = 1.0 MHz, Inputs = 0V
C_I	Input Capacitance		8	15	pF	
C_{IO}	I/O Capacitance		10	18	pF	

A.C. CHARACTERISTICS—DMA (MASTER) MODE

T_A = 0°C to 70°C, T_{CASE} = 0°C to 75°C, V_{CC} = +5V ±5%, GND = 0V

Symbol	Parameter	Min	Max	Unit
TAEL	AEN HIGH from CLK LOW (S1) Delay Time		200	ns
TAET	AEN LOW from CLK HIGH (SI) Delay Time		130	ns
TAFAB	ADR Active to Float Delay from CLK HIGH		90	ns
TAFC	$\overline{READ}$ or $\overline{WRITE}$ Float from CLK HIGH		120	ns
TAFDB	DB Active to Float Delay from CLK HIGH		170	ns
TAHR	ADR from $\overline{READ}$ HIGH Hold Time	TCY-100		ns
TAHS	DB from ADSTB LOW Hold Time	30		ns
TAHW	ADR from $\overline{WRITE}$ HIGH Hold Time	TCY-50		ns
TAK	DACK Valid from CLK LOW Delay Time (Note 3)		170	ns
	$\overline{EOP}$ HIGH from CLK HIGH Delay Time (Note 4)		170	ns
	$\overline{EOP}$ LOW from CLK HIGH Delay Time		170	ns
TASM	ADR Stable from CLK HIGH		170	ns
TASS	DB to ADSTB LOW Setup Time	100		ns
TCH	Clock High Time (Transitions ≤ 10 ns)	68		ns
TCL	Clock LOW Time (Transitions ≤ 10 ns)	68		ns
TCY	CLK Cycle Time	200		ns
TDCL	CLK HIGH to $\overline{READ}$ or $\overline{WRITE}$ LOW Delay (Note 2)		190	ns
TDCTR	$\overline{READ}$ HIGH from CLK HIGH (S4) Delay Time (Note 2)		190	ns
TDCTW	$\overline{WRITE}$ HIGH from CLK HIGH (S4) Delay Time (Note 2)		130	ns
TDQ1	HRQ Valid from CLK HIGH Delay Time		120	ns
TEPS	$\overline{EOP}$ LOW from CLK LOW Setup Time	40		ns
TEPW	$\overline{EOP}$ Pulse Width	220		ns
TFAAB	ADR Float to Active Delay from CLK HIGH		170	ns
TFAC	$\overline{READ}$ or $\overline{WRITE}$ Active from CLK HIGH		150	ns
TFADB	DB Float to Active Delay from CLK HIGH		200	ns
THS	HLDA Valid to CLK HIGH Setup Time	75		ns
TIDH	Input Data from $\overline{MEMR}$ HIGH Hold Time	0		ns
TIDS	Input Data to $\overline{MEMR}$ HIGH Setup Time	170		ns
TODH	Output Data from $\overline{MEMW}$ HIGH Hold Time	10		ns
TODV	Output Data Valid to $\overline{MEMW}$ HIGH	125		ns
TQS	DREQ to CLK LOW (SI, S4) Setup Time (Note 3)	0		ns
TRH	CLK to READY LOW Hold Time	20		ns
TRS	READY to CLK LOW Setup Time	60		ns
TSTL	ADSTB HIGH from CLK HIGH Delay Time		130	ns
TSTT	ADSTB LOW from CLK HIGH Delay Time		90	ns

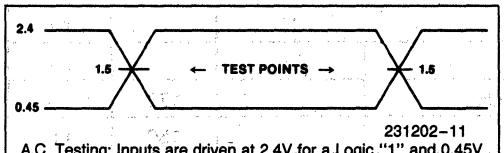
A.C. CHARACTERISTICS—PERIPHERAL (SLAVE) MODE

T_A = 0°C to 70°C, T_{CASE} = 0°C to 75°C, V_{CC} = +5V ±5%, GND = 0V

Symbol	Parameter	Min	Max	Unit
TAR	ADR Valid or $\overline{CS}$ LOW to $\overline{READ}$ LOW	50		ns
TAW	ADR Valid to $\overline{WRITE}$ HIGH Setup Time	130		ns
TCW	CS LOW to $\overline{WRITE}$ HIGH Setup Time	130		ns
TDW	Data Valid to $\overline{WRITE}$ HIGH Setup Time	130		ns
TRA	ADR or CS Hold from $\overline{READ}$ HIGH	0		ns
TRDE	Data Access from $\overline{READ}$ LOW		140	ns
TRDF	DB Float Delay from $\overline{READ}$ HIGH	0	70	ns
TRSTD	Power Supply HIGH to RESET LOW Setup Time	500		ns
TRSTS	RESET to First $\overline{IOWR}$	2TCY		ns
TRSTW	RESET Pulse Width	300		ns
TRW	$\overline{READ}$ Width	200		ns
TWA	ADR from $\overline{WRITE}$ HIGH Hold Time	20		ns
TWC	CS HIGH from $\overline{WRITE}$ HIGH Hold Time	20		ns
TWD	Data from $\overline{WRITE}$ HIGH Hold Time	30		ns
TWWS	Write Width	160		ns

NOTES:
1. Input frequency 5 MHz, when RESET, V_{IN} = 0V/V_{CC}, C_L = 0 pF.
2. The net $\overline{IOW}$ or $\overline{MEMW}$ Pulse width for normal write will be TCY-100 ns and for extended write will be 2TCY-100 ns. The net $\overline{IOR}$ or $\overline{MEMR}$ pulse width for normal read will be 2TCY-50 ns and for compressed read will be TCY-50 ns.
3. DREQ and DACK signals may be active high or active low. Timing diagrams assume the active high mode for DREQ and active low for DACK.
4. $\overline{EOP}$ is an open collector output. This parameter assumes the presence of a 2.2K pullup to V_{CC}.

A.C. TESTING INPUT/OUTPUT WAVEFORM



231202-11

A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0." All timing measurements are made at 1.5V.

WAVEFORMS

SLAVE MODE WRITE TIMING

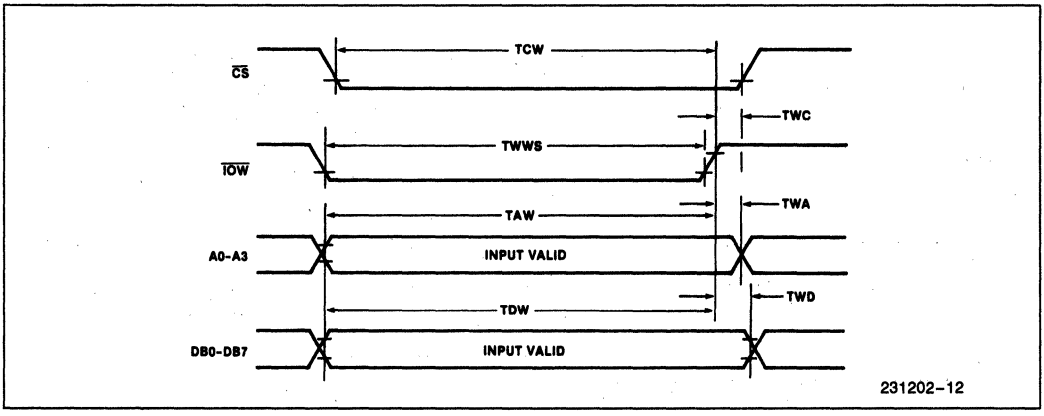


Figure 9. Slave Mode Write

SLAVE MODE READ TIMING

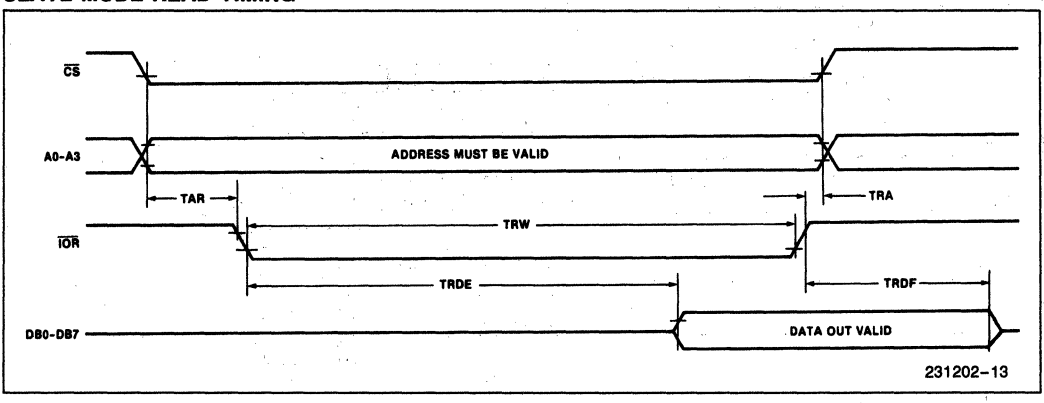


Figure 10. Slave Mode Read

WAVEFORMS (Continued)

DMA TRANSFER TIMING

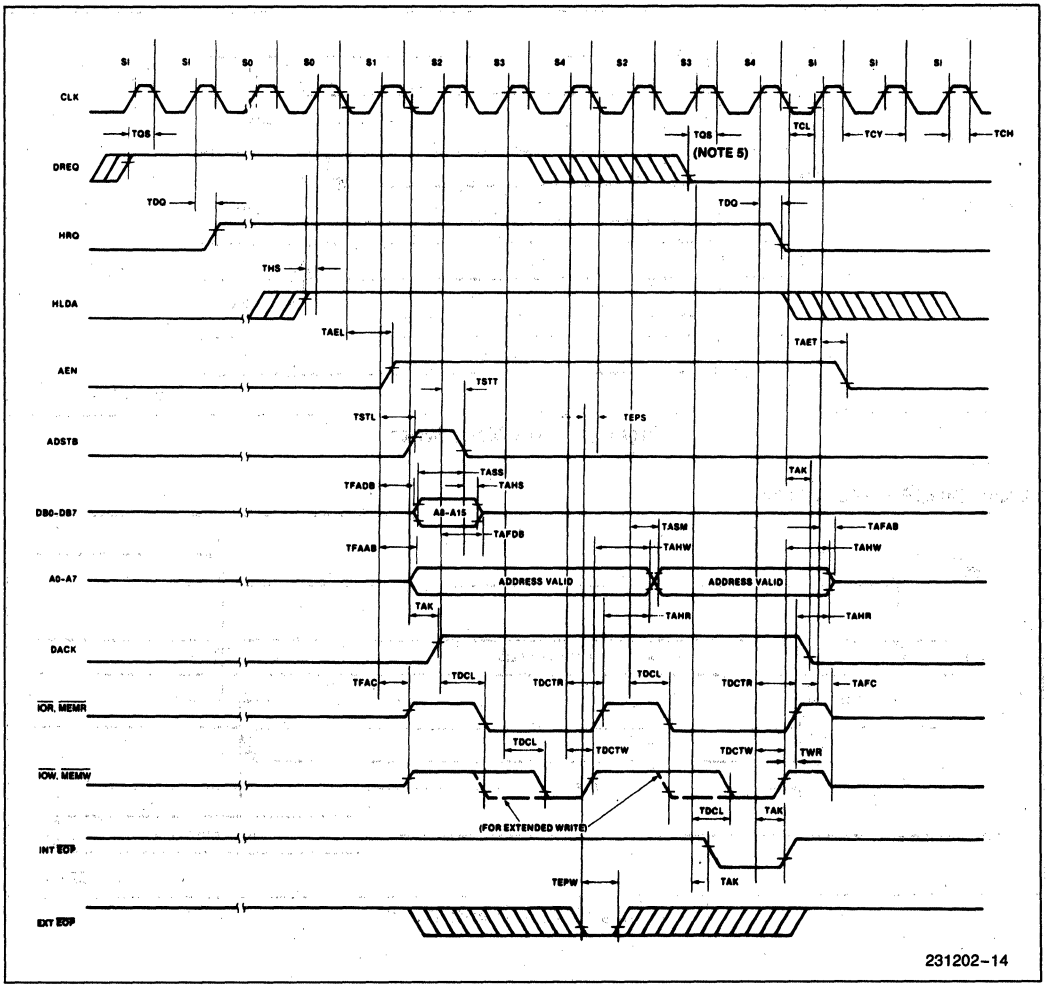


Figure 11. DMA Transfer

WAVEFORMS (Continued)

MEMORY-TO-MEMORY TRANSFER TIMING

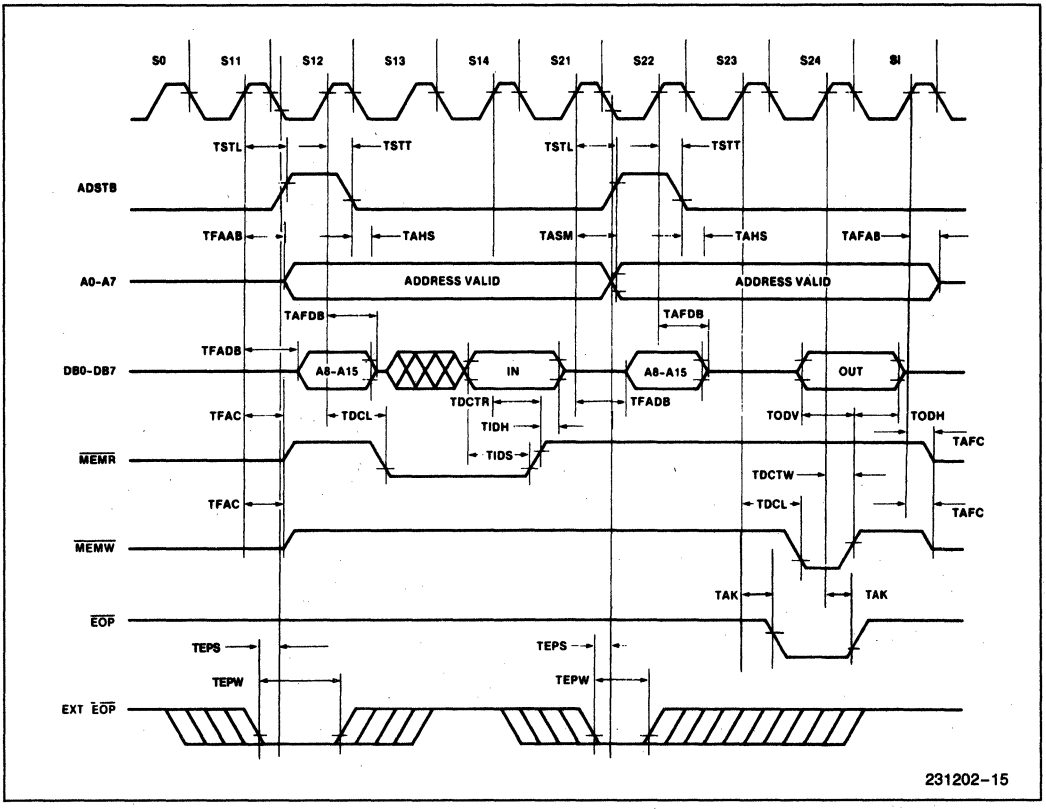


Figure 12. Memory-to-Memory Transfer

READY TIMING

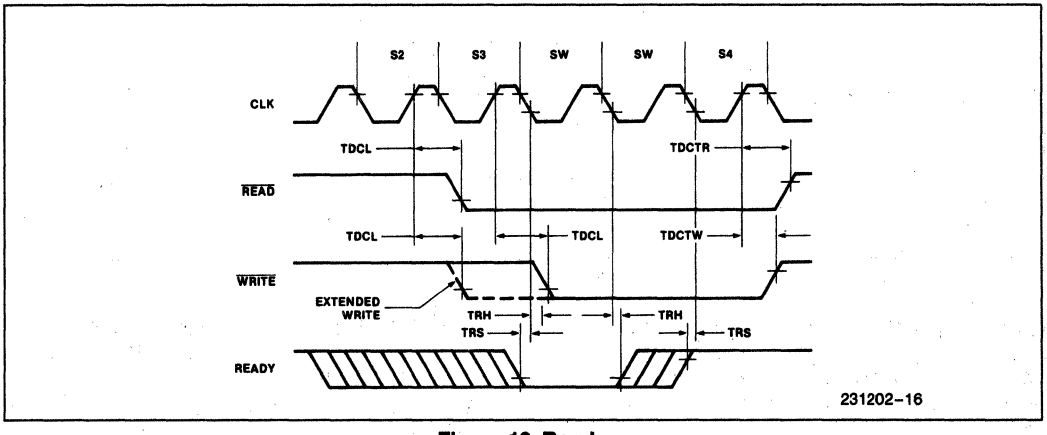


Figure 13. Ready

WAVEFORMS (Continued)

COMPRESSED TRANSFER TIMING

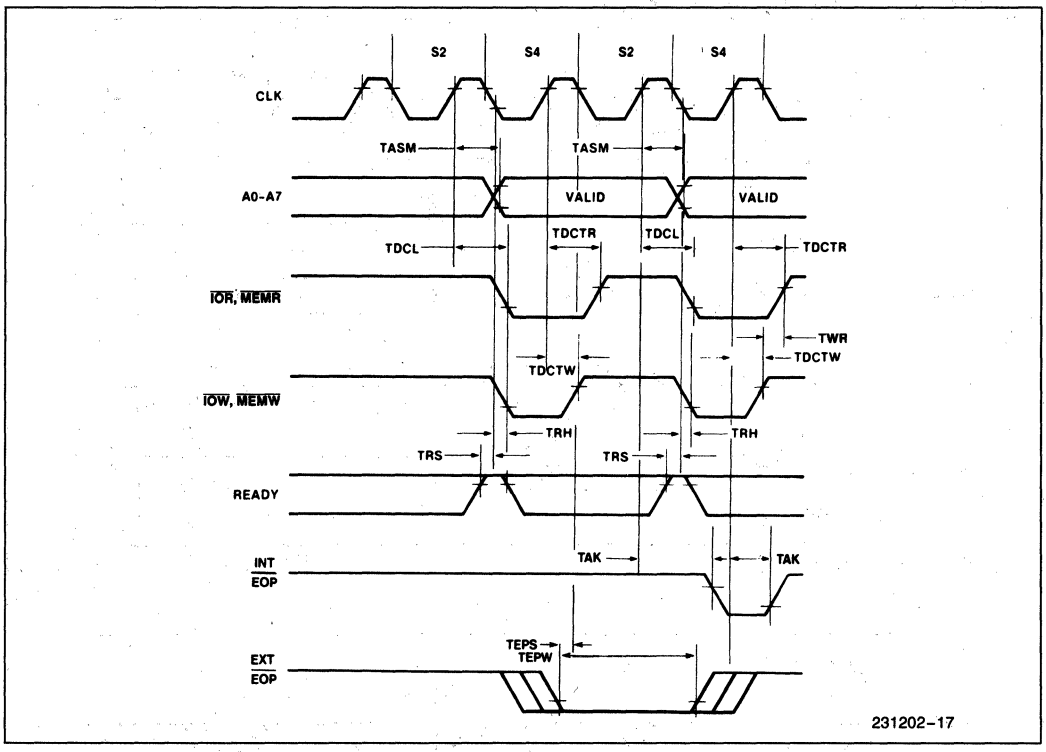


Figure 14. Compressed Transfer

RESET TIMING

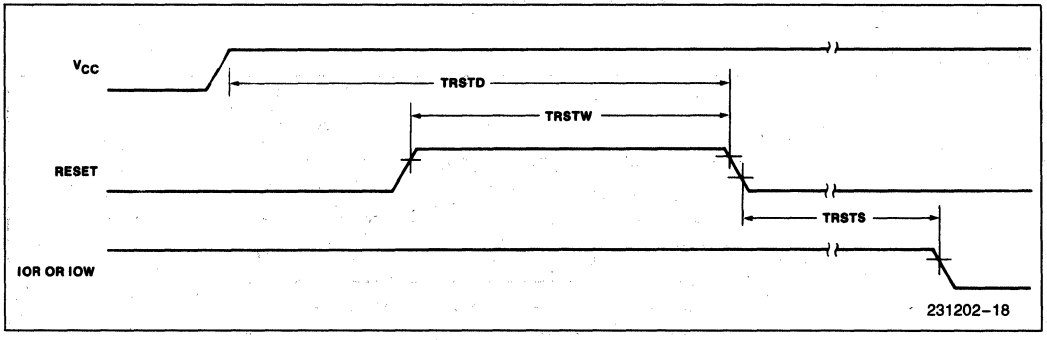


Figure 15. Reset

DATA SHEET REVISION REVIEW

The following list represents key differences between this and the -004 data sheet. Please review this summary carefully.

1. The "PRELIMINARY" markings have been removed from the data sheet. The 82C37A-5 is no longer a preliminary part.
2. A section of the Functional Description describing 82C37A-5 operation with the 8085 CPU has been deleted.

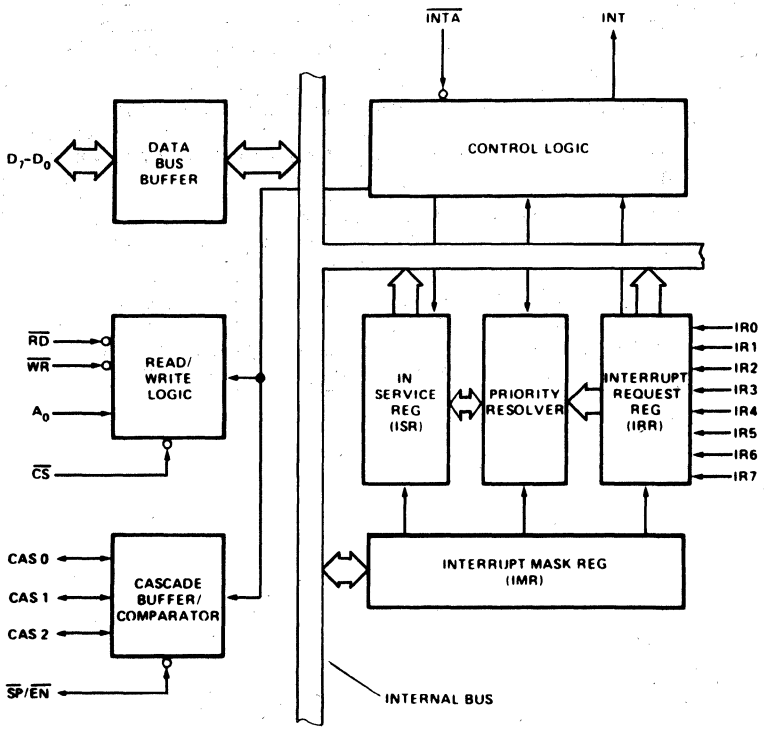
8259A
PROGRAMMABLE INTERRUPT CONTROLLER
(8259A/8259A-2)

- **8086, 8088 Compatible**
- **MCS-80®, MCS-85® Compatible**
- **Eight-Level Priority Controller**
- **Expandable to 64 Levels**
- **Programmable Interrupt Modes**
- **Individual Request Mask Capability**
- **Single +5V Supply (No Clocks)**
- **Available in 28-Pin DIP and 28-Lead PLCC Package**
  (See Packaging Spec., Order #231369)
- **Available in EXPRESS**
  — Standard Temperature Range
  — Extended Temperature Range

The Intel 8259A Programmable Interrupt Controller handles up to eight vectored priority interrupts for the CPU. It is cascadable for up to 64 vectored priority interrupts without additional circuitry. It is packaged in a 28-pin DIP, uses NMOS technology and requires a single +5V supply. Circuitry is static, requiring no clock input.

The 8259A is designed to minimize the software and real time overhead in handling multi-level priority interrupts. It has several modes, permitting optimization for a variety of system requirements.

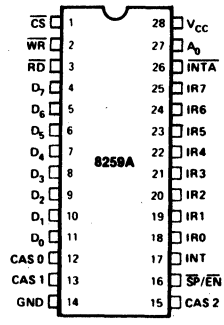
The 8259A is fully upward compatible with the Intel 8259. Software originally written for the 8259 will operate the 8259A in all 8259 equivalent modes (MCS-80/85, Non-Buffered, Edge Triggered).



231468-1

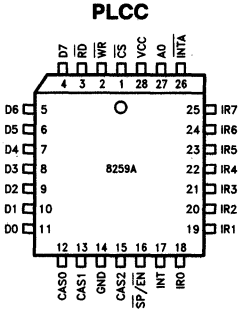
Figure 1. Block Diagram

DIP



231468-2

PLCC



231468-31

Figure 2. Pin Configurations

October 1988
Order Number: 231468-003

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
V_{CC}	28	I	**SUPPLY:** +5V Supply.
GND	14	I	**GROUND**
$\overline{CS}$	1	I	**CHIP SELECT:** A low on this pin enables $\overline{RD}$ and $\overline{WR}$ communication between the CPU and the 8259A. INTA functions are independent of CS.
$\overline{WR}$	2	I	**WRITE:** A low on this pin when CS is low enables the 8259A to accept command words from the CPU.
$\overline{RD}$	3	I	**READ:** A low on this pin when CS is low enables the 8259A to release status onto the data bus for the CPU.
D_7–D_0	4–11	I/O	**BIDIRECTIONAL DATA BUS:** Control, status and interrupt-vector information is transferred via this bus.
CAS_0–CAS_2	12, 13, 15	I/O	**CASCADE LINES:** The CAS lines form a private 8259A bus to control a multiple 8259A structure. These pins are outputs for a master 8259A and inputs for a slave 8259A.
$\overline{SP}/\overline{EN}$	16	I/O	**SLAVE PROGRAM/ENABLE BUFFER:** This is a dual function pin. When in the Buffered Mode it can be used as an output to control buffer transceivers (EN). When not in the buffered mode it is used as an input to designate a master (SP = 1) or slave (SP = 0).
INT	17	O	**INTERRUPT:** This pin goes high whenever a valid interrupt request is asserted. It is used to interrupt the CPU, thus it is connected to the CPU's interrupt pin.
IR_0–IR_7	18–25	I	**INTERRUPT REQUESTS:** Asynchronous inputs. An interrupt request is executed by raising an IR input (low to high), and holding it high until it is acknowledged (Edge Triggered Mode), or just by a high level on an IR input (Level Triggered Mode).
$\overline{INTA}$	26	I	**INTERRUPT ACKNOWLEDGE:** This pin is used to enable 8259A interrupt-vector data onto the data bus by a sequence of interrupt acknowledge pulses issued by the CPU.
A_0	27	I	**AO ADDRESS LINE:** This pin acts in conjunction with the $\overline{CS}$, $\overline{WR}$, and $\overline{RD}$ pins. It is used by the 8259A to decipher various Command Words the CPU writes and status the CPU wishes to read. It is typically connected to the CPU A0 address line (A1 for 8086, 8088).

FUNCTIONAL DESCRIPTION

Interrupts In Microcomputer Systems

Microcomputer system design requires that I.O devices such as keyboards, displays, sensors and other components receive servicing in a an efficient manner so that large amounts of the total system tasks can be assumed by the microcomputer with little or no effect on throughput.

The most common method of servicing such devices is the *Polled* approach. This is where the processor must test each device in sequence and in effect "ask" each one if it needs servicing. It is easy to see that a large portion of the main program is looping through this continuous polling cycle and that such a method would have a serious detrimental effect on system throughput, thus limiting the tasks that could be assumed by the microcomputer and reducing the cost effectiveness of using such devices.

A more desirable method would be one that would allow the microprocessor to be executing its main program and only stop to service peripheral devices when it is told to do so by the device itself. In effect, the method would provide an external asynchronous input that would inform the processor that it should complete whatever instruction that is currently being executed and fetch a new routine that will service the requesting device. Once this servicing is complete, however, the processor would resume exactly where it left off.

This method is called *Interrupt*. It is easy to see that system throughput would drastically increase, and thus more tasks could be assumed by the microcomputer to further enhance its cost effectiveness.

The Programmable Interrupt Controller (PIC) functions as an overall manager in an Interrupt-Driven system environment. It accepts requests from the peripheral equipment, determines which of the incoming requests is of the highest importance (priority), ascertains whether the incoming request has a higher priority value than the level currently being serviced, and issues an interrupt to the CPU based on this determination.

Each peripheral device or structure usually has a special program or "routine" that is associated with its specific functional or operational requirements; this is referred to as a "service routine". The PIC, after issuing an Interrupt to the CPU, must somehow input information into the CPU that can "point" the Program Counter to the service routine associated with the requesting device. This "pointer" is an address in a vectoring table and will often be referred to, in this document, as vectoring data.

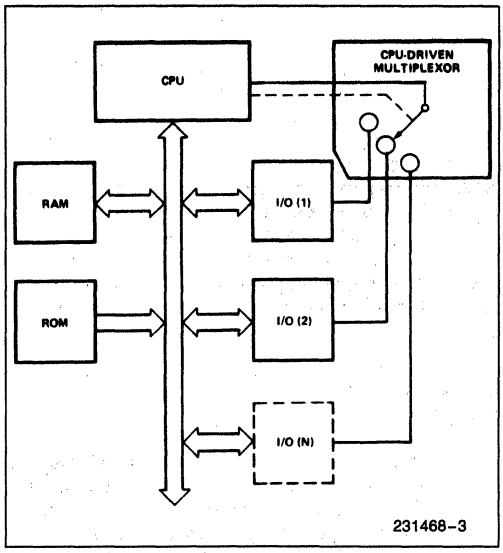


Figure 3a. Polled Method

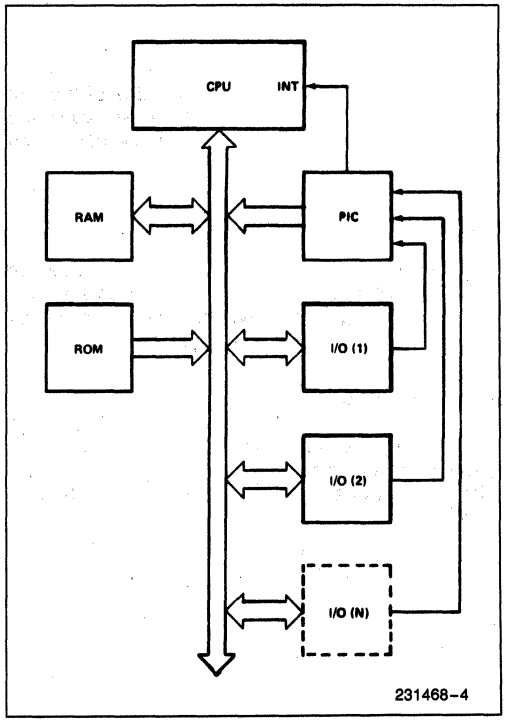


Figure 3b. Interrupt Method

The 8259A is a device specifically designed for use in real time, interrupt driven microcomputer systems. It manages eight levels or requests and has built-in features for expandability to other 8259A's (up to 64 levels). It is programmed by the system's software as an I/O peripheral. A selection of priority modes is available to the programmer so that the manner in which the requests are processed by the 8259A can be configured to match his system requirements. The priority modes can be changed or reconfigured dynamically at any time during the main program. This means that the complete interrupt structure can be defined as required, based on the total system environment.

INTERRUPT REQUEST REGISTER (IRR) AND IN-SERVICE REGISTER (ISR)

The interrupts at the IR input lines are handled by two registers in cascade, the Interrupt Request Register (IRR) and the In-Service (ISR). The IRR is used to store all the interrupt levels which are requesting service; and the ISR is used to store all the interrupt levels which are being serviced.

PRIORITY RESOLVER

This logic block determines the priorites of the bits set in the IRR. The highest priority is selected and strobed into the corresponding bit of the ISR during $\overline{INTA}$ pulse.

INTERRUPT MASK REGISTER (IMR)

The IMR stores the bits which mask the interrupt lines to be masked. The IMR operates on the IRR. Masking of a higher priority input will not affect the interrupt request lines of lower quality.

INT (INTERRUPT)

This output goes directly to the CPU interrupt input. The V_{OH} level on this line is designed to be fully compatible with the 8080A, 8085A and 8086 input levels.

$\overline{INTA}$ (INTERRUPT ACKNOWLEDGE)

$\overline{INTA}$ pulses will cause the 8259A to release vectoring information onto the data bus. The format of this data depends on the system mode (μPM) of the 8259A.

DATA BUS BUFFER

This 3-state, bidirectional 8-bit buffer is used to interface the 8259A to the system Data Bus. Control words and status information are transferred through the Data Bus Buffer.

READ/WRITE CONTROL LOGIC

The function of this block is to accept OUTput commands from the CPU. It contains the Initialization Command Word (ICW) registers and Operation Command Word (OCW) registers which store the various control formats for device operation. This function block also allows the status of the 8259A to be transferred onto the Data Bus.

$\overline{CS}$ (CHIP SELECT)

A LOW on this input enables the 8259A. No reading or writing of the chip will occur unless the device is selected.

$\overline{WR}$ (WRITE)

A LOW on this input enables the CPU to write control words (ICWs and OCWs) to the 8259A.

$\overline{RD}$ (READ)

A LOW on this input enables the 8259A to send the status of the Interrupt Request Register (IRR), In Service Register (ISR), the Interrupt Mask Register (IMR), or the Interrupt level onto the Data Bus.

A_0

This input signal is used in conjunction with $\overline{WR}$ and $\overline{RD}$ signals to write commands into the various command registers, as well as reading the various status registers of the chip. This line can be tied directly to one of the address lines.

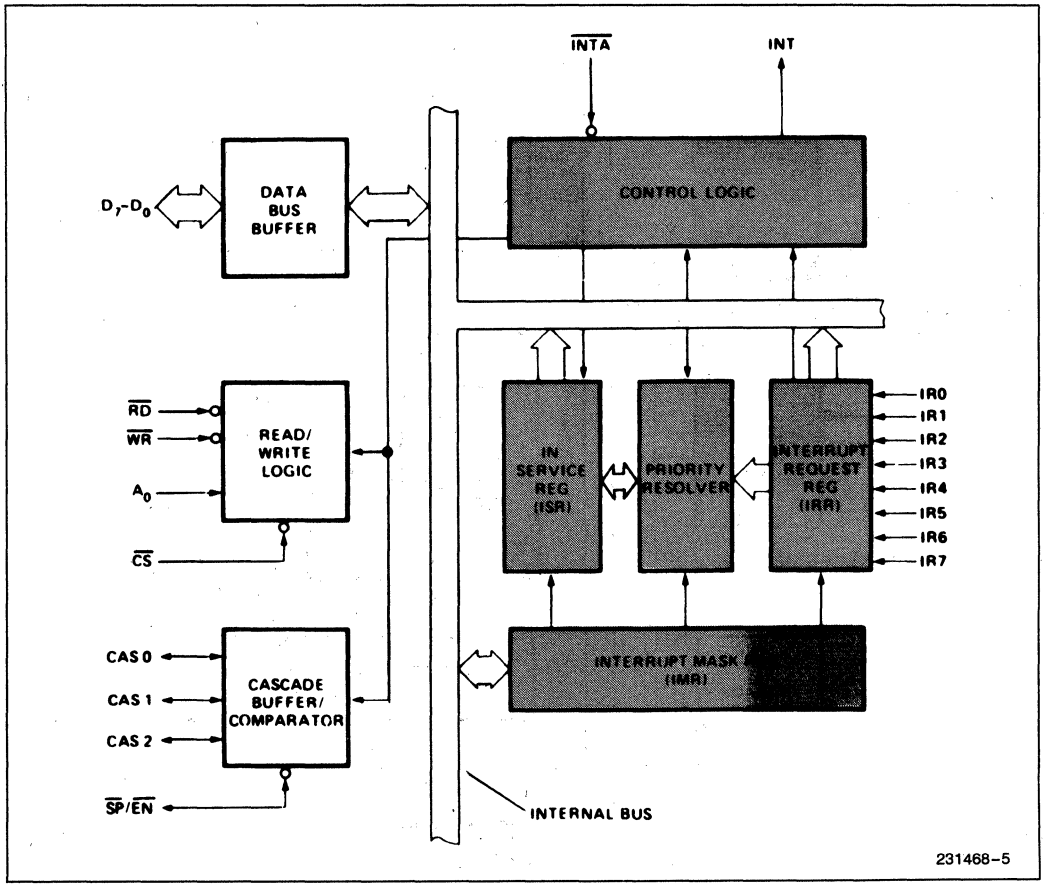


Figure 4a. 8259A Block Diagram

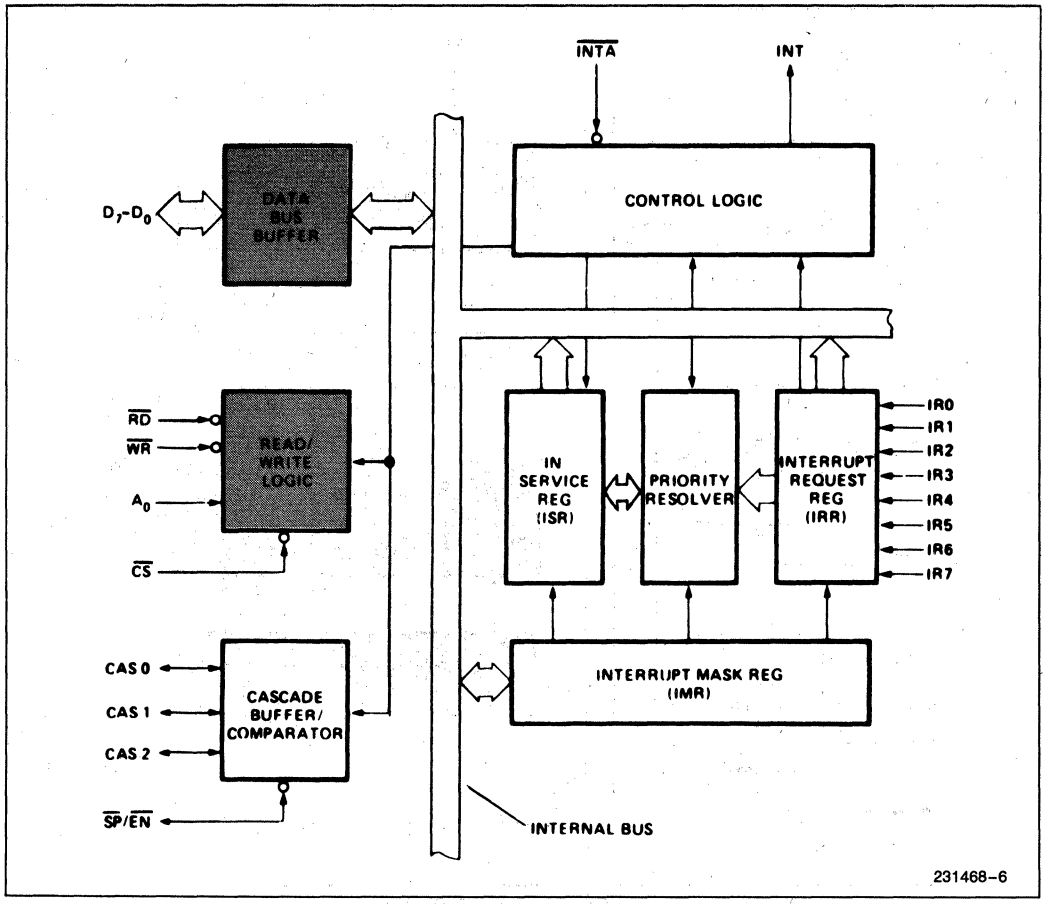


Figure 4b. 8259A Block Diagram

THE CASCADE BUFFER/COMPARATOR

This function block stores and compares the IDs of all 8259A's used in the system. The associated three I/O pins (CAS0-2) are outputs when the 8259A is used as a master and are inputs when the 8259A is used as a slave. As a master, the 8259A sends the ID of the interrupting slave device onto the CAS0–2 lines. The slave thus selected will send its preprogrammed subroutine address onto the Data Bus during the next one or two consecutive $\overline{INTA}$ pulses. (See section "Cascading the 8259A".)

INTERRUPT SEQUENCE

The powerful features of the 8259A in a microcomputer system are its programmability and the interrupt routine addressing capability. The latter allows direct or indirect jumping to the specific interrupt routine requested without any polling of the interrupting devices. The normal sequence of events during an interrupt depends on the type of CPU being used.

The events occur as follows in an MCS-80/85 system:

1. One or more of the INTERRUPT REQUEST lines (IR7–0) are raised high, setting the corresponding IRR bit(s).
2. The 8259A evaluates these requests, and sends an INT to the CPU, if appropriate.
3. The CPU acknowledges the INT and responds with an $\overline{INTA}$ pulse.
4. Upon receiving an $\overline{INTA}$ from the CPU group, the highest priority ISR bit is set, and the corresponding IRR bit is reset. The 8259A will also release a CALL instruction code (11001101) onto the 8-bit Data Bus through its D7–0 pins.
5. This CALL instruction will initiate two more $\overline{INTA}$ pulses to be sent to the 8259A from the CPU group.
6. These two $\overline{INTA}$ pulses allow the 8259A to release its preprogrammed subroutine address onto the Data Bus. The lower 8-bit address is released at the first $\overline{INTA}$ pulse and the higher 8-bit address is released at the second $\overline{INTA}$ pulse.
7. This completes the 3-byte CALL instruction released by the 8259A. In the AEOI mode the ISR bit is reset at the end of the third $\overline{INTA}$ pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt sequence.

The events occuring in an 8086 system are the same until step 4.

4. Upon receiving an $\overline{INTA}$ from the CPU group, the highest priority ISR bit is set and the corresponding IRR bit is reset. The 8259A does not drive the Data Bus during this cycle.
5. The 8086 will initiate a second $\overline{INTA}$ pulse. During this pulse, the 8259A releases an 8-bit pointer onto the Data Bus where it is read by the CPU.
6. This completes the interrupt cycle. In the AEOI mode the ISR bit is reset at the end of the second $\overline{INTA}$ pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt subroutine.

If no interrupt request is present at step 4 of either sequence (i.e., the request was too short in duration) the 8259A will issue an interrupt level 7. Both the vectoring bytes and the CAS lines will look like an interrupt level 7 was requested.

When the 8259A PIC receives an interrupt, INT becomes active and an interrupt acknowledge cycle is started. If a higher priority interrupt occurs between the two INTA pulses, the INT line goes inactive immediately after the second INTA pulse. After an unspecified amount of time the INT line is activated again to signify the higher priority interrupt waiting for service. This inactive time is not specified and can vary between parts. The designer should be aware of this consideration when designing a system which uses the 8259A. It is recommended that proper asynchronous design techniques be followed.

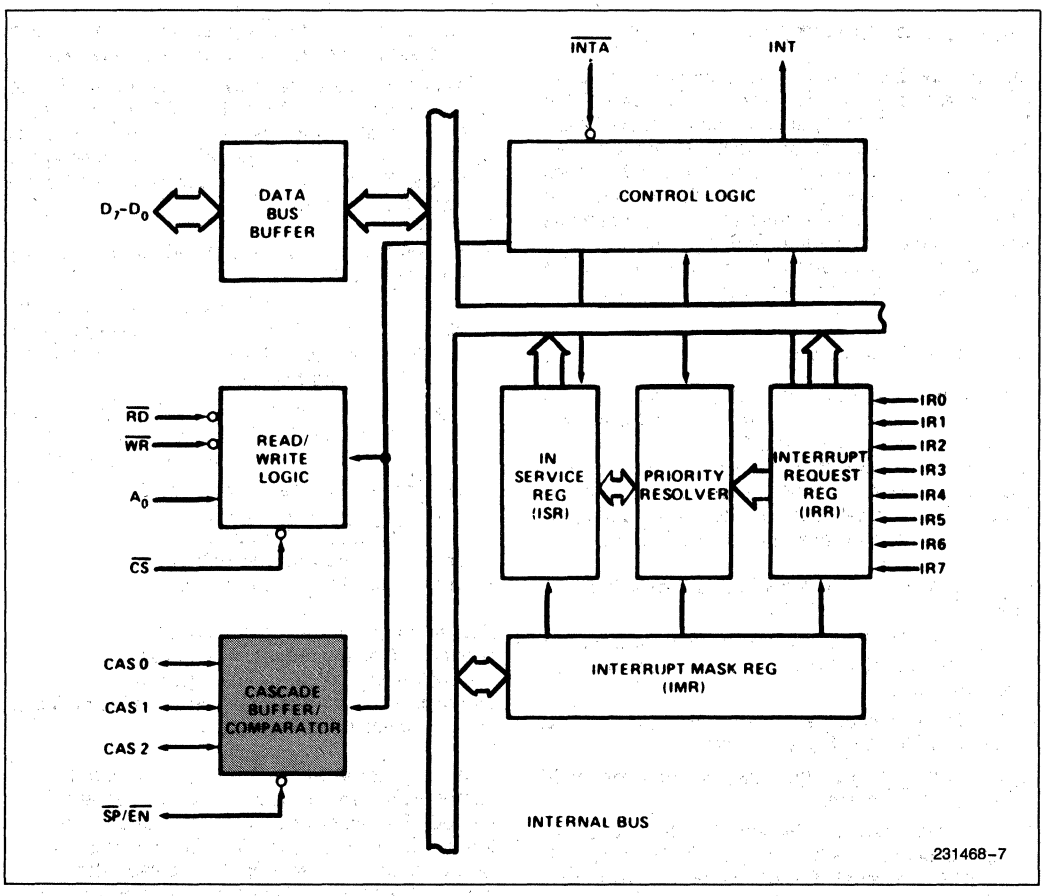


Figure 4c. 8259A Block Diagram

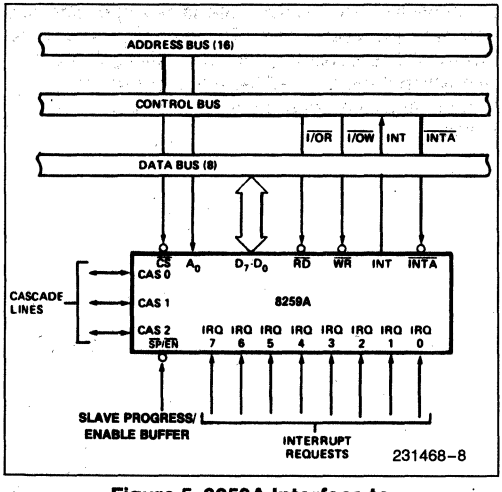


Figure 5. 8259A Interface to Standard System Bus

INTERRUPT SEQUENCE OUTPUTS

MCS-80®, MCS-85®

This sequence is timed by three $\overline{INTA}$ pulses. During the first $\overline{INTA}$ pulse the CALL opcode is enabled onto the data bus.

Content of First Interrupt Vector Byte

	D7	D6	D5	D4	D3	D2	D1	D0
CALL CODE	1	1	0	0	1	1	0	1

During the second $\overline{INTA}$ pulse the lower address of the appropriate service routine is enabled onto the data bus. When Interval = 4 bits A_5–A_7 are programmed, while A_0–A_4 are automatically inserted by the 8259A. When Interval = 8 only A_6 and A_7 are programmed, while A_0–A_5 are automatically inserted.

Content of Second Interrupt Vector Byte

IR	Interval = 4							
	D7	D6	D5	D4	D3	D2	D1	D0
7	A7	A6	A5	1	1	1	0	0
6	A7	A6	A5	1	1	0	0	0
5	A7	A6	A5	1	0	1	0	0
4	A7	A6	A5	1	0	0	0	0
3	A7	A6	A5	0	1	1	0	0
2	A7	A6	A5	0	1	0	0	0
1	A7	A6	A5	0	0	1	0	0
0	A7	A6	A5	0	0	0	0	0

IR	Interval = 8							
	D7	D6	D5	D4	D3	D2	D1	D0
7	A7	A6	1	1	1	0	0	0
6	A7	A6	1	1	0	0	0	0
5	A7	A6	1	0	1	0	0	0
4	A7	A6	1	0	0	0	0	0
3	A7	A6	0	1	1	0	0	0
2	A7	A6	0	1	0	0	0	0
1	A7	A6	0	0	1	0	0	0
0	A7	A6	0	0	0	0	0	0

During the third $\overline{INTA}$ pulse the higher address of the appropriate service routine, which was programmed as byte 2 of the initialization sequence (A_8–A_{15}), is enabled onto the bus.

Content of Third Interrupt Vector Byte

D7	D6	D5	D4	D3	D2	D1	D0
A15	A14	A13	A12	A11	A10	A9	A8

8086, 8088

8086 mode is similar to MCS-80 mode except that only two Interrupt Acknowledge cycles are issued by the processor and no CALL opcode is sent to the processor. The first interrupt acknowledge cycle is similar to that of MCS-80, 85 systems in that the 8259A uses it to internally freeze the state of the interrupts for priority resolution and as a master it issues the interrupt code on the cascade lines at the end of the $\overline{INTA}$ pulse. On this first cycle it does not issue any data to the processor and leaves its data bus buffers disabled. On the second interrupt acknowledge cycle in 8086 mode the master (or slave if so programmed) will send a byte of data to the processor with the acknowledged interrupt code composed as follows (note the state of the ADI mode control is ignored and A_5–A_{11} are unused in 8086 mode):

Content of Interrupt Vector Byte for 8086 System Mode

	D7	D6	D5	D4	D3	D2	D1	D0
IR7	T7	T6	T5	T4	T3	1	1	1
IR6	T7	T6	T5	T4	T3	1	1	0
IR5	T7	T6	T5	T4	T3	1	0	1
IR4	T7	T6	T5	T4	T3	1	0	0
IR3	T7	T6	T5	T4	T3	0	1	1
IR2	T7	T6	T5	T4	T3	0	1	0
IR1	T7	T6	T5	T4	T3	0	0	1
IR0	T7	T6	T5	T4	T3	0	0	0

PROGRAMMING THE 8259A

The 8259A accepts two types of command words generated by the CPU:

1. *Initialization Command Words (ICWs):* Before normal operation can begin, each 8259A in the system must be brought to a starting point—by a sequence of 2 to 4 bytes timed by $\overline{WR}$ pulses.
2. *Operation Command Words (OCWs):* These are the command words which command the 8259A to operate in various interrupt modes. These modes are:
   a. Fully nested mode
   b. Rotating priority mode
   c. Special mask mode
   d. Polled mode

The OCWs can be written into the 8259A anytime after initialization.

INITIALIZATION COMMAND WORDS (ICWS)

General

Whenever a command is issued with A0 = 0 and D4 = 1, this is interpreted as Initialization Command Word 1 (ICW1). ICW1 starts the intiitalization sequence during which the following automatically occur.

a. The edge sense circuit is reset, which means that following initialization, an interrupt request (IR) input must make a low-to-high transistion to generate an interrupt.

b. The Interrupt Mask Register is cleared.

c. IR7 input is assigned priority 7.

d. The slave mode address is set to 7.

e. Special Mask Mode is cleared and Status Read is set to IRR.

f. If IC4 = 0, then all functions selected in ICW4 are set to zero. (Non-Buffered mode*, no Auto-EOI, MCS-80, 85 system).

***NOTE:**
Master/Slave in ICW4 is only used in the buffered mode.

Initialization Command Words 1 and 2 (ICW1, ICW2)

A_5–A_{15}: *Page starting address of service routines.* In an MCS 80/85 system, the 8 request levels will generate CALLs to 8 locations equally spaced in memory. These can be programmed to be spaced at intervals of 4 or 8 memory locations, thus the 8 routines will occupy a page of 32 or 64 bytes, respectively.

The address format is 2 bytes long (A_0–A_{15}). When the routine interval is 4, A_0–A_4 are automatically inserted by the 8259A, while A_5–A_{15} are programmed externally. When the routine interval is 8, A_0–A_5 are automatically inserted by the 8259A, while A_6–A_{15} are programmed externally.

The 8-byte interval will maintain compatibility with current software, while the 4-byte interval is best for a compact jump table.

In an 8086 system A_{15}–A_{11} are inserted in the five most significant bits of the vectoring byte and the 8259A sets the three least significant bits according to the interrupt level. A_{10}–A_5 are ignored and ADI (Address interval) has no effect.

LTIM: If LTIM = 1, then the 8259A will operate in the level interrupt mode. Edge detect logic on the interrupt inputs will be disabled.

ADI: CALL address interval. ADI = 1 then interval = 4; ADI = 0 then interval = 8.

SNGL: Single. Means that this is the only 8259A in the system. If SNGL = 1 no ICW3 will be issued.

IC4: If this bit is set—ICW4 has to be read. If ICW4 is not needed, set IC4 = 0.

Initialization Command Word 3 (ICW3)

This word is read only when there is more than one 8259A in the system and cascading is used, in which case SNGL = 0. It will load the 8-bit slave register. The functions of this register are:

a. In the master mode (either when SP = 1, or in buffered mode when M/S = 1 in ICW4) a "1" is set for each slave in the system. The master then will release byte 1 of the call sequence (for MCS-80/85 system) and will enable the corresponding slave to release bytes 2 and 3 (for 8086 only byte 2) through the cascade lines.

b. In the slave mode (either when $\overline{SP}$ = 0, or if BUF = 1 and M/S = 0 in ICW4) bits 2–0 identify the slave. The slave compares its cascade input with these bits and, if they are equal, bytes 2 and 3 of the call sequence (or just byte 2 for 8086) are released by it on the Data Bus.

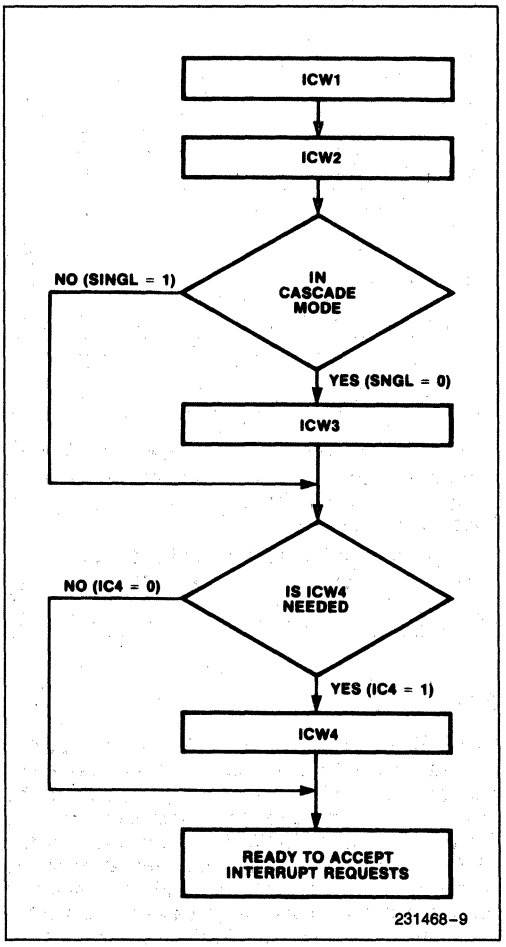


Figure 6. Initialization Sequence

Initialization Command Word 4 (ICW4)

SFNM: If SFNM = 1 the special fully nested mode is programmed.

BUF: If BUF = 1 the buffered mode is programmed. In buffered mode $\overline{SP}/\overline{EN}$ becomes an enable output and the master/slave determination is by M/S.

M/S: If buffered mode is selected: M/S = 1 means the 8259A is programmed to be a master, M/S = 0 means the 8259A is programmed to be a slave. If BUF = 0, M/S has no function.

AEOI: If AEOI = 1 the automatic end of interrupt mode is programmed.

μPM: Microprocessor mode: μPM = 0 sets the 8259A for MCS-80, 85 system operation, μPM = 1 sets the 8259A for 8086 system operation.

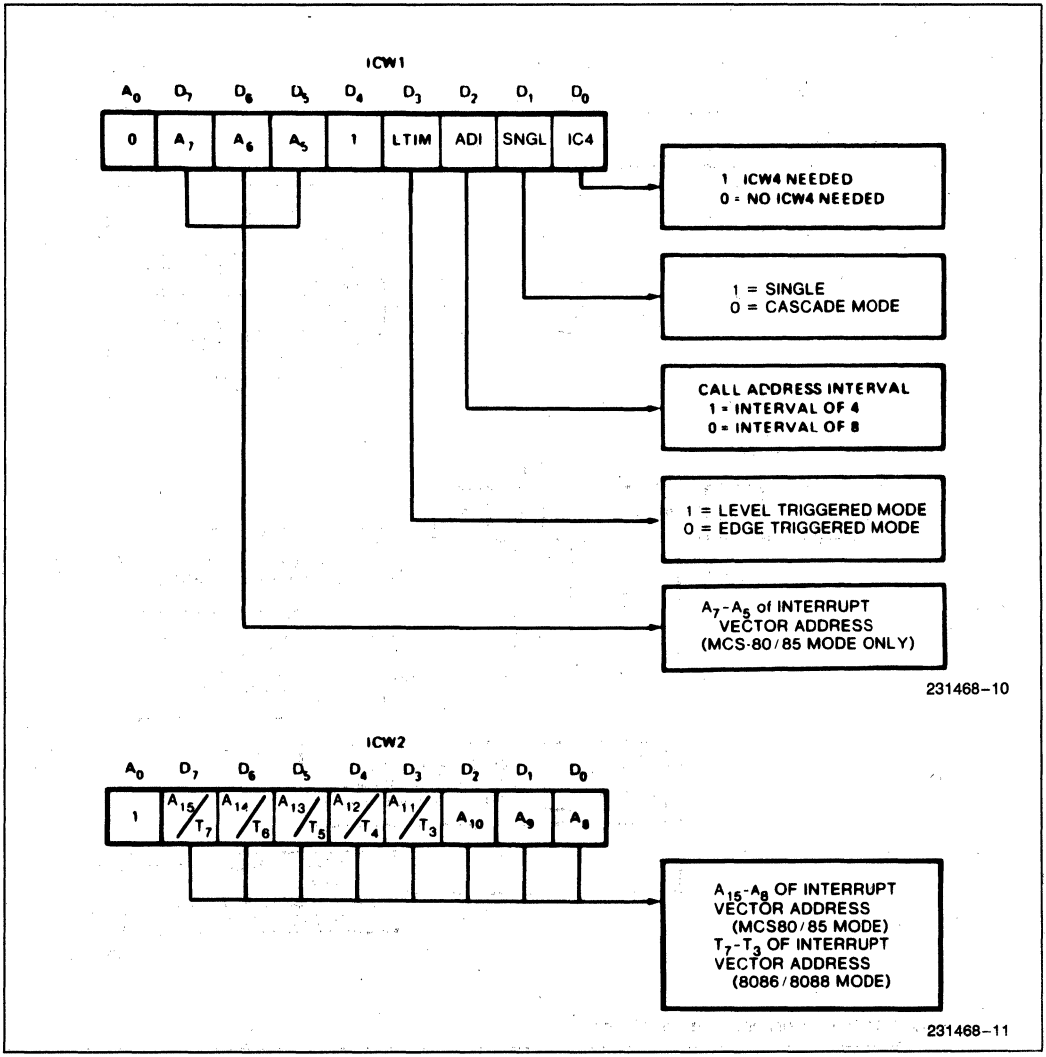
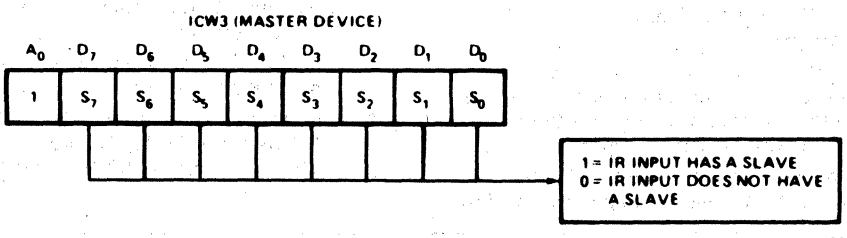
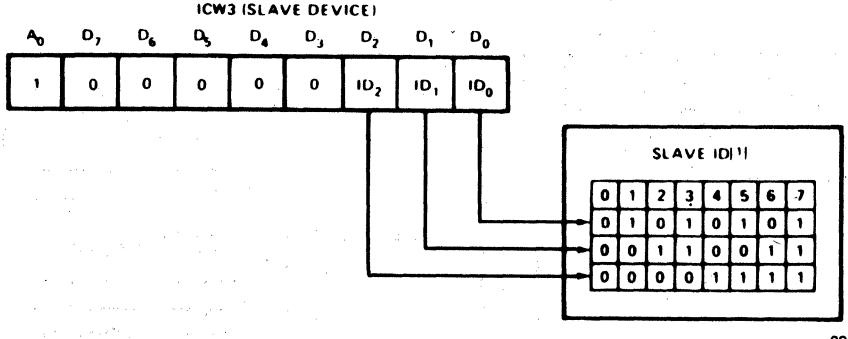


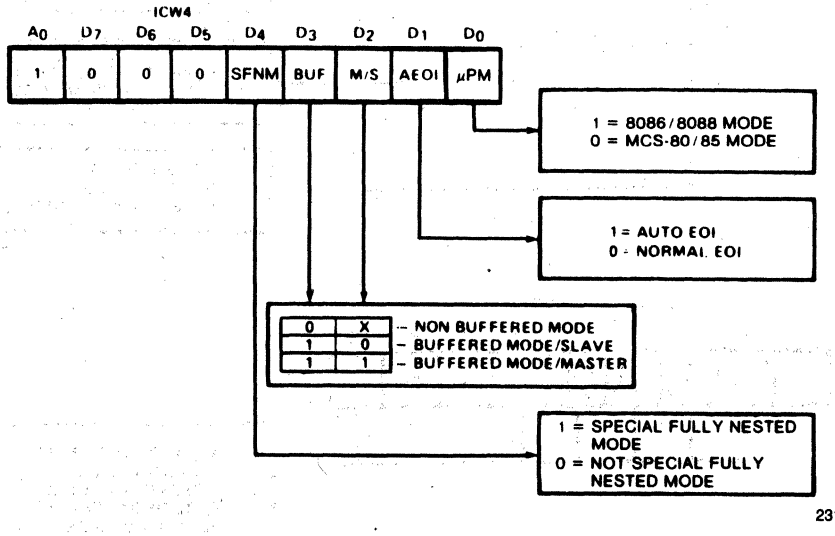
Figure 7. Initialization Command Word Format



231468-12



231468-13



231468-14

NOTE:
Slave ID is equal to the corresponding master IR input.

Figure 7. Initialization Command Word Format (Continued)

OPERATION COMMAND WORDS (OCWS)

After the Initialization Command Words (ICWs) are programmed into the 8259A, the chip is ready to accept interrupt requests at its input lines. However, during the 8259A operation, a selection of algorithms can command the 8259A to operate in various modes through the Operation Command Words (OCWs).

Operation Control Words (OCWs)

OCW1

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7	M6	M5	M4	M3	M2	M1	M0

OCW2

0	R	SL	EOI	0	0	L2	L1	L0

OCW3

0	0	ESMM	SMM	0	1	P	RR	RIS

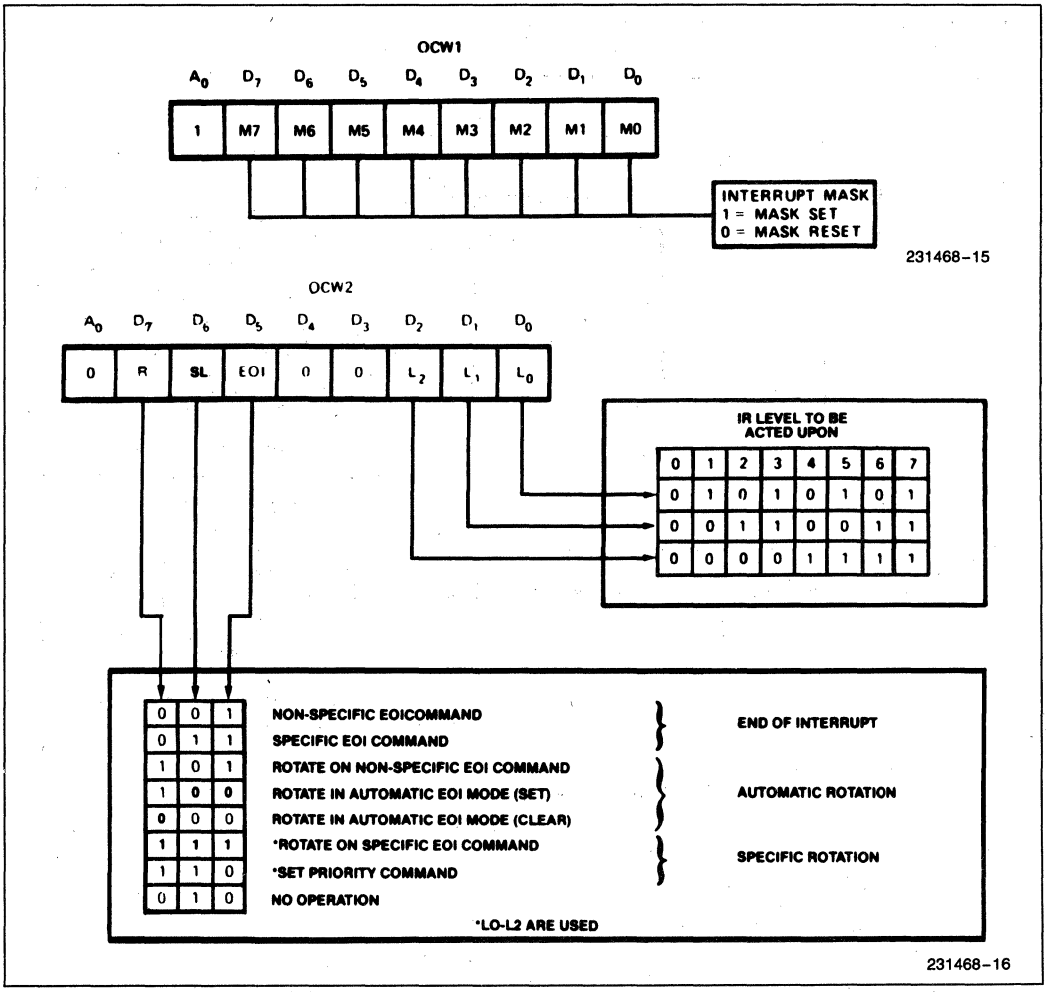


Figure 8. Operation Command Word Format

Operation Control Word 1 (OCW1)

OCW1 sets and clears the mask bits in the interrupt Mask Register (IMR). M_7–M_0 represent the eight mask bits. M = 1 indicates the channel is masked (inhibited), M = 0 indicates the channel is enabled.

Operation Control Word 2 (OCW2)

R, SL, EOI—These three bits control the Rotate and End of Interrupt modes and combinations of the two. A chart of these combinations can be found on the Operation Command Word Format.

L_2, L_1, L_0—These bits determine the interrupt level acted upon when the SL bit is active.

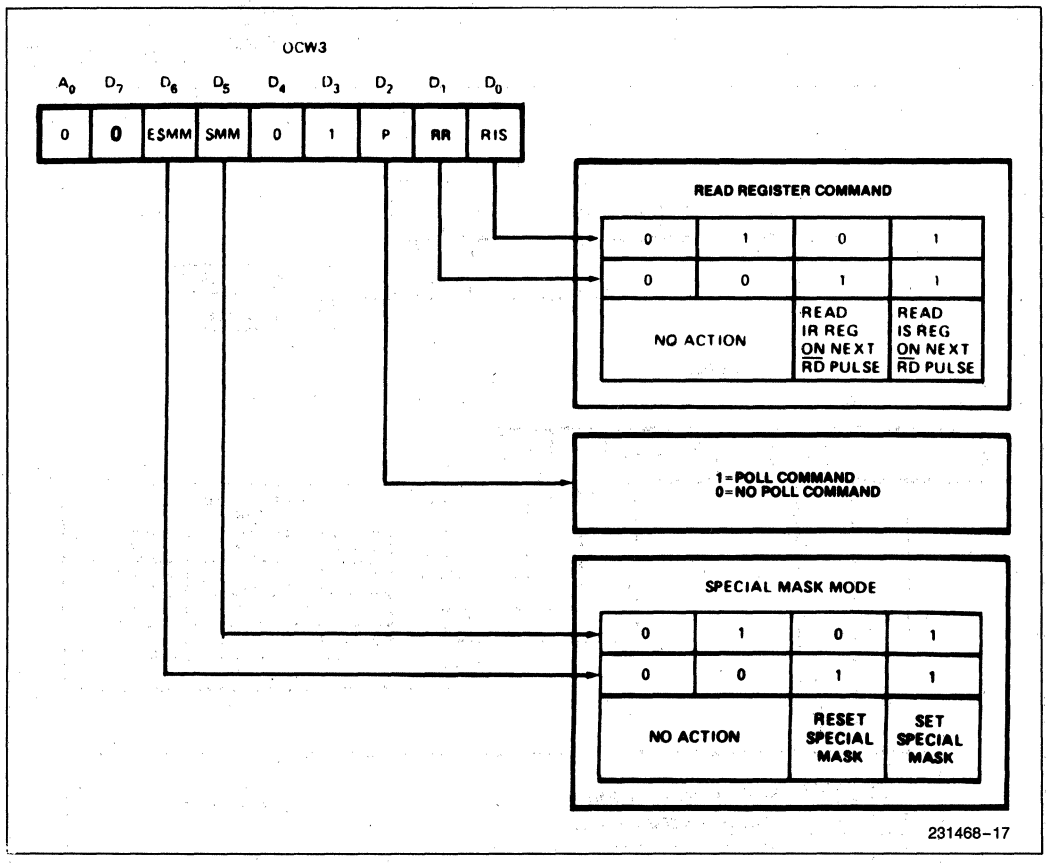


Figure 8. Operation Command Word Format (Continued)

Operation Control Word 3 (OCW3)

ESMM—Enable Special Mask Mode. When this bit is set to 1 it enables the SMM bit to set or reset the Special Mask Mode. When ESMM = 0 the SMM bit becomes a "don't care".

SMM—Special Mask Mode. If ESMM = 1 and SMM = 1 the 8259A will enter Special Mask Mode. If ESMM = 1 and SMM = 0 the 8259A will revert to normal mask mode. When ESMM = 0, SMM has no effect.

Fully Nested Mode

This mode is entered after initialization unless another mode is programmed. The interrupt requests are ordered in priority from 0 through 7 (0 highest). When an interrupt is acknowledged the highest priority request is determined and its vector placed on the bus. Additionally, a bit of the Interrupt Service register (ISO-7) is set. This bit remains set until the microprocessor issues an End of Interrupt (EOI) command immediately before returning from the service routine, or if AEOI (Automatic End of Interrupt) bit is set, until the trailing edge of the last INTA. While the IS bit is set, all further interrupts of the same or lower priority are inhibited, while higher levels will generate an interrupt (which will be acknowledged only if the microprocessor internal Interupt enable flip-flop has been re-enabled through software).

After the initialization sequence, IR0 has the highest prioirity and IR7 the lowest. Priorities can be changed, as will be explained, in the rotating priority mode.

End of Interrupt (EOI)

The In Service (IS) bit can be reset either automatically following the trailing edge of the last in sequence $\overline{\text{INTA}}$ pulse (when AEOI bit in ICW1 is set) or by a command word that must be issued to the 8259A before returning from a service routine (EOI command). An EOI command must be issued twice if in the Cascade mode, once for the master and once for the corresponding slave.

There are two forms of EOI command: Specific and Non-Specific. When the 8259A is operated in modes which perserve the fully nested structure, it can determine which IS bit to reset on EOI. When a Non-Specific EOI command is issued the 8259A will automatically reset the highest IS bit of those that are set, since in the fully nested mode the highest IS level was necessarily the last level acknowledged and serviced. A non-specific EOI can be issued with OCW2 (EOI = 1, SL = 0, R = 0).

When a mode is used which may disturb the fully nested structure, the 8259A may no longer be able to determine the last level acknowledged. In this case a Specific End of Interrupt must be issued which includes as part of the command the IS level to be reset. A specific EOI can be issued with OCW2 (EOI = 1, SL = 1, R = 0, and L0–L2 is the binary level of the IS bit to be reset).

It should be noted that an IS bit that is masked by an IMR bit will not be cleared by a non-specific EOI if the 8259A is in the Special Mask Mode.

Automatic End of Interrupt (AEOI) Mode

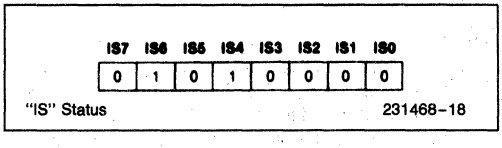
If AEOI = 1 in ICW4, then the 8259A will operate in AEOI mode continuously until reprogrammed by ICW4. in this mode the 8259A will automatically perform a non-specific EOI operation at the trailing edge of the last interrupt acknowledge pulse (third pulse in MCS-80/85, second in 8086). Note that from a system standpoint, this mode should be used only when a nested multilevel interrupt structure is not required within a single 8259A.

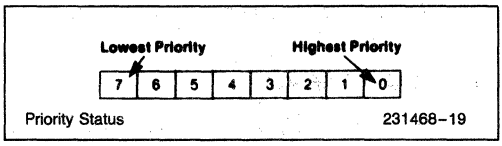
The AEOI mode can only be used in a master 8259A and not a slave. 8259As with a copyright date of 1985 or later will operate in the AEOI mode as a master or a slave.

Automatic Rotation (Equal Priority Devices)

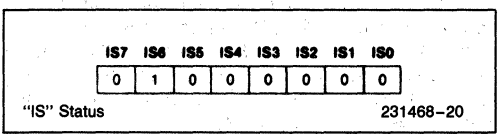
In some applications there are a number of interrupting devices of equal priority. In this mode a device, after being serviced, receives the lowest priority, so a device requesting an interrupt will have to wait, in the worst case until each of 7 other devices are serviced at most *once*. For example, if the priority and "in service" status is:

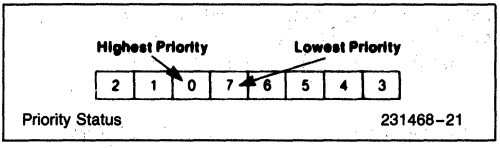
Before Rotate (IR4 the highest prioirity requiring service)





After Rotate (IR4 was serviced, all other priorities rotated correspondingly)





There are two ways to accomplish Automatic Rotation using OCW2, the Rotation on Non-Specific EOI Command (R = 1, SL = 0, EOI = 1) and the Rotate in Automatic EOI Mode which is set by (R = 1, SL = 0, EOI = 0) and cleared by (R = 0, SL = 0, EOI = 0).

Specific Rotation (Specific Priority)

The programmer can change priorities by programming the bottom priority and thus fixing all other priorities; i.e., if IR5 is programmed as the bottom priority device, then IR6 will have the highest one.

The Set Priority command is issued in OCW2 where: R = 1, SL = 1, L0–L2 is the binary priority level code of the bottom priority device.

Observe that in this mode internal status is updated by software control during OCW2. However, it is independent of the End of Interrupt (EOI) command (also executed by OCW2). Priority changes can be executed during an EOI command by using the Rotate on Specific EOI command in OCW2 (R = 1, SL = 1, EOI = 1 and L0–L2 = IR level to receive bottom priority).

Interrupt Masks

Each Interrupt Request input can bem masked individually by the Interrupt Mask Register (IMR) programmed through OCW1. Each bit in the IMR masks one interrupt channel if it is set (1). Bit 0 masks IR0, Bit 1 masks IR1 and so forth. Masking an IR channel does not affect the other channels operation.

Special Mask Mode

Some applications may require an interrupt service routine to dynamically alter the system priority structure during its execution under software control. For example, the routine may wish to inhibit lower priority requests for a portion of its execution but enable some of them for another portion.

The difficulty here is that if an Interrupt Request is acknowledged and an End of Interrupt command did not reset its IS bit (i.e., while executing a service routine), the 8259A would have inhibited all lower priority requests with no easy way for the routine to enable them.

That is where the Special Mask Mode comes in. In the special Mask Mode, when a mask bit is set in OCW1, it inhibits further interrupts at that level *and enables* interrupts from *all other* levels (lower as well as higher) that are not masked.

Thus, any interrupts may be selectively enabled by loading the mask register.

The special Mask Mode is set by OWC3 where: SSMM = 1, SMM = 1, and cleared where SSMM = 1, SMM = 0.

Poll Command

In Poll mode the INT output functions as it normally does. The microprocessor should ignore this output. This can be accomplished either by not connecting the INT output or by masking interrupts within the microprocessor, thereby disabling its interrupt input. Service to devices is achieved by software using a Poll command.

The Poll command is issued by setting P = '1" in OCW3. The 8259A treats the next $\overline{RD}$ pulse to the 8259A (i.e., $\overline{RD}$ = 0, $\overline{CS}$ = 0) as an interrupt acknowledge, sets the appropriate IS bit if there is a request, and reads the priority level. Interrupt is frozen from $\overline{WR}$ to $\overline{RD}$.

The word enabled onto the data bus during $\overline{RD}$ is:

D7	D6	D5	D4	D3	D2	D1	D0
I	—	—	—	—	W2	W1	W0

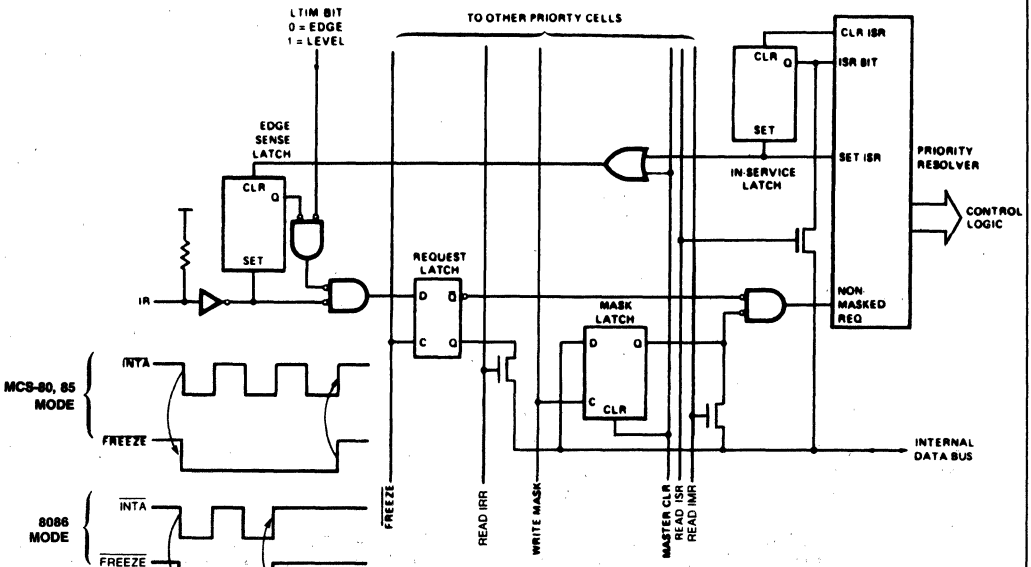
W0–W2: Binary code of the highest priority level requesting service.

I: Equal to "1" if there is an interrupt.

This mode is useful if there is a routine command common to several levels so that the $\overline{INTA}$ sequence is not needed (saves ROM space). Another application is to use the poll mode to expand the number of priority levels to more than 64.

Reading the 8259A Status

The input status of several internal registers can be read to update the user information on the system.



NOTES:
1. Master clear active only during ICW1.
2. $\overline{\text{FREEZE}}$ is active during $\overline{\text{INTA}}$ and poll sequences only.
3. Truth Table for a D-Latch.

C	D	Q	Operation
1	Di	Di	Follow
0	X	Qn-1	Hold

231468-22

Figure 9. Priority Cell—Simplified Logic Diagram

The following registers can be read via OCW3 (IRR and ISR or OCW1 [IMR]).

Interrupt Request Register (IRR): 8-bit register which contains the levels requesting an interrupt to be acknowledged. The highest request level is reset from the IRR when an interrupt is acknowledged. (Not affected by IMR.)

In-Service Register (ISR): 8-bit register which contains the priority levels that are being serviced. The ISR is updated when an End of Interrupt Command is issued.

Interrupt Mask Register: 8-bit register which contains the interrupt request lines which are masked.

The IRR can be read when, prior to the RD pulse, a Read Register Command is issued with OCW3 (RR = 1, RIS = 0.)

The ISR can be read, when, prior to the RD pulse, a Read Register Command is issued with OCW3 (RR = 1, RIS = 1).

There is no need to write an OCW3 before every status read operation, as long as the status read corresponds with the previous one; i.e., the 8259A "remembers" whether the IRR or ISR has been previously selected by the OCW3. This is not true when poll is used.

After initialization the 8259A is set to IRR.

For reading the IMR, no OCW3 is needed. The output data bus will contain the IMR whenever $\overline{\text{RD}}$ is active and A0 = 1 (OCW1).

Polling overrides status read when P = 1, RR = 1 in OCW3.

Edge and Level Triggered Modes

This mode is programmed using bit 3 in ICW1.

If LTIM = '0', an interrupt request will be recognized by a low to high transition on an IR input. The IR input can remain high without generating another interrupt.

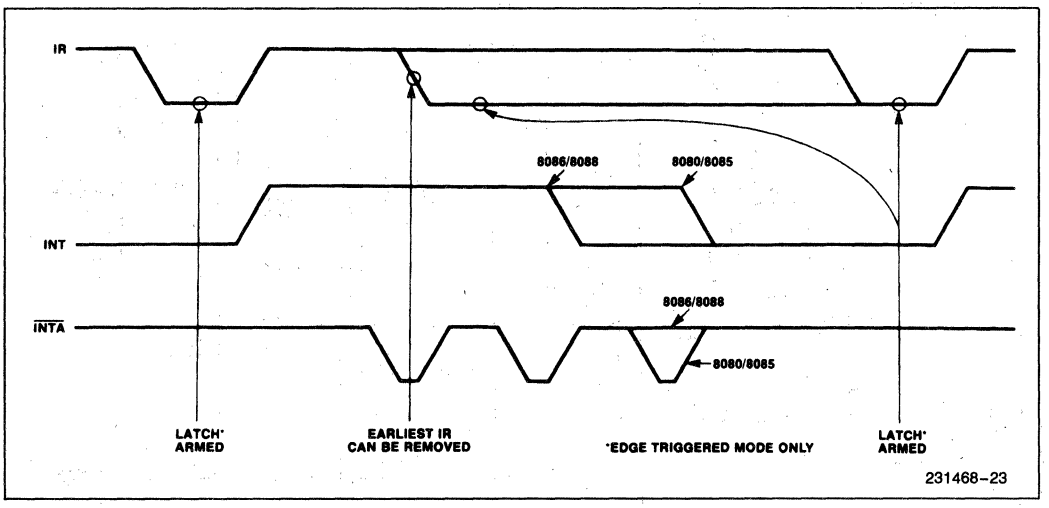


Figure 10. IR Triggering Timing Requirements

If LTIM = '1', an interrupt request will be recognized by a 'high' level on IR Input, and there is no need for an edge detection. The interrupt request must be removed before the EOI command is issued or the CPU interrupts is enabled to prevent a second interrupt from occurring.

The priority cell diagram shows a conceptual circuit of the level sensitive and edge sensitive input circuitry of the 8259A. Be sure to note that the request latch is a transparent D type latch.

In both the edge and level triggered modes the IR inputs must remain high until after the falling edge of the first $\overline{INTA}$. If the IR input goes low before this time a DEFAULT IR7 will occur when the CPU acknowledges the interrupt. This can be a useful safeguard for detecting interrupts caused by spurious noise glitches on the IR inputs. To implement this feature the IR7 routine is used for "clean up" simply executing a return instruction, thus ignoring the interrupt. If IR7 is needed for other purposes a default IR7 can still be detected by reading the ISR. A normal IR7 interrupt will set the corresponding ISR bit, a default IR7 won't. If a default IR7 routine occurs during a normal IR7 routine, however, the ISR will remain set. In this case it is necessary to keep track of whether or not the IR7 routine was previously entered. If another IR7 occurs it is a default.

The Special Fully Nest Mode

This mode will be used in the case of a big system where cascading is used, and the priority has to be conserved within each slave. In this case the fully nested mode will be programmed to the master (using ICW4). This mode is similar to the normal nested mode with the following exceptions:

a. When an interrupt request from a certain slave is in service this slave is not locked out from the master's priority logic and further interrupt requests from higher priority IR's within the slave will be recognized by the master and will initiate interrupts to the processor. (In the normal nested mode a slave is masked out when its request is in service and no higher requests from the same slave can be serviced.)

b. When exiting the Interrupt Service routine the software has to check whether the interrupt serviced was the only one from that slave. This is done by sending a non-specific End of Interrupt (EOI) command to the slave and then reading its In-Service register and checking for zero. If it is empty, a non-specific EOI can be sent to the master too. If not, no EOI should be sent.

Buffered Mode

When the 8259A is used in a large system where bus driving buffers are required on the data bus and the cascading mode is used, there exists the problem of enabling buffers.

The buffered mode will structure the 8259A to send an enable signal on $\overline{SP}/\overline{EN}$ to enable the buffers. In this mode, whenever the 8259A's data bus outputs are enabled, the $\overline{SP}/\overline{EN}$ output becomes active.

This modification forces the use of software programming to determine whether the 8259A is a master or a slave. Bit 3 in ICW4 programs the buffered mode, and bit 2 in ICW4 determines whether it is a master or a slave.

CASCADE MODE

The 8259A can be easily interconnected in a system of one master with up to eight slaves to handle up to 64 priority levels.

The master controls the slaves through the 3 line cascade bus. The cascade bus acts like chip selects to the slaves during the $\overline{INTA}$ sequence.

In a cascade configuration, the slave interrupt outputs are connected to the master interrupt request inputs. When a slave request line is activated and afterwards acknowledged, the master will enable the corresponding slave to release the device routine address during bytes 2 and 3 of $\overline{INTA}$. (Byte 2 only for 8086/8088).

The cascade bus lines are normally low and will contain the slave address code from the trailing edge of the first $\overline{INTA}$ pulse to the trailing edge of the third pulse. Each 8259A in the system must follow a separate initialization sequence and can be programmed to work in a different mode. An EOI command must be issued twice: once for the master and once for the corresponding slave. An address decoder is required to activate the Chip Select (CS) input of each 8259A.

The cascade lines of the Master 8259A are activated only for slave inputs, non-slave inputs leave the cascade line inactive (low).

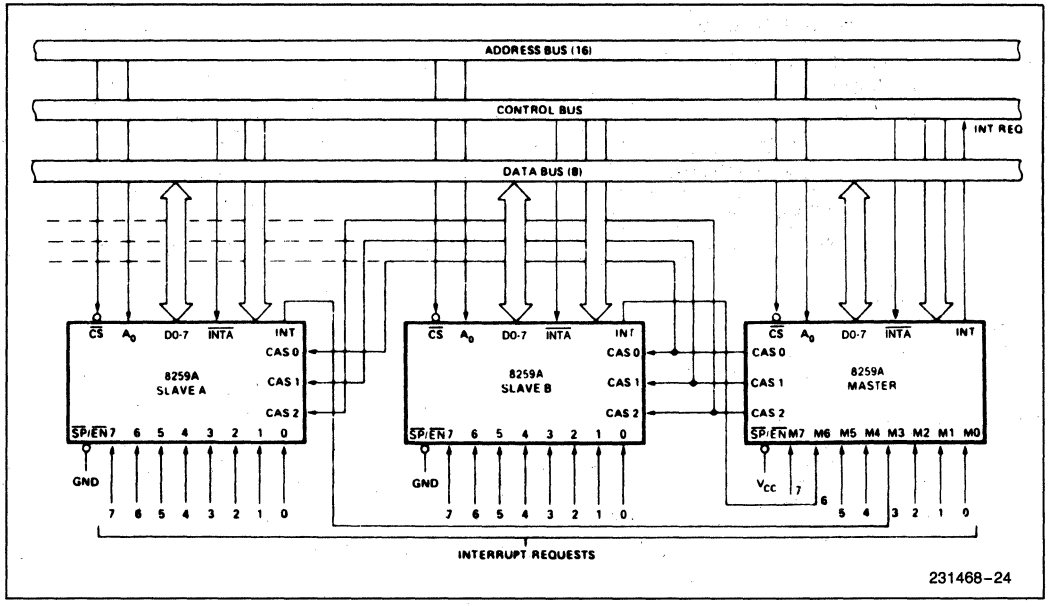


Figure 11. Cascading the 8259A

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature .....−65°C to +150°C
Voltage on Any Pin
with Respect to Ground.....−0.5V to +7V
Power Dissipation 1W

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ± 10%

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2.0*	V_{CC} + 0.5V	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −400 μA
$V_{OH(INT)}$	Interrupt Output High Voltage	3.5		V	I_{OH} = −100 μA
		2.4		V	I_{OH} = −400 μA
I_{LI}	Input Load Current	−10	+10	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{LOL}	Output Leakage Current	−10	+10	μA	0.45V ≤ V_{OUT} ≤ V_{CC}
I_{CC}	V_{CC} Supply Current		85	mA	
I_{LIR}	IR Input Load Current		−300	μA	V_{IN} = 0
			10	μA	V_{IN} = V_{CC}

***NOTE:**
For Extended Temperature EXPRESS V_{IH} = 2.3V.

CAPACITANCE T_A = 25°C; V_{CC} = GND = 0V

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	fc = 1 MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured Pins Returned to V_{SS}

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±10%

TIMING REQUIREMENTS

Symbol	Parameter	8259A		8259A-2		Units	Test Conditions
		Min	Max	Min	Max		
TAHRL	AO/$\overline{CS}$ Setup to $\overline{RD}$/$\overline{INTA}$ ↓	0		0		ns	
TRHAX	AO/$\overline{CS}$ Hold after $\overline{RD}$/$\overline{INTA}$ ↑	0		0		ns	
TRLRH	$\overline{RD}$ Pulse Width	235		160		ns	
TAHWL	AO/$\overline{CS}$ Setup to $\overline{WR}$ ↓	0		0		ns	
TWHAX	AO/$\overline{CS}$ Hold after $\overline{WR}$ ↑	0		0		ns	
TWLWH	$\overline{WR}$ Pulse Width	290		190		ns	
TDVWH	Data Setup to $\overline{WR}$ ↑	240		160		ns	
TWHDX	Data Hold after $\overline{WR}$ ↑	0		0		ns	
TJLJH	Interrupt Request Width (Low)	100		100		ns	See Note 1
TCVIAL	Cascade Setup to Second or Third $\overline{INTA}$ ↓ (Slave Only)	55		40		ns	
TRHRL	End of $\overline{RD}$ to Next $\overline{RD}$ End of $\overline{INTA}$ to Next $\overline{INTA}$ within an $\overline{INTA}$ Sequence Only	160		100		ns	
TWHWL	End of $\overline{WR}$ to Next $\overline{WR}$	190		100		ns	
*TCHCL	End of Command to Next Command (Not Same Command Type)	500		150		ns	
	End of $\overline{INTA}$ Sequence to Next $\overline{INTA}$ Sequence.	500		300			

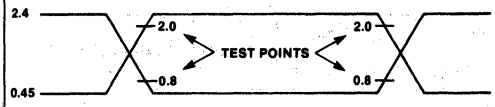
*Worst case timing for TCHCL in an actual microprocessor system is typically much greater than 500 ns (i.e. 8085A = 1.6 μs, 8085A-2 = 1 μs, 8086 = 1 μs, 8086-2 = 625 ns)

NOTE:
This is the low time required to clear the input latch in the edge triggered mode.

TIMING RESPONSES

Symbol	Parameter	8259A		8259A-2		Units	Test Conditions
		Min	Max	Min	Max		
TRLDV	Data Valid from $\overline{RD}$/$\overline{INTA}$ ↓		200		120	ns	C of Data Bus = 100 pF
TRHDZ	Data Float after $\overline{RD}$/$\overline{INTA}$ ↑	10	100	10	85	ns	C of Data Bus Max Test C = 100 pF Min Test C = 15 pF
TJHIH	Interrupt Output Delay		350		300	ns	
TIALCV	Cascade Valid from First $\overline{INTA}$ ↓ (Master Only)		565		360	ns	C_{INT} = 100 pF
TRLEL	Enable Active from $\overline{RD}$ ↓ or $\overline{INTA}$ ↓		125		100	ns	$C_{CASCADE}$ = 100 pF
TRHEH	Enable Inactive from $\overline{RD}$ ↑ or $\overline{INTA}$ ↑		150		150	ns	
TAHDV	Data Valid from Stable Address		200		200	ns	
TCVDV	Cascade Valid to Valid Data		300		200	ns	

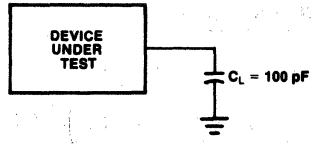
A.C. TESTING INPUT/OUTPUT WAVEFORM



231468-25

A.C. Testing: Inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0". Timing measurements are made at 2.0V for a logic "1" and 0.8V for a logic "0".

A.C. TESTING LOAD CIRCUIT

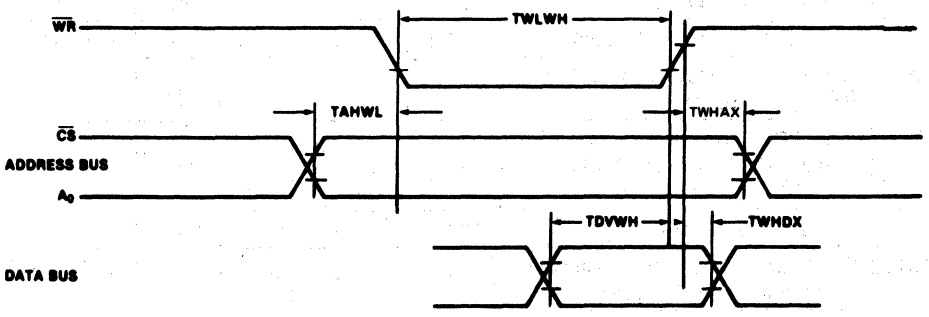


231468-26

C_L = 100 pF
C_L Includes Jig Capacitance

WAVEFORMS

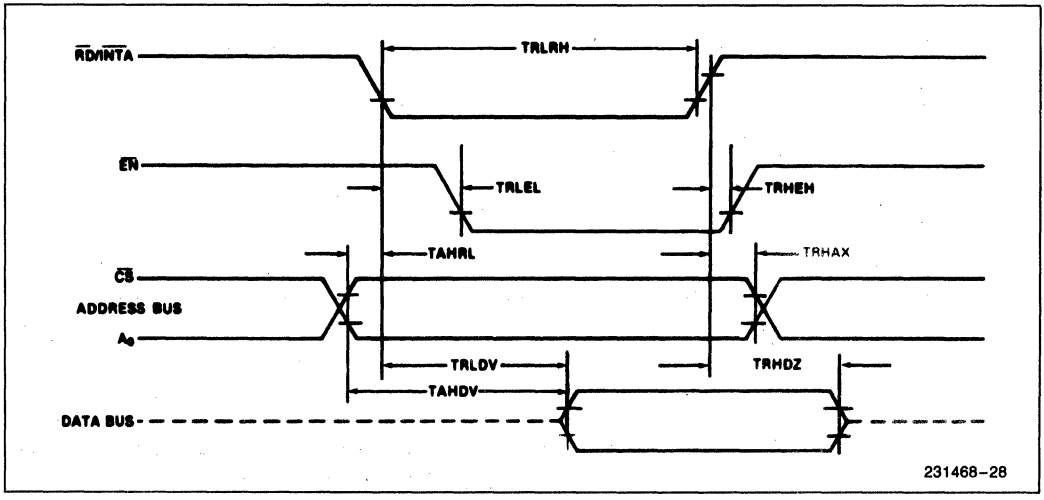
WRITE



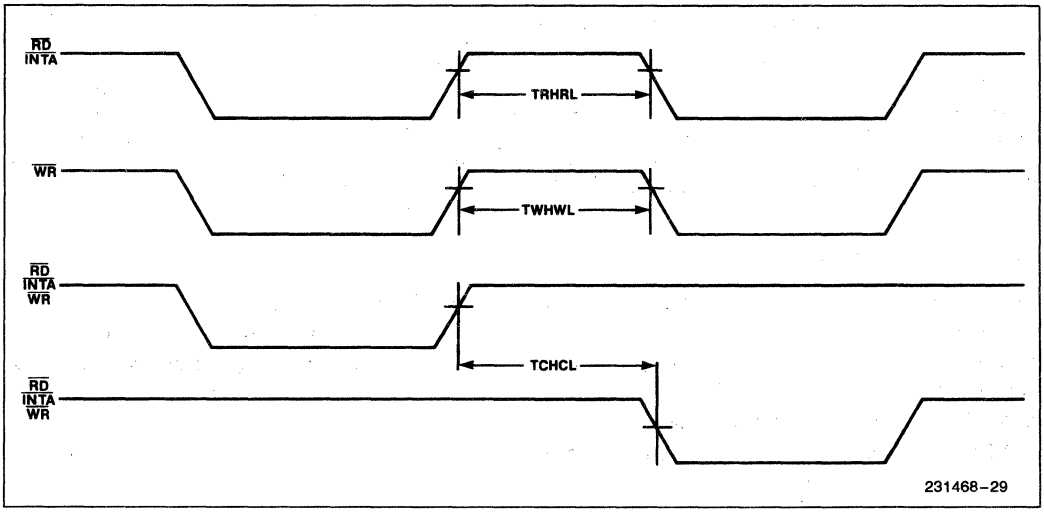
231468-27

WAVEFORMS (Continued)

READ/INTA

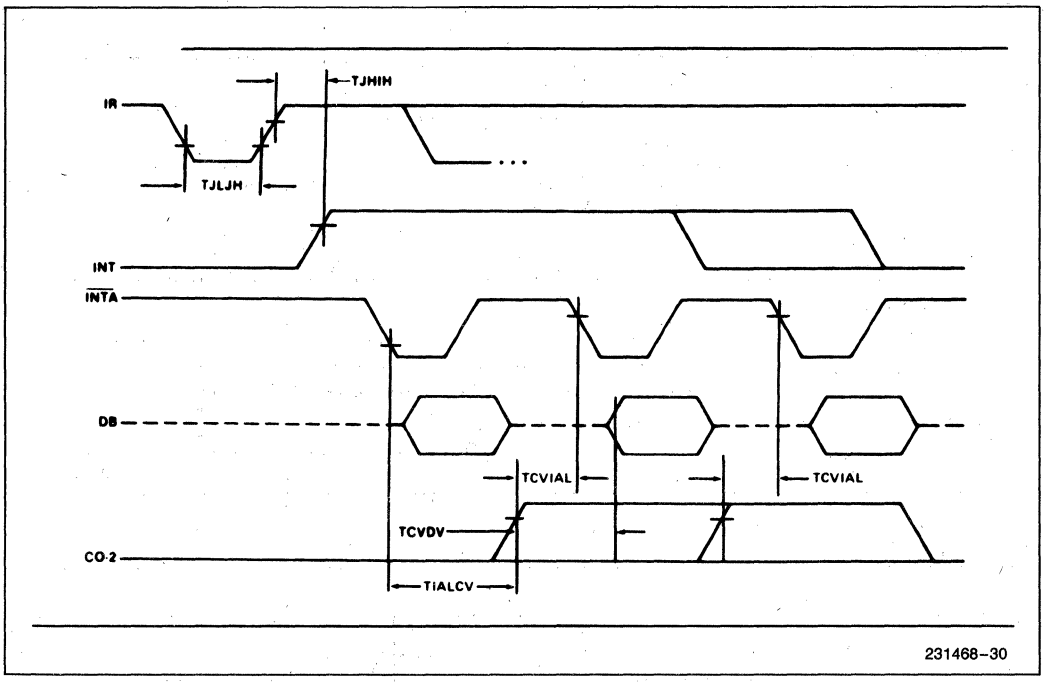


OTHER TIMING



WAVEFORMS (Continued)

$\overline{\text{INTA}}$ SEQUENCE



NOTES:
Interrupt output must remain HIGH at least until leading edge of first $\overline{\text{INTA}}$.
1. Cycle 1 in 8086, 8088 systems, the Data Bus is not active.

Data Sheet Revision Review

The following changes have been made since revision 2 of the 8259A data sheet.

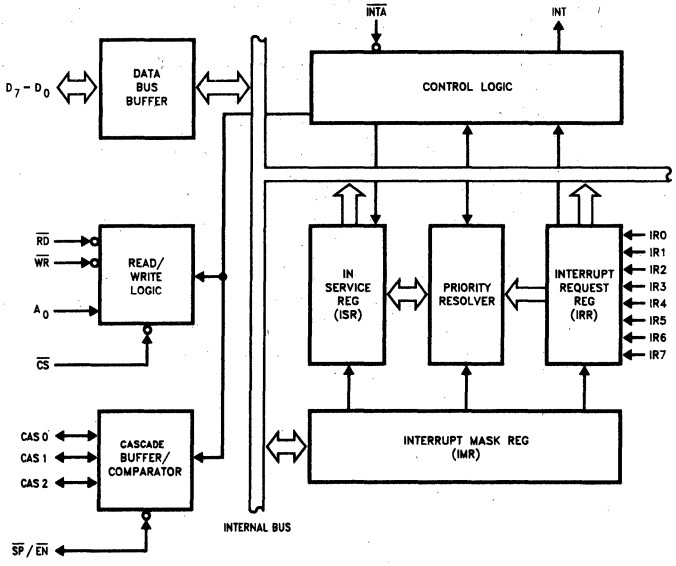
1. The first paragraph of the Poll Command section was rewritten to clarify the status of the INT pin.
2. A paragraph was added to the Interrupt Sequence section to indicate the status of the INT pin during multiple interrupts.
3. A reference to PLCC packaging was added.
4. All references to the 8259A-8 have been deleted.

82C59A-2
CHMOS Programmable Interrupt Controller

- **Pin Compatible with NMOS 8259A-2**
- **Eight-Level Priority Controller**
- **Expandable to 64 levels**
- **Programmable Interrupt Modes**
- **Low Standby Power—10 μA**
- **Individual Request Mask Capability**
- **80C86/88 and 8080/85/86/88 Compatible**
- **Fully Static Design**
- **Single 5V Power Supply**
- **Available in 28-Pin Plastic DIP**
  (See Packaging Spec., Order #231369)

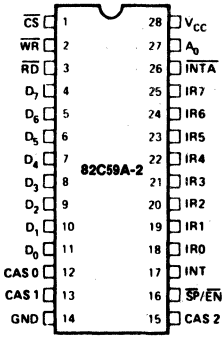
The Intel 82C59A-2 is a high performance CHMOS version of the NMOS 8259A-2 Priority Interrupt Controller. The 82C59A-2 is designed to relieve the system CPU from the task of polling in a multi-level priority interrupt system. The high speed and industry standard configuration of the 82C59A-2, make it compatible with microprocessors such as the 80C86/88, 8086/88 and 8080/85.

The 82C59A-2 can handle up to 8 vectored priority interrupts for the CPU and is cascadable to 64 without additional circuitry. It is designed to minimize the software and real time overhead in handling multi-level priority interrupts. Two modes of operation make the 82C59A-2 optimal for a variety of system requirements. Static CHMOS circuit design, requiring no clock input, insures low operating power. It is packaged in a 28-pin plastic DIP.



231201-1

Figure 1. Block Diagram



231201-2

Figure 2. Pin Configuration

October 1988
Order Number: 231201-004

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
V_{CC}	28	I	**SUPPLY:** +5V Supply.
GND	14	I	**GROUND.**
$\overline{CS}$	1	I	**CHIP SELECT:** A low on this pin enables $\overline{RD}$ and $\overline{WR}$ communication between the CPU and the 82C59A-2. INTA functions are independent of CS.
$\overline{WR}$	2	I	**WRITE:** A low on this pin when $\overline{CS}$ is low enables the 82C59A-2 to accept command words from the CPU.
$\overline{RD}$	3	I	**READ:** A low on this pin when $\overline{CS}$ is low enables the 82C59A-2 to release status onto the data bus for the CPU.
D_7–D_0	4–11	I/O	**BIDIRECTIONAL DATA BUS:** Control, status and interrupt-vector information is transferred via this bus.
CAS_0–CAS_2	12, 13, 15	I/O	**CASCADE LINES:** The CAS lines form a private 82C59A-2 bus to control a multiple 82C59A-2 structure. These pins are outputs for a master 82C59A-2 and inputs for a slave 82C59A-2.
$\overline{SP}/\overline{EN}$	16	I/O	**SLAVE PROGRAM/ENABLE BUFFER:** This is a dual function pin. When in the Buffered Mode it can be used as an output to control buffer transceivers (EN). When not in the buffered mode it is used as an input to designate a master (SP = 1) or slave (SP = 0).
INT	17	O	**INTERRUPT:** This pin goes high whenever a valid interrupt request is asserted. It is used to interrupt the CPU, thus it is connected to the CPU's interrupt pin.
IR_0–IR_7	18–25	I	**INTERRUPT REQUESTS:** Asynchronous inputs. An interrupt request is executed by raising an IR input (low to high), and holding it high until it is acknowledged (Edge Triggered Mode), or just by a high level on an IR input (Level Triggered Mode). Internal pull-up resistors are implemented on IR0–7.
$\overline{INTA}$	26	I	**INTERRUPT ACKNOWLEDGE:** This pin is used to enable 82C59A-2 interrupt-vector data onto the data bus by a sequence of interrupt acknowledge pulses issued by the CPU.
A_0	27	I	**AO ADDRESS LINE:** This pin acts in conjunction with the $\overline{CS}$, $\overline{WR}$, and $\overline{RD}$ pins. It is used by the 82C59A-2 to decipher various Command Words the CPU writes and status the CPU wishes to read. It is typically connected to the CPU A0 address line (A1 for 80C86, 80C88).

FUNCTIONAL DESCRIPTION

Interrupts in Microcomputer Systems

Microcomputer system design requires that I/O devices such as keyboards, displays, sensors and other components receive servicing in an efficient manner so that large amounts of the total system tasks can be assumed by the microcomputer with little or no effect on throughput.

The most common method of servicing such devices is the *Polled* approach. This is where the processor must test each device in sequence and in effect "ask" each one if it needs servicing. It is easy to see that a large portion of the main program is looping through this continuous polling cycle and that such a method would have a serious, detrimental effect on system throughput, thus limiting the tasks that could be assumed by the microcomputer and reducing the cost effectiveness of using such devices.

A more desirable method would be one that would allow the microprocessor to be executing its main program and only stop to service peripheral devices when it is told to do so by the device itself. In effect, the method would provide an external asynchronous input that would inform the processor that it should complete whatever instruction that is currently being executed and fetch a new routine that will service the requesting device. Once this servicing is complete, however, the processor would resume exactly where it left off.

This method is called *Interrupt*. It is easy to see that system throughput would drastically increase, and thus more tasks could be assumed by the microcomputer to further enhance its cost effectiveness.

The Programmable Interrupt Controller (PIC) functions as an overall manager in an Interrupt-Driven system environment. It accepts requests from the peripheral equipment, determines which of the incoming requests is of the highest importance (priority), ascertains whether the incoming request has a higher priority value than the level currently being serviced, and issues an interrupt to the CPU based on this determination.

Each peripheral device or structure usually has a special program or "routine" that is associated with its specific functional or operational requirements; this is referred to as a "service routine". The PIC, after issuing an Interrupt to the CPU, must somehow input information into the CPU that can "point" the Program Counter to the service routine associated with the requesting device. This "pointer" is an address in a vectoring table and will often be referred to, in this document, as vectoring data.

The 82C59A-2

The 82C59A-2 is a device specifically designed for use in real time, interrupt driven microcomputer sys-

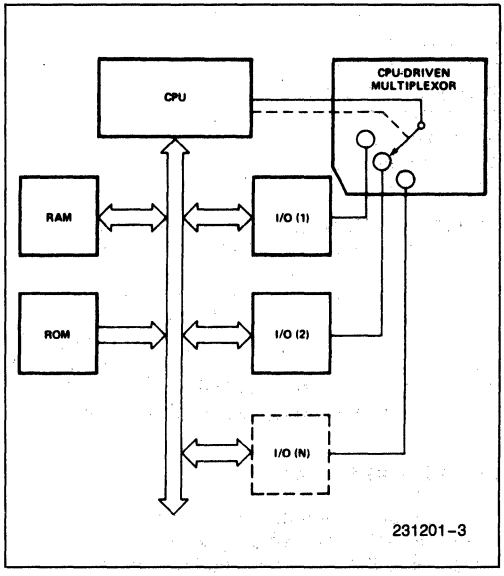


Figure 3a. Polled Method

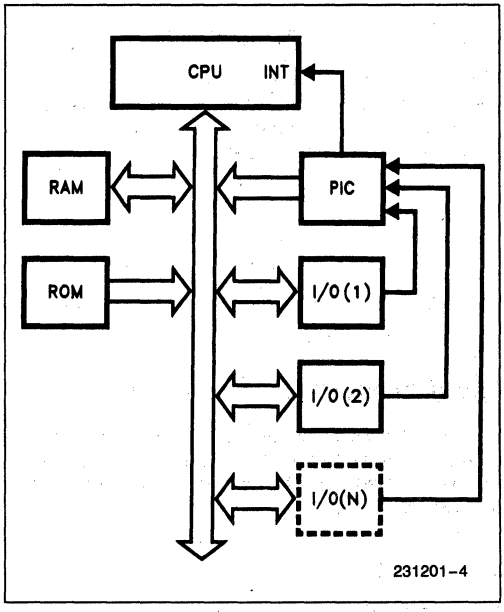


Figure 3b. Interrupt Method

tems. It manages eight levels or requests and has built-in features for expandability to other 82C59A-2's (up to 64 levels). It is programmed by the system's software as an I/O peripheral. A selection of priority modes is available to the programmer so that the manner in which the requests are processed by the 82C59A-2 can be configured to match system requirements. The priority modes can be changed or reconfigured dynamically at any time during the main program. This means that the complete interrupt structure can be defined as required, based on the total system environment.

INTERRUPT REQUEST REGISTER (IRR) AND IN-SERVICE REGISTER (ISR)

The interrupts at the IR input lines are handled by two registers in cascade, the Interrupt Request Register (IRR) and the In-Service Register (ISR). The IRR is used to store all the interrupt levels which are requesting service; and the ISR is used to store all the interrupt levels which are being serviced.

PRIORITY RESOLVER

This logic block determines the priorities of the bits set in the IRR. The highest priority is selected and strobed into the corresponding bit of the ISR during $\overline{INTA}$ pulse.

INTERRUPT MASK REGISTER (IMR)

The IMR stores the bits which mask the interrupt lines to be masked. The IMR operates on the IRR. Masking of a higher priority input will not affect the interrupt request lines of lower priority.

INT (INTERRUPT)

This output goes directly to the CPU interrupt input. The V_{OH} level on this line is designed to be fully compatible with the 8080A, 8085A, 80C88 and 80C86 input levels.

$\overline{INTA}$ (INTERRUPT ACKNOWLEDGE)

$\overline{INTA}$ pulses will cause the 82C59A-2 to release vectoring information onto the data bus. The format of this data depends on the system mode (μPM) of the 82C59A-2.

DATA BUS BUFFER

This 3-state, bidirectional 8-bit buffer is used to interface the 82C59A-2 to the system Data Bus. Control words and status information are transferred through the Data Bus Buffer.

READ/WRITE CONTROL LOGIC

The function of this block is to accept OUTput commands from the CPU. It contains the Initialization Command Word (ICW) registers and Operation Command Word (OCW) registers which store the various control formats for device operation. This function block also allows the status of the 82C59A-2 to be transferred onto the Data Bus.

$\overline{CS}$ (CHIP SELECT)

A LOW on this input enables the 82C59A-2. No reading or writing of the chip will occur unless the device is selected.

$\overline{WR}$ (WRITE)

A LOW on this input enables the CPU to write control words (ICWs and OCWs) to the 82C59A-2.

$\overline{RD}$ (READ)

A LOW on this input enables the 82C59A-2 to send the status of the Interrupt Request Register (IRR), In Service Register (ISR), the Interrupt Mask Register (IMR), or the Interrupt level onto the Data Bus.

A_0

This input signal is used in conjunction with $\overline{WR}$ and $\overline{RD}$ signals to write commands into the various command registers, as well as reading the various status registers of the chip. This line can be tied directly to one of the address lines.

THE CASCADE BUFFER/COMPARATOR

This function block stores and compares the IDs of all 82C59A-2's used in the system. The associated three I/O pins (CAS0-2) are outputs when the 82C59A-2 is used as a master and are inputs when the 82C59A-2 is used as a slave. As a master, the 82C59A-2 sends the ID of the interrupting slave device onto the CAS0-2 lines. The slave thus selected will send its preprogrammed subroutine address onto the Data Bus during the next one or two consecutive $\overline{INTA}$ pulses. (See section "Cascading the 82C59A-2".)

INTERRUPT SEQUENCE

The powerful features of the 82C59A-2 in a microcomputer system are its programmability and the interrupt routine addressing capability. The latter allows direct or indirect jumping to the specific interrupt routine requested without any polling of the interrupting devices. The normal sequence of events during an interrupt depends on the type of CPU being used.

The events occur as follows in an MCS-80/85 system:

1. One or more of the INTERRUPT REQUEST Lines (IR7–0) are raised high, setting the corresponding IRR bit(s).

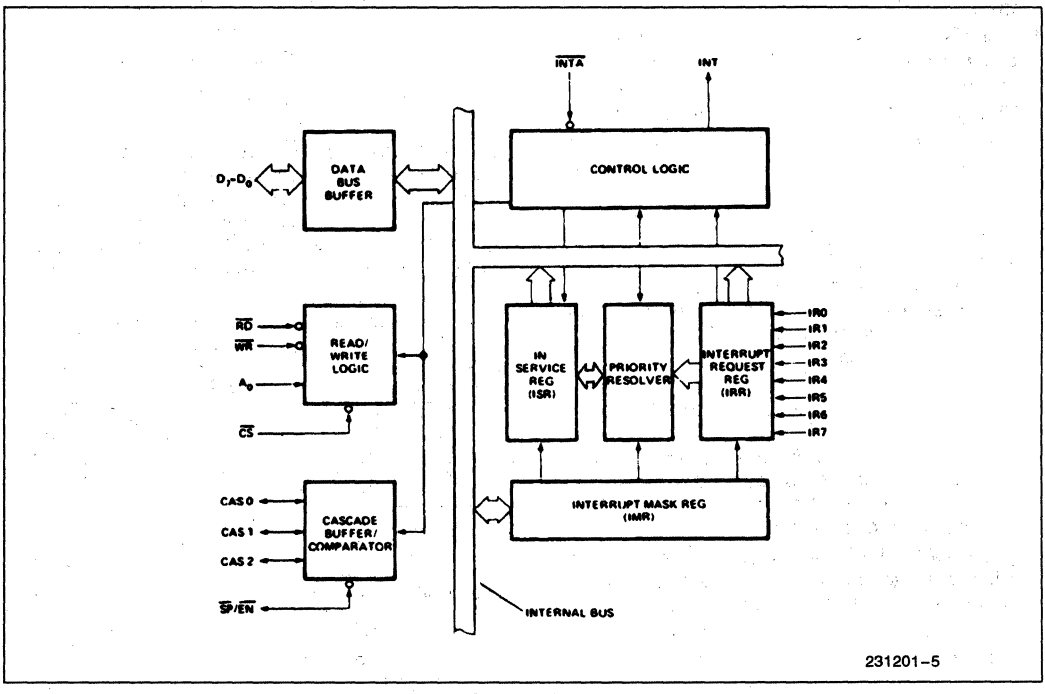


Figure 4. 82C59A-2 Block Diagram

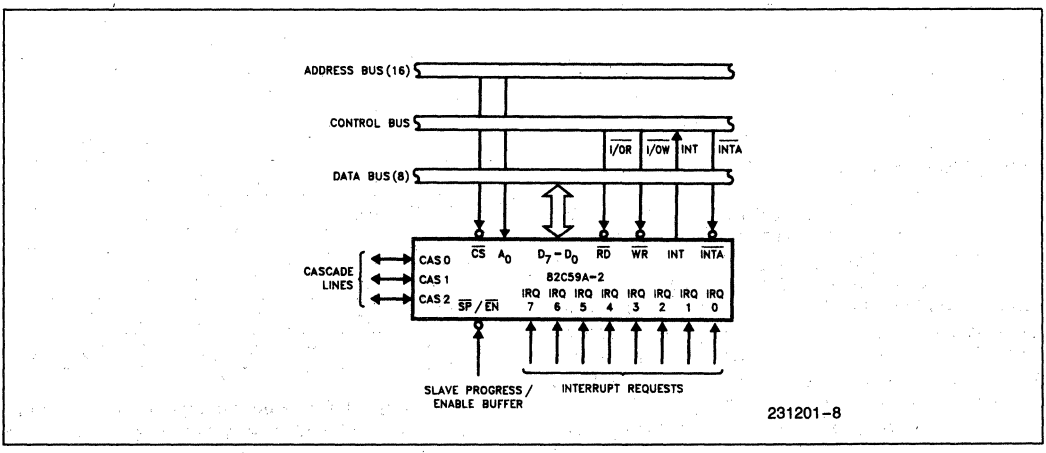


Figure 5. 82C59A-2 Interface to Standard System Bus

2. The 82C59A-2 evaluates these requests, and sends an INT to the CPU, if appropriate.
3. The CPU acknowledges the INT and responds with an $\overline{INTA}$ pulse.
4. Upon receiving an $\overline{INTA}$ from the CPU group, the highest priority ISR bit is set, and the corresponding IRR bit is reset. The 82C59A-2 will also release a CALL instruction code (11001101) onto the 8-bit Data Bus through its D7–0 pins.
5. This CALL instruction will initiate two more $\overline{INTA}$ pulses to be sent to the 82C59A-2 from the CPU group.
6. These two $\overline{INTA}$ pulses allow the 82C59A-2 to release its preprogrammed subroutine address onto the Data Bus. The lower 8-bit address is released at the first $\overline{INTA}$ pulse and the higher 8-bit address is released at the second $\overline{INTA}$ pulse.
7. This completes the 3-byte CALL instruction released by the 82C59A-2. In the AEOI mode the ISR bit is reset at the end of the third $\overline{INTA}$ pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt sequence.

The events occurring in an 80C86 system are the same until step 4.

4. Upon receiving an $\overline{INTA}$ from the CPU group, the highest priority ISR bit is set and the corresponding IRR bit is reset. The 82C59A-2 does not drive the Data Bus during this cycle.
5. The 80C86 will initiate a second $\overline{INTA}$ pulse. During this pulse, the 82C59A-2 releases an 8-bit pointer onto the Data Bus where it is read by the CPU.
6. This completes the interrupt cycle. In the AEOI mode the ISR bit is reset at the end of the second $\overline{INTA}$ pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt subroutine.

If no interrupt is present at step 4 of either sequence (i.e., the request was too short in duration) the 82C59A-2 will issue an interrupt level 7. Both the vectoring bytes and the CAS lines will look like an interrupt level 7 was requested.

When the 82C59A-2 PIC receives an interrupt, INT becomes active and an interrupt acknowledge cycle is started. If a higher priority interrupt occurs between the two INTA pulses, the INT line goes inactive immediately after the second INTA pulse. After an unspecified amount of time the INT line is activated again to signify the higher priority interrupt waiting for service. This inactive time is not specified and can vary between parts. The designer should be aware of this consideration when designing a system which uses the 82C59A-2. It is recommended that proper asynchronous design techniques be followed.

INTERRUPT SEQUENCE OUTPUTS

MCS®-80, MCS-85

This sequence is timed by three $\overline{INTA}$ pulses. During the first $\overline{INTA}$ pulse the CALL opcode is enabled onto the data bus.

Content of First Interrupt Vector Byte

	D7	D6	D5	D4	D3	D2	D1	D0
CALL CODE	1	1	0	0	1	1	0	1

During the second $\overline{INTA}$ pulse the lower address of the appropriate service routine is enabled onto the data bus. When Interval = 4 bits A_5–A_7 are programmed, while A_0–A_4 are automatically inserted by the 82C59A-2. When Interval = 8 only A_6 and A_7 are programmed, while A_0–A_5 are automatically inserted.

Content of Second Interrupt Vector Byte

IR	Interval = 4							
	D7	D6	D5	D4	D3	D2	D1	D0
7	A7	A6	A5	1	1	1	0	0
6	A7	A6	A5	1	1	0	0	0
5	A7	A6	A5	1	0	1	0	0
4	A7	A6	A5	1	0	0	0	0
3	A7	A6	A5	0	1	1	0	0
2	A7	A6	A5	0	1	0	0	0
1	A7	A6	A5	0	0	1	0	0
0	A7	A6	A5	0	0	0	0	0

IR	Interval = 8							
	D7	D6	D5	D4	D3	D2	D1	D0
7	A7	A6	1	1	1	0	0	0
6	A7	A6	1	1	0	0	0	0
5	A7	A6	1	0	1	0	0	0
4	A7	A6	1	0	0	0	0	0
3	A7	A6	0	1	1	0	0	0
2	A7	A6	0	1	0	0	0	0
1	A7	A6	0	0	1	0	0	0
0	A7	A6	0	0	0	0	0	0

During the third INTA pulse the higher address of the appropriate service routine, which was programmed as byte 2 of the initialization sequence (A_8 – A_{15}), is enabled onto the bus.

Content of Third Interrupt Vector Byte

D7	D6	D5	D4	D3	D2	D1	D0
A15	A14	A13	A12	A11	A10	A9	A8

80C86, 80C88

80C86, 80C88 mode is similar to MCS-80 mode except that only two Interrupt Acknowledge cycles are issued by the processor and no CALL opcode is sent to the processor. The first interrupt acknowledge cycle is similar to that of MCS-80, 85 systems in that the 82C59A-2 uses it to internally freeze the state of the interrupts for priority resolution and as a master it issues the interrupt code on the cascade lines at the end of the INTA pulse. On this first cycle it does not issue any data to the processor and leaves its data bus buffers disabled. On the second interrupt acknowledge cycle in 80C86, 80C88 mode the master (or slave if so programmed) will send a byte of data to the processor with the acknowledged interrupt code composed as follows (note the state of the ADI mode control is ignored and A_5–A_{11} are unused in 80C86, 80C88 mode):

Content of Interrupt Vector Byte for 80C86, 80C88 System Mode

	D7	D6	D5	D4	D3	D2	D1	D0
IR7	T7	T6	T5	T4	T3	1	1	1
IR6	T7	T6	T5	T4	T3	1	1	0
IR5	T7	T6	T5	T4	T3	1	0	1
IR4	T7	T6	T5	T4	T3	1	0	0
IR3	T7	T6	T5	T4	T3	0	1	1
IR2	T7	T6	T5	T4	T3	0	1	0
IR1	T7	T6	T5	T4	T3	0	0	1
IR0	T7	T6	T5	T4	T3	0	0	0

PROGRAMMING THE 82C59A-2

The 82C59A-2 accepts two types of command words generated by the CPU:

1. *Initialization Command Words (ICWs)*: Before normal operation can begin, each 82C59A-2 in the system must be brought to a starting point — by a sequence of 2 to 4 bytes timed by $\overline{WR}$ pulses.
2. *Operation Command Words (OCWs)*: These are the command words which command the 82C59A-2 to operate in various interrupt modes. These modes are:
   a. Fully nested mode
   b. Rotating priority mode
   c. Special mask mode
   d. Polled mode

The OCWs can be written into the 82C59A-2 anytime after initialization.

INITIALIZATION COMMAND WORDS (ICWS)

GENERAL

Whenever a command is issued with A0 = 0 and D4 = 1, this is interpreted as Initialization Command Word 1 (ICW1). ICW1 starts the initialization sequence during which the following automatically occur.

a. The edge sense circuit is reset, which means that following initialization, an interrupt request (IR) input must make a low-to-high transition to generate an interrupt.
b. The Interrupt Mask Register is cleared.
c. IR7 input is assigned priority 7.
d. The slave mode address is set to 7.
e. Special Mask Mode is cleared and Status Read is set to IRR.
f. If IC4 = 0, then all functions selected in ICW4 are set to zero. (Non-Buffered mode*, no Auto-EOI, MCS-80, 85 system).

***NOTE:**
Master/Slave in ICW4 is only used in the buffered mode.

INITIALIZATION COMMAND WORDS 1 AND 2 (ICW1, ICW2)

A_5–A_{15}: *Page starting address of service routines.* In an MCS 80/85 system, the 8 request levels will generate CALLs to 8 locations equally spaced in memory. These can be programmed to be spaced at intervals of 4 or 8 memory locations, thus the 8 routines will occupy a page of 32 or 64 bytes, respectively.

The address format is 2 bytes long (A_0–A_{15}). When the routine interval is 4, A_0–A_4 are automatically inserted by the 82C59A-2, while A_5–A_{15} are programmed externally. When the routine interval is 8, A_0–A_5 are automatically inserted by the 82C59A-2, while A_6–A_{15} are programmed externally.

The 8-byte interval will maintain compatibility with current software, while the 4-byte interval is best for a compact jump table.

In an 80C86, 80C88 system A_{15}–A_{11} are inserted in the five most significant bits of the vectoring

byte and the 82C59A-2 sets the three least significant bits according to the interrupt level. $A_{10}-A_5$ are ignored and ADI (Address Interval) has no effect:

LTIM: If LTIM = 1, then the 82C59A-2 will operate in the level interrupt mode. Edge detect logic on the interrupt inputs will be disabled.

ADI: CALL address interval. ADI = 1 then interval = 4; ADI = 0 then interval = 8.

SNGL: Single. Means that this is the only 82C59A-2 in the system. If SNGL = 1 no ICW3 will be issued.

IC4: If this bit is set — ICW4 has to be read. If ICW4 is not needed, set IC4 = 0.

INITIALIZATION COMMAND WORD 3 (ICW3)

This word is read only when there is more than one 82C59A-2 in the system and cascading is used, in which case SNGL = 0. It will load the 8-bit slave register. The functions of this register are:

a. In the master mode (either when SP = 1, or in buffered mode when M/S = 1 in ICW4) a "1" is set for each slave in the system. The master then will release byte 1 of the call sequence (for MCS-80/85 system) and will enable the corresponding slave to release bytes 2 and 3 (for 80C86, 80C88 only byte 2) through the cascade lines.

b. In the slave mode (either when $\overline{SP}$ = 0, or if BUF = 1 and M/S = 0 in ICW4) bits 2-0 identify the slave. The slave compares its cascade input with these bits and, if they are equal, bytes 2 and 3 of the call sequence (or just byte 2 for 80C86, 80C88 are released by it on the Data Bus.

INITIALIZATION COMMAND WORD 4 (ICW4)

SFNM: If SFNM = 1 the special fully nested mode is programmed.

BUF: If BUF = 1 the buffered mode is programmed. In buffered mode $\overline{SP}/\overline{EN}$ becomes an enable output and the master/slave determination is by M/S.

M/S: If buffered mode is selected: M/S = 1 means the 82C59A-2 is programmed to be a master, M/S = 0 means the 82C59A-2 is programmed to be a slave. If BUF = 0, M/S has no function.

AEOI: If AEOI = 1 the automatic end of interrupt mode is programmed.

μPM: Microprocessor mode: μPM = 0 sets the 82C59A-2 for MCS-80, 85 system operation, μPM = 1 sets the 82C59A-2 for 80C86 system operation.

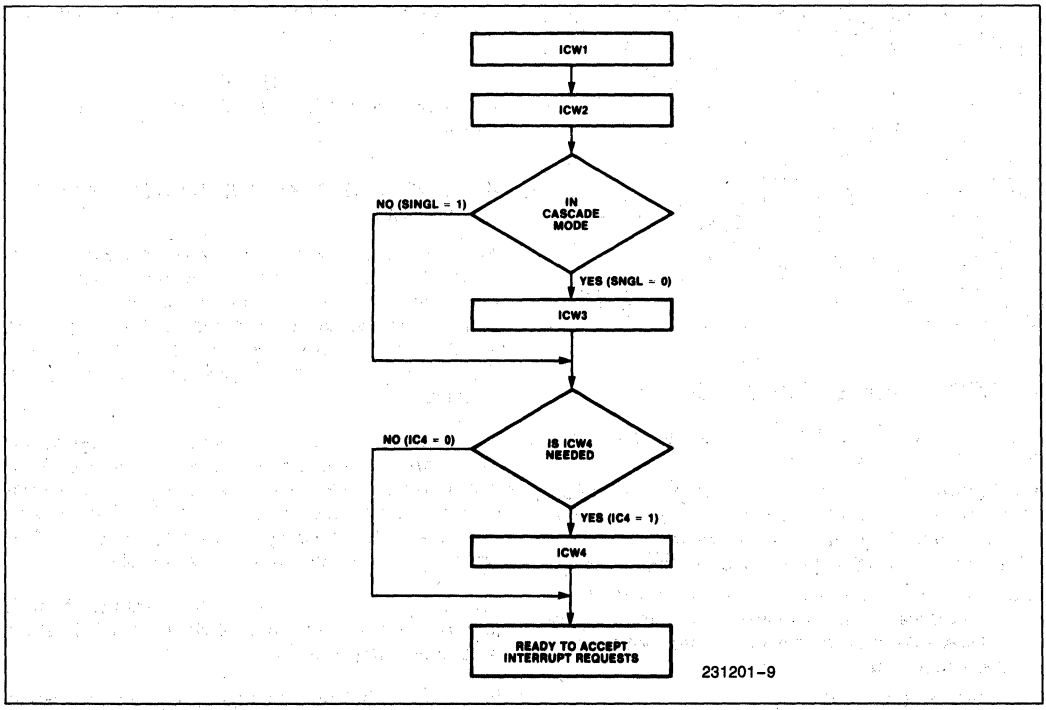
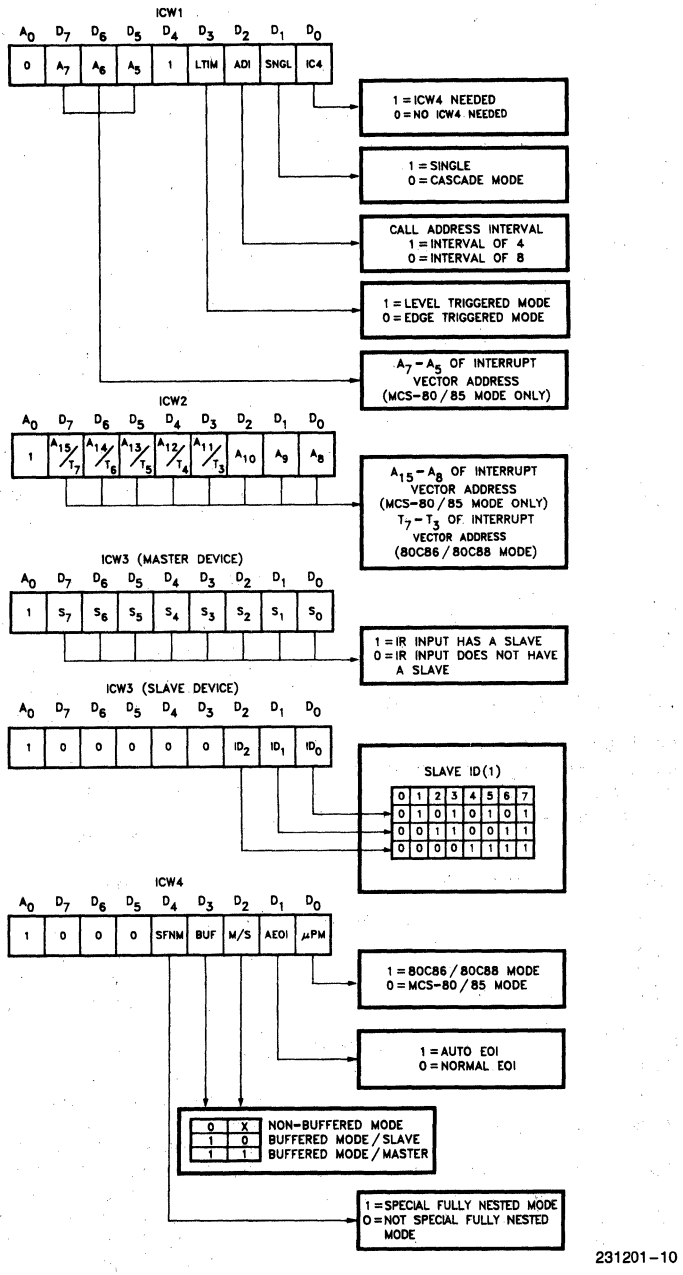


Figure 6. Initialization Sequence



NOTE:
Slave ID is equal to the corresponding master IR input.

Figure 7. Initialization Command Word Format

OPERATION COMMAND WORDS (OCWs)

After the initialization Command Words (ICWs) are programmed into the 82C59A-2, the chip is ready to accept interrupt requests at its input lines. However, during the 82C59A-2 operation, a selection of algorithms can command the 82C59A-2 to operate in various modes through the Operation Command Words (OCWs).

OPERATION CONTROL WORDS (OCWs)

OCW1

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7	M6	M5	M4	M3	M2	M1	M0

OCW2

0	R	SL	EOI	0	0	L2	L1	L0

OCW3

0	0	ESMM	SMM	0	1	P	RR	RIS

OPERATION CONTROL WORD 1 (OCW1)

OCW1 sets and clears the mask bits in the interrupt Mask Register (IMR). $M_7 - M_0$ represent the eight mask bits. M = 1 indicates the channel is masked (inhibited), M = 0 indicates the channel is enabled.

OPERATION CONTROL WORD 2 (OCW2)

R, SL, EOI — These three bits control the Rotate and End of Interrupt modes and combinations of the two. A chart of these combinations can be found on the Operation Command Word Format.

L_2, L_1, L_0—These bits determine the interrupt level acted upon when the SL bit is active.

OPERATION CONTROL WORD 3 (OCW3)

ESMM — Enable Special Mask Mode. When this bit is set to 1 it enables the SMM bit to set or reset the Special Mask Mode. When ESMM = 0 the SMM bit becomes a "don't care".

SMM — Special Mask Mode. If ESMM = 1 and SMM = 1 the 82C59A-2 will enter Special Mask Mode. If ESMM = 1 and SMM = 0 the 82C59A-2 will revert to normal mask mode. When ESMM = 0, SMM has no effect.

FULLY NESTED MODE

This mode is entered after initialization unless another mode is programmed. The interrupt requests are ordered in priority form 0 through 7 (0 highest). When an interrupt is acknowledged the highest priority request is determined and its vector placed on the bus. Additionally, a bit of the Interrupt Service register (ISO-7) is set. This bit remains set until the microprocessor issues an End of Interrupt (EOI) command immediately before returning from the service routine, or if AEOI (Automatic. End of Interrupt) bit is set, until the trailing edge of the last INTA. While the IS bit is set, all further interrupts of the same or lower priority are inhibited, while higher levels will generate an interrupt (which will be acknowledged only if the microprocessor internal interrupt enable flip-flop has been re-enabled through software).

After the initialization sequence, IR0 has the highest priority and IR7 the lowest. Priorities can be changed, as will be explained, in the rotating priority mode.

END OF INTERRUPT (EOI)

The In Service (IS) bit can be reset either automatically following the trailing edge of the last in sequence $\overline{\text{INTA}}$ pulse (when AEOI bit in ICW4 is set) or by a command word that must be issued to the 82C59A-2 before returning from a service routine (EOI command). An EOI command must be issued twice if in the Cascade mode, once for the master and once for the corresponding slave.

There are two forms of EOI command: Specific and Non-Specific. When the 82C59A-2 is operated in modes which preserve the fully nested structure, it can determine which IS bit to reset on EOI. When a Non-Specific EOI command is issued the 82C59A-2 will automatically reset the highest IS bit of those that are set, since in the fully nested mode the highest IS level was necessarily the last level acknowledged and serviced. A non-specific EOI can be issued with OCW2 (EOI = 1, SL = 0, R = 0).

When a mode is used which may disturb the fully nested structure, the 82C59A-2 may no longer be able to determine the last level acknowledged. In this case a Specific End of Interrupt must be issued which includes as part of the command the IS level to be reset. A specific EOI can be issued with OCW2 (EOI = 1, SL = 1, R = 0, and L0-L2 is the binary level of the IS bit to be reset).

It should be noted that an IS bit that is masked by an IMR bit will not be cleared by a non-specific EOI if the 82C59A-2 is in the Special Mask Mode.

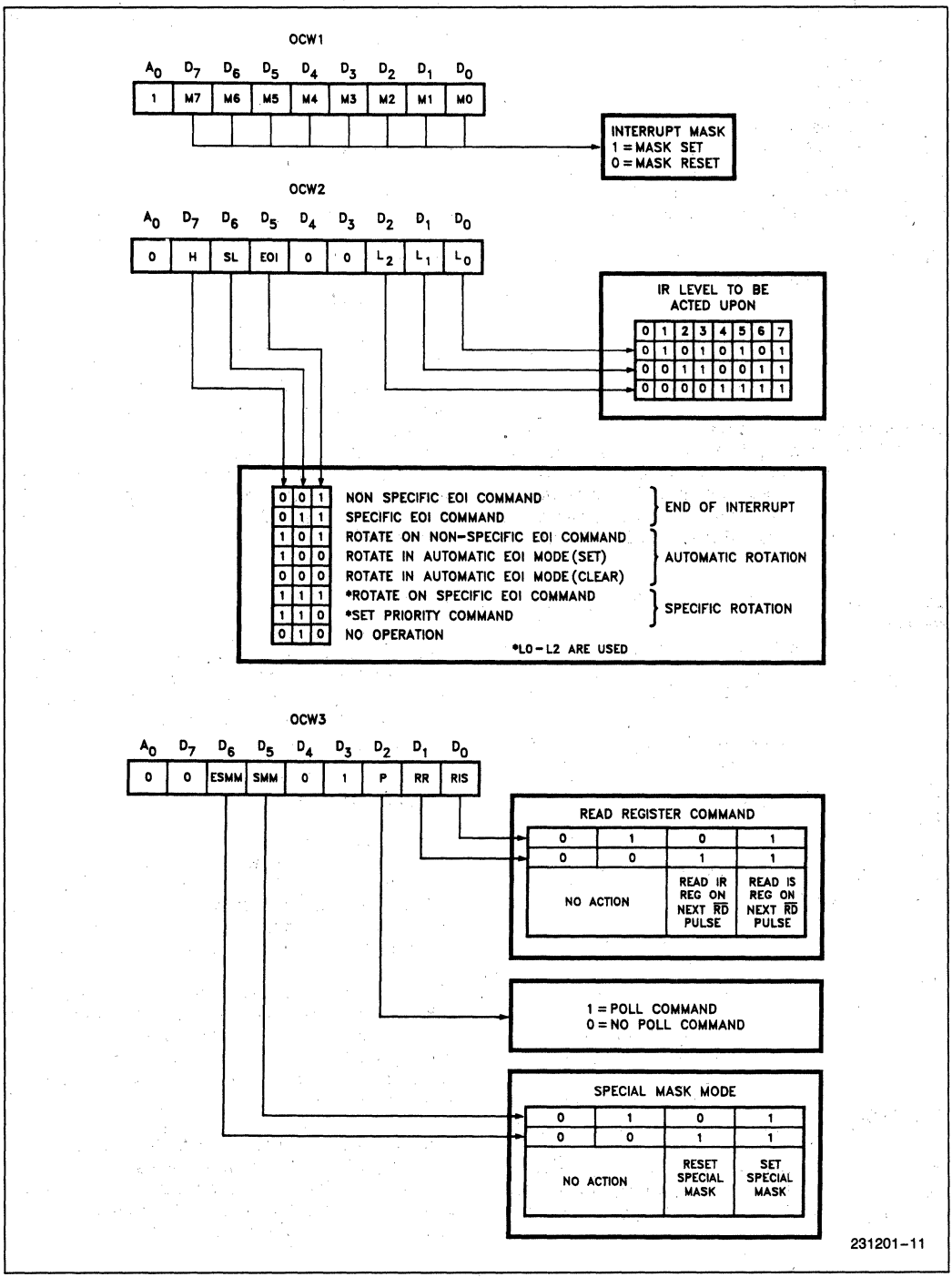


Figure 8. Operation Command Word Format

AUTOMATIC END OF INTERRUPT (AEOI) MODE

If AEOI = 1 in ICW4, then the 82C59A-2 will operate in AEOI mode continuously until reprogrammed by ICW4. In this mode the 82C59A-2 will automatically perform a non-specific EOI operation at the trailing edge of the last interrupt acknowledge pulse (third pulse in MCS-80/85, second in 80C86/88). Note that from a system standpoint, this mode should be used only when a nested multilevel interrupt structure is not required within a single 82C59A.

The AEOI mode can only be used in a master 82C59A and not a slave.

AUTOMATIC ROTATION

(Equal Priority Devices)

In some applications there are a number of interrupting devices of equal priority. In this mode a device, after being serviced, receives the lowest priority, so a device requesting an interrupt will have to wait, in the worst case until each of 7 other devices are serviced at most *once*. For example, if the priority and "in service" status is:

Before Rotate (IR4 the highest priority requiring service)

	IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
"IS" Status	0	1	0	1	0	0	0	0

	Lowest Priority ↓							Highest Priority ↓
Priority Status	7	6	5	4	3	2	1	0

After Rotate (IR4 was serviced, all other priorities rotated correspondingly)

	IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
"IS" Status	0	1	0	0	0	0	0	0

	Highest Priority ↓			Lowest Priority ↓				
Priority Status	2	1	0	7	6	5	4	3

There are two ways to accomplish Automatic Rotation using OCW2, the Rotation on Non-Specific EOI Command (R = 1, SL = 0, EOI = 1) and the Rotate in Automatic EOI Mode which is set by (R = 1, SL = 0, EOI = 0) and cleared by (R = 0, SL = 0, EOI = 0).

SPECIFIC ROTATION

(Specific Priority)

The programmer can change priorities by programming the bottom priority and thus fixing all other priorities; i.e., if IR5 is programmed as the bottom priority device, then IR6 will have the highest one.

The Set Priority command is issued in OCW2 where: R = 1, SL = 1; LO–L2 is the binary priority level code of the bottom priority device.

Observe that in this mode internal status is updated by software control during OCW2. However, it is independent of the End of Interrupt (EOI) command (also executed by OCW2). Priority changes can be executed during an EOI command by using the Rotate on Specific EOI command in OCW2 (R = 1, SL = 1, EOI = 1 and LO–L2 = IR level to receive bottom priority).

INTERRUPT MASKS

Each Interrupt Request input can be masked individually by the Interrupt Mask Register (IMR) programmed through OCW1. Each bit in the IMR masks one interrupt channel if it is set (1). Bit 0 masks IR0, Bit 1 masks IR1 and so forth. Masking an IR channel does not affect the other channels operation.

SPECIAL MASK MODE

Some applications may require an interrupt service routine to dynamically alter the system priority structure during its execution under software control. For example, the routine may wish to inhibit lower priority requests for a portion of its execution but enable some of them for another portion.

The difficulty here is that if an interrupt Request is acknowledged and an End of Interrupt command did not reset its IS bit (i.e., while executing a service routine), the 82C59A-2 would have inhibited all lower priority requests with no easy way for the routine to enable them.

That is where the Special Mask Mode comes in. In the special Mask Mode, when a mask bit is set in OCW1, it inhibits further interrupts at that level *and enables* interrupts from *all other* levels (lower as well as higher) that are not masked.

Thus, any interrupts may be selectivity enabled by loading the mask register.

The special Mask Mode is set by OCW3 where: SSMM = 1, SMM = 1, and cleared where SSMM = 1, SMM = 0.

POLL COMMAND

In Poll mode the INT output functions as it normally does. The microprocessor should ignore this output. This can be accomplished either by not connecting the INT output or by masking interrupts within the microprocessor, thereby disabling its interrupt input. Service to devices is achieved by software using a Poll command.

The Poll command is issued by setting P = "1" in OCW3. The 82C59A-2 treats the next $\overline{RD}$ pulse to the 82C59A-2 (i.e., $\overline{RD}$ = 0, $\overline{CS}$ = 0) as an interrupt acknowledge, sets the appropriate IS bit if there is a request, and reads the priority level. Interrupt is frozen from $\overline{WR}$ to $\overline{RD}$.

The word enabled onto the data bus during $\overline{RD}$ is:

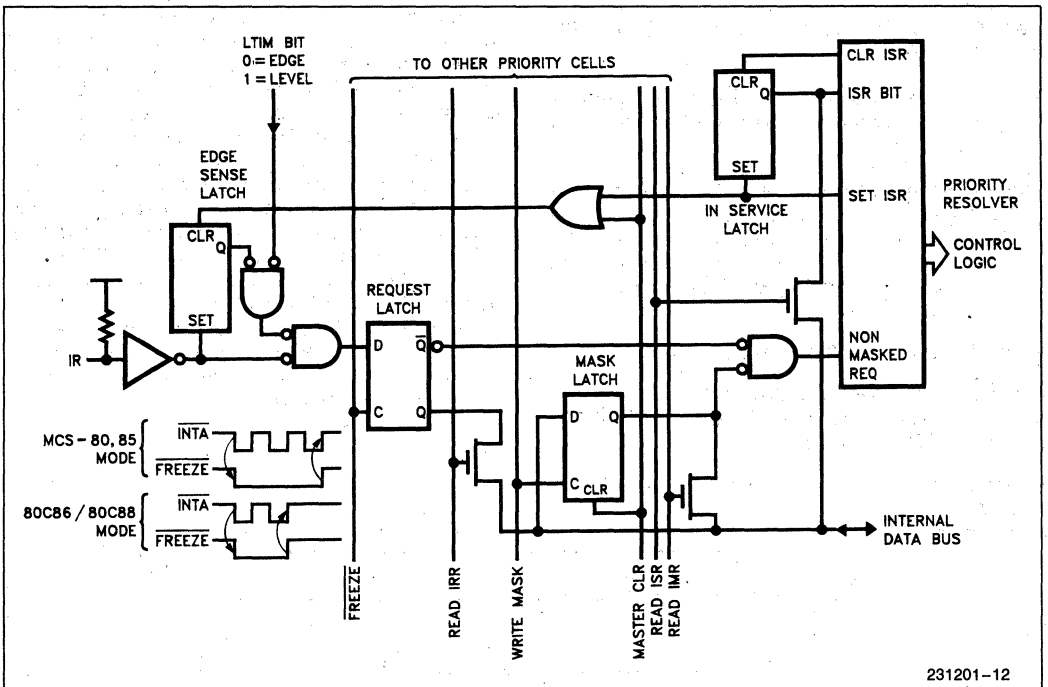
D7	D6	D5	D4	D3	D2	D1	D0
I	—	—	—	—	W2	W1	W0

WO–W2:

Binary code of the highest priority level requesting service.

I: Equal to a "1" if there is an interrupt.

This mode is useful if there is a routine command common to several levels so that the $\overline{INTA}$ sequence is not needed (saves ROM space). Another application is to use the poll mode to expand the number of priority levels to more than 64.



NOTES:
1. Master Clear active only during ICW1
2. Freeze/ is active during $\overline{INTA}$/and poll sequences only
3. Truth Table for D-Latch

C	D	Q	OPERATION
1	Di	Di	FOLLOW
0	X	Qn-1	HOLD

Figure 9. Priority Cell—Simplified Logic Diagram

READING THE 82C59A-2 STATUS

The input status of several internal registers can be read to update the user information on the system. The following registers can be read via OCW3 (IRR and ISR or OCW1 [IMR]).

Interrupt Request Register (IRR): 8-bit register which contains the levels requesting an interrupt to be acknowledged. The highest request level is reset from the IRR when an interrupt is acknowledged. (Not affected by IMR).

In-Service Register (ISR): 8-bit register which contains the priority levels that are being serviced. The ISR is updated when an End of Interrupt Command is issued.

Interrupt Mask Register: 8-bit register which contains the interrupt request lines which are masked.

The IRR can be read when, prior to the RD pulse, a Read Register Command is issued with OCW3 (RR = 1, RIS = 0.)

The ISR can be read when, prior to the RD pulse, a Read Register Command is issued with OCW3 (RR = 1, RIS = 1):

There is no need to write an OCW3 before every status read operation, as long as the status read corresponds with the previous one; i.e., the 82C59A-2 "remembers" whether the IRR or ISR has been previously selected by the OCW3. This is not true when poll is used.

After initialization the 82C59A-2 is set to IRR.

For reading the IMR, no OCW3 is needed. The output data bus will contain the IMR whenever $\overline{RD}$ is active and AO = 1 (OCW1).

Polling overrides status read when P = 1, RR = 1 in OCW3.

EDGE AND LEVEL TRIGGERED MODES

This mode is programmed using bit 3 in ICW1.

If LTIM = '0', an interrupt request will be recognized by a low to high transition on an IR input. The IR input can remain high without generating another interrupt.

If LTIM = '1', an interrupt request will be recognized by a 'high' level on IR Input, and there is no need for an edge detection. The interrupt request must be removed before the EOI command is issued or the CPU interrupt is enabled to prevent a second interrupt from occurring.

The priority cell diagram shows a conceptual circuit of the level sensitive and edge sensitive input circuitry of the 82C59A-2. Be sure to note that the request latch is a transparent D type latch.

In both the edge and level triggered modes the IR inputs must remain high until after the falling edge of the first INTA. If the IR input goes low before this time a DEFAULT IR7 will occur when the CPU acknowledges the interrupt. This can be a useful safeguard for detecting interrupts caused by spurious noise glitches on the IR inputs. To implement this feature the IR7 routine is used for "clean up" simply executing a return instruction, thus ignoring the interrupt. If IR7 is needed for other purposes a default IR7 can still be detected by reading the ISR. A normal IR7 interrupt will set the corresponding ISR bit, a default IR7 won't. If a default IR7 routine occurs during a normal IR7 routine, however, the ISR will remain set. In this case it is necessary to keep track of whether or not the IR7 routine was previously entered. If another IR7 occurs it is a default.

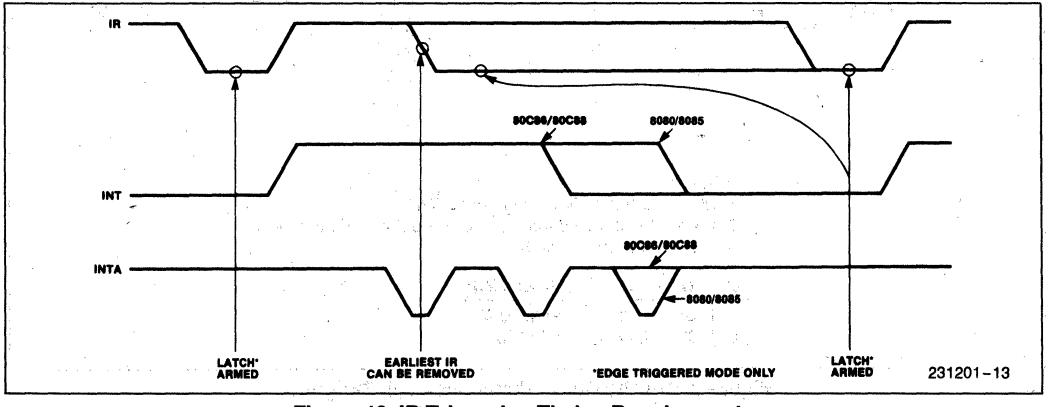


Figure 10. IR Triggering Timing Requirements

THE SPECIAL FULLY NESTED MODE

This mode will be used in the case of a big system where cascading is used, and the priority has to be conserved within each slave. In this case the fully nested mode will be programmed to the master (using ICW4). This mode is similar to the normal nested mode with the following exceptions:

a. When an interrupt request from a certain slave is in service this slave is not locked out from the master's priority logic and further interrupt requests from higher priority IR's within the slave will be recognized by the master and will initiate interrupts to the processor. (In the normal nestled mode a slave is masked out when its request is in service and no higher requests from the same slave can be serviced.)

b. When exiting the Interrupt Service routine the software has to check whether the interrupt serviced was the only one from that slave. This is done by sending a non-specific End of Interrupt (EOI) command to the slave and then reading its In-Service register and checking for zero. If it is empty, a non-specific EOI can be sent to the master too. If not, no EOI should be sent.

BUFFERED MODE

When the 82C59A-2 is used in a large system where bus driving buffers are required on the data bus and the cascading mode is used, there exists the problem of enabling buffers.

The buffered mode will structure the 82C59A-2 to send an enable signal on $\overline{SP}/\overline{EN}$ to enable the buffers. In this mode, whenever the 82C59A-2's data bus outputs are enabled, the $\overline{SP}/\overline{EN}$ output becomes active.

This modification forces the use of software programming to determine whether the 82C59A-2 is a master or a slave. Bit 3 in ICW4 programs the buffered mode, and bit 2 in ICW3 determines whether it is a master or a slave.

CASCADE MODE

The 82C59A-2 can be easily interconnected in a system of one master with up to eight slaves to handle up to 64 priority levels.

The master controls the slaves through the 3 line cascade bus. The cascade bus acts like chip selects to the slaves during the $\overline{INTA}$ sequence.

In a cascade configuration, the slave interrupt outputs are connected to the master interrupt request inputs. When a slave request line is activated and afterwards acknowledged, the master will enable the corresponding slave to release the device routine address during bytes 2 and 3 of INTA. (Byte 2 only for 80C86/80C88).

The cascade bus lines are normally low and will contain the slave address code from the trailing edge of the first INTA pulse to the trailing edge of the third pulse. Each 82C59A-2 in the system must follow a separate initialization sequence and can be programmed to work in a different mode. An EOI command must be issued twice: once for the master and once for the corresponding slave. An address decoder is required to activate the Chip Select (CS) input of each 82C59A-2.

The cascade lines of the Master 82C59A-2 are activated only for slave inputs, non slave inputs leave the cascade line inactive (low).

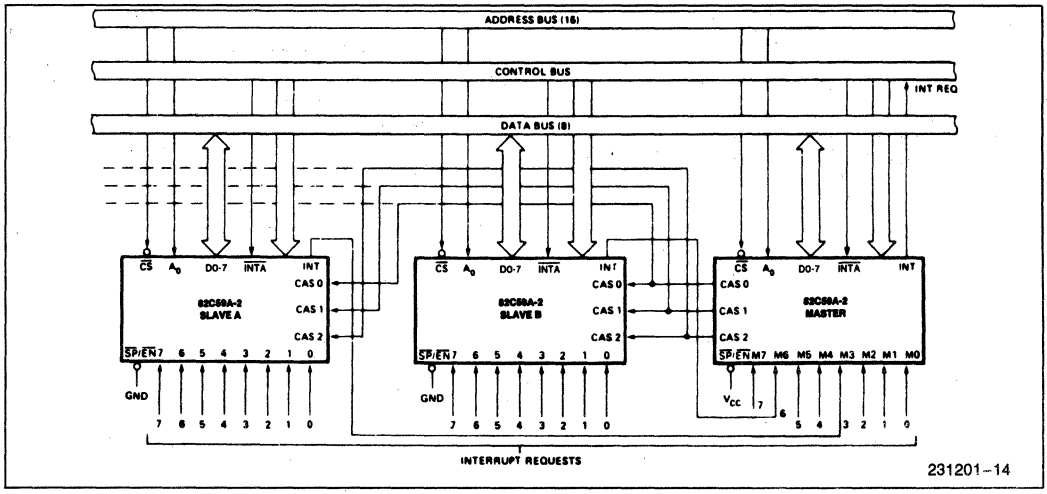


Figure 11. Cascading the 82C59A-2

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature .....−65°C to + 150°C
Supply Voltage (w.r.t. ground) .....−0.5 to 7.0V
Input Voltage (w.r.t. ground) ...−0.5 to V_{CC} + 0.5V
Output Voltage (w.r.t. ground)..−0.5 to V_{CC} + 0.5V
Power Dissipation.....0.9 Watt

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Min	Max	Units	Test Conditions
I_{CCS}	Standby Supply Current		10	μA	$V_{IN} = V_{CC}$ or GND All IR = GND Outputs Unloaded V_{CC} = 5.5V
I_{CC}	Operating Supply Current		5	mA	(Note)
V_{IH}	Input High Voltage	2.2	V_{CC} + 0.5	V	
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2.5 mA
V_{OH}	Output High Voltage	3.0 V_{CC} −0.4		V	I_{OH} = −2.5 mA I_{OH} = −100 μA
I_{LI}	Input Leakage Current		±1.0	μA	$0V \le V_{IN} \le V_{CC}$
I_{LO}	Output Leakage Current		±10	μA	$0V \le V_{OUT} \le V_{CC}$
I_{LIR}	IR Input Leakage Current		−300 +10	μA	V_{IN} = 0 $V_{IN} = V_{CC}$

NOTE:
Repeated data input with 80C86-2 timings.

CAPACITANCE T_A = 25°C; V_{CC} = GND = 0V

Symbol	Parameter	Min	Max	Units	Test Conditions	
C_{IN}	Input Capacitance		7	pF		fc = 1 MHz
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins at GND	
C_{OUT}	Output Capacitance		15	pF		

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±10%

TIMING REQUIREMENTS

Symbol	Parameter	82C59A-2		Units	Test Conditions
		Min	Max		
TAHRL	A0/$\overline{CS}$ Setup to $\overline{RD}$/$\overline{INTA}$ ↓	10		ns	
TRHAX	A0/$\overline{CS}$ Hold after $\overline{RD}$/$\overline{INTA}$ ↑	5		ns	
TRLRH	$\overline{RD}$/$\overline{INTA}$ Pulse Width	160		ns	
TAHWL	A0/$\overline{CS}$ Setup to $\overline{WR}$ ↓	0		ns	
TWHAX	A0/$\overline{CS}$ Hold after $\overline{WR}$ ↑	0		ns	
TWLWH	$\overline{WR}$ Pulse Width	190		ns	
TDVWH	Data Setup to $\overline{WR}$ ↑	160		ns	
TWHDX	Data Hold after $\overline{WR}$ ↑	0		ns	
TJLJH	Interrupt Request Width (Low)	100		ns	(See Note)
TCVIAL	Cascade Setup to Second or Third $\overline{INTA}$ ↓ (Slave Only)	40		ns	
TRHRL	End of $\overline{RD}$ to next $\overline{RD}$ End of $\overline{INTA}$ to next $\overline{INTA}$ within an $\overline{INTA}$ sequence only	160		ns	
TWHWL	End of $\overline{WR}$ to next $\overline{WR}$	190		ns	
*TCHCL	End of Command to next Command (Not same command type) End of $\overline{INTA}$ sequence to next $\overline{INTA}$ sequence.	400		ns	

*Worst case timing for TCHCL in an actual microprocessor system is typically much greater than 400 ns (i.e. 8085A = 1.6 μs, 8085-A2 = 1 μs, 80C86 = 1 μs, 80C86-2 = 625 ns)

NOTE:
This is the low time required to clear the input latch in the edge triggered mode.

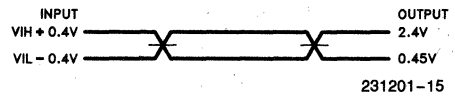
TIMING RESPONSES

Symbol	Parameter	8259A-2		Units	Test Conditions**
		Min	Max		
TRLDV	Data Valid from $\overline{RD}/\overline{INTA}\downarrow$		120	ns	1
TRHDZ	Data Float after $\overline{RD}/\overline{INTA}\uparrow$	10	85	ns	2
TJHIH	Interrupt Output Delay		300	ns	1
TIALCV	Cascade Valid from First $\overline{INTA}\downarrow$ (Master Only)		360	ns	1
TRLEL	Enable Active from $\overline{RD}\downarrow$ or $\overline{INTA}\downarrow$		110	ns	1
TRHEH	Enable Inactive from $\overline{RD}\uparrow$ or $\overline{INTA}\uparrow$		150	ns	1
TAHDV	Data Valid from Stable Address		200	ns	1
TCVDV	Cascade Valid to Valid Data		200	ns	1

****Test Condition Definition Table**

TEST CONDITION	V1	R1	R2	C1
1	1.7V	523Ω	OPEN	100 pf
2	4.5V	1.8 kΩ	1.8 kΩ	30 pf

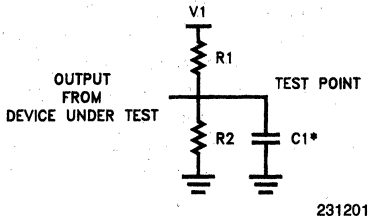
A.C. TESTING INPUT, OUTPUT WAVEFORM



231201-15

A.C. Testing: All input signals must switch between V_{IL} – 0.4V and V_{IH} + 0.4V. Input Rise and Fall Times must be ≤ 15 ns. All timing measurements are made at 2.4V and 0.45V.

A.C. TESTING LOAD CIRCUIT

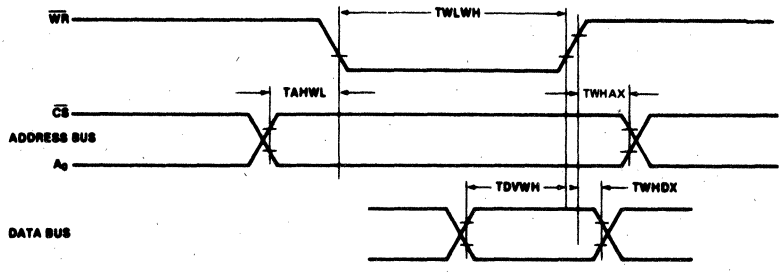


231201-16

*Includes Stray and Jig Capacitance

WAVEFORMS

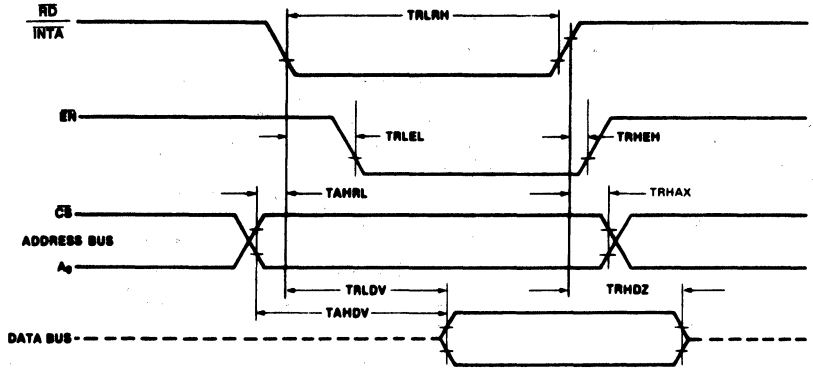
WRITE



231201-17

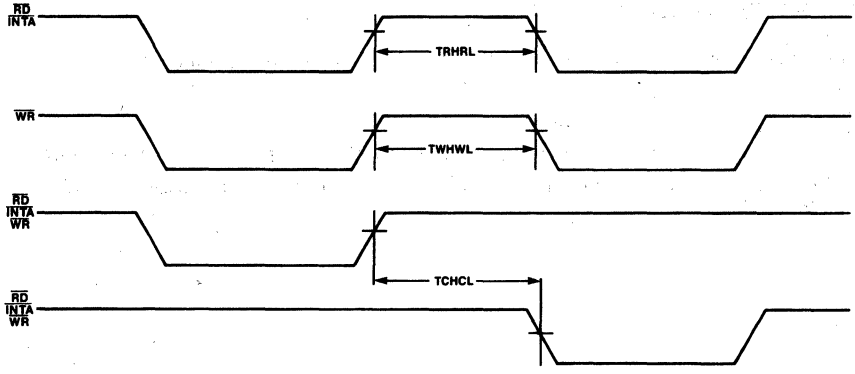
WAVEFORMS (Continued)

READ/INTA



231201-18

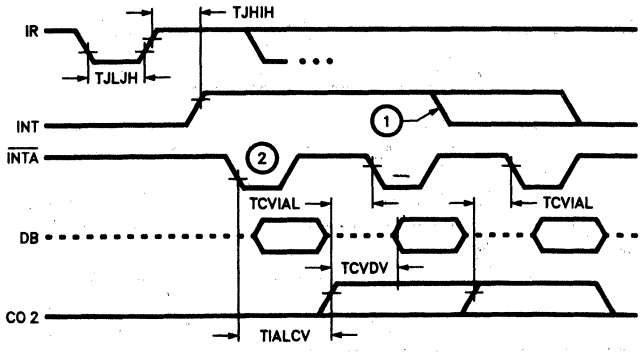
OTHER TIMING



231201-19

WAVEFORMS (Continued)

$\overline{\text{INTA}}$ SEQUENCE



231201-20

NOTES:
1. Interrupt output must remain HIGH at least until leading edge of first INTA.
2. Cycle 1 in 80C86 and 80C88 systems, the Data Bus is not active.

DATA SHEET REVISION REVIEW

The following changes have been made since revision 003 of the 82C59A-2 data sheet.

1. Preliminary was removed.
2. A reference to PLCC packaging was removed.
3. The first paragraph of the Poll Command section was rewritten to clarify the status of the INT pin.
4. A paragraph was added to the Interrupt Sequence section to indicate the status of the INT pin during multiple interrupts.

80286 Microprocessor Family

3

80286
High Performance Microprocessor with Memory Management and Protection
(80286-12, 80286-10, 80286-8)

- **High Performance Processor (Up to six times 8086)**
- **Large Address Space:**
  - **— 16 Megabytes Physical**
  - **— 1 Gigabyte Virtual per Task**
- **Integrated Memory Management, Four-Level Memory Protection and Support for Virtual Memory and Operating Systems**
- **High Bandwidth Bus Interface (12.5 Megabyte/Sec)**
- **Industry Standard O.S. Support:**
  - **— iRMX®**
  - **— XENIX***
  - **— UNIX***
  - **— MS-DOS***
- **Optional Processor Extension:**
  - **— 80287 High Performance 80-bit Numeric Data Processor**
- **Two 8086 Upward Compatible Operating Modes:**
  - **— 8086 Real Address Mode**
  - **— Protected Virtual Address Mode**
- **Range of Clock Rates**
  - **— 12.5 MHz for 80286-12**
  - **— 10 MHz for 80286-10**
  - **— 8 MHz for 80826-8**
  - **— 6 MHz for 80286-6**
- **Complete System Development Support:**
  - **— Development Software: Assembler, PL/M, Pascal, FORTRAN, and System Utilities**
  - **— In-Circuit-Emulator (ICE™-286)**
- **Available in 68 Pin Ceramic LCC (Leadless Chip Carrier), PGA (Pin Grid Array), and PLCC (Plastic Leaded Chip Carrier) Packages**

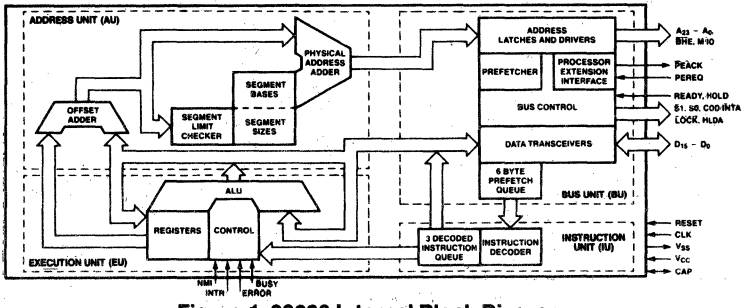
(See Packaging Spec., Order #231369)

The 80286 is an advanced, high-performance microprocessor with specially optimized capabilities for multiple user and multi-tasking systems. The 80286 has built-in memory protection that supports operating system and task isolation as well as program and data privacy within tasks. A 12 MHz 80286 provides six times or more throughput than the standard 5 MHz 8086. The 80286 includes memory management capabilities that map 2^{30} (one gigabyte) of virtual address space per task into 2^{24} bytes (16 megabytes) of physical memory.

The 80286 is upward compatible with 8086 and 88 software. Using 8086 real address mode, the 80286 is object code compatible with existing 8086, 88 software. In protected virtual address mode, the 80286 is source code compatible with 8086, 88 software and may require upgrading to use virtual addresses supported by the 80286's integrated memory management and protection mechanism. Both modes operate at full 80286 performance and execute a superset of the 8086 and 88 instructions.

The 80286 provides special operations to support the efficient implementation and execution of operating systems. For example, one instruction can end execution of one task, save its state, switch to a new task, load its state, and start execution of the new task. The 80286 also supports virtual memory systems by providing a segment-not-present exception and restartable instructions.

*XENIX and MS-DOS are trademarks of Microsoft Corp.
*UNIX is a trademark of Bell Labs or AT&T



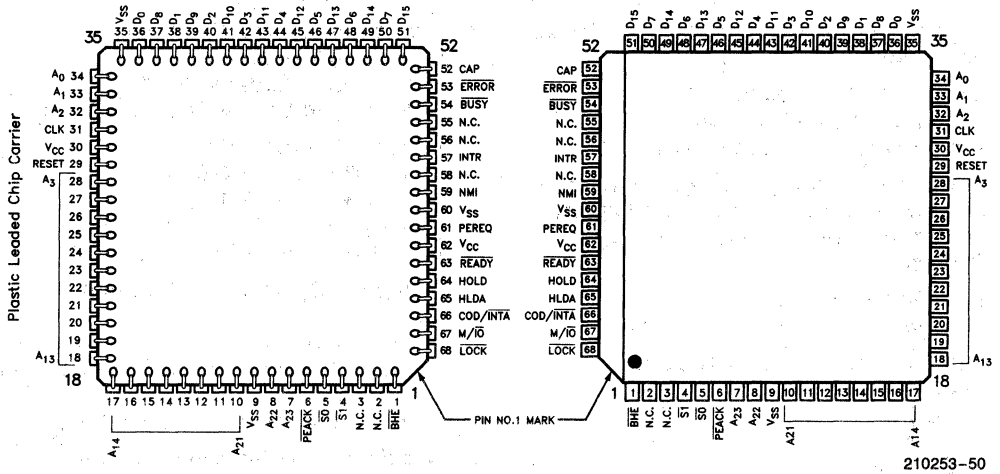
210253-1

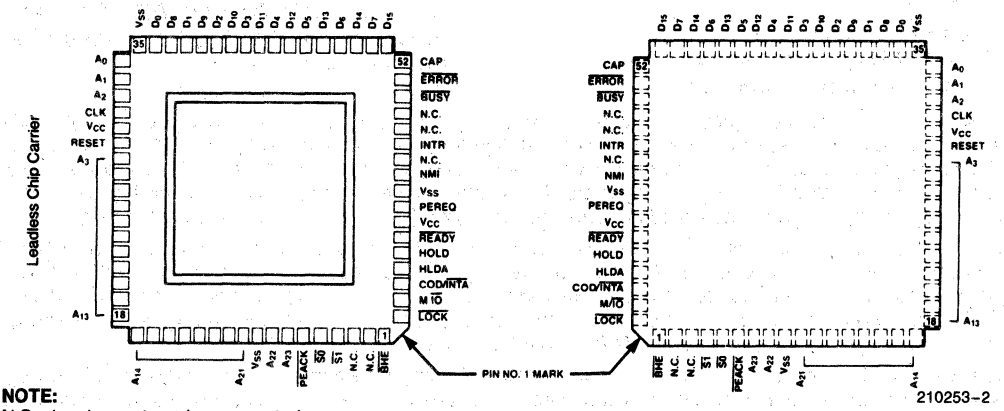
Figure 1. 80286 Internal Block Diagram

September 1988
Order Number: 210253-013

Component Pad Views—As viewed from underside of component when mounted on the board.

P.C. Board Views—As viewed from the component side of the P.C. board.





NOTE:
N.C. signals must not be connected

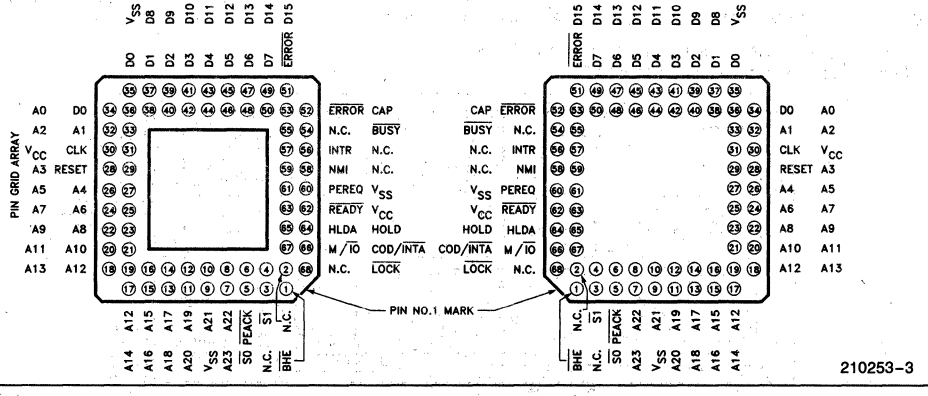


Figure 2. 80286 Pin Configuration

Table 1. Pin Description

The following pin function descriptions are for the 80286 microprocessor:

Symbol	Type	Name and Function
CLK	I	**SYSTEM CLOCK** provides the fundamental timing for 80286 systems. It is divided by two inside the 80286 to generate the processor clock. The internal divide-by-two circuitry can be synchronized to an external clock generator by a LOW to HIGH transition on the RESET input.
$D_{15}-D_0$	I/O	**DATA BUS** inputs data during memory, I/O, and interrupt acknowledge read cycles; outputs data during memory and I/O write cycles. The data bus is active HIGH and floats to 3-state OFF during bus hold acknowledge.
$A_{23}-A_0$	O	**ADDRESS BUS** outputs physical memory and I/O port addresses. A0 is LOW when data is to be transferred on pins D_{7-0}. $A_{23}-A_{16}$ are LOW during I/O transfers. The address bus is active HIGH and floats to 3-state OFF during bus hold acknowledge.
$\overline{BHE}$	O	**BUS HIGH ENABLE** indicates transfer or data on the upper byte of the data bus. D_{15-8}. Eight-bit oriented devices assigned to the upper byte of the data bus would normally use $\overline{BHE}$ to condition chip select functions. $\overline{BHE}$ is active LOW and floats to 3-state OFF during bus hold acknowledge.

$\overline{BHE}$ and A0 Encodings

$\overline{BHE}$ Value	A0 Value	Function
0	0	Word transfer
0	1	Byte transfer on upper half of data bus ($D_{15}-D_8$)
1	0	Byte transfer on lower half of data bus (D_{7-0})
1	1	Will never occur

$\overline{S1}$, $\overline{S0}$	O	**BUS CYCLE STATUS** indicates initiation of a bus cycle and, along with M/$\overline{IO}$ and COD/$\overline{INTA}$, defines the type of bus cycle. The bus is in a T_s state whenever one or both are LOW, $\overline{S1}$ and $\overline{S0}$ are active LOW and float to 3-state OFF during bus hold acknowledge.

80286 Bus Cycle Status Definition

COD/$\overline{INTA}$	M/$\overline{IO}$	$\overline{S1}$	$\overline{S0}$	Bus Cycle Initiated
0 (LOW)	0	0	0	Interrupt acknowledge
0	0	0	1	Will not occur
0	0	1	0	Will not occur
0	0	1	1	None; not a status cycle
0	1	0	0	IF A1 = 1 then halt; else shutdown
0	1	0	1	Memory data read
0	1	1	0	Memory data write
0	1	1	1	None; not a status cycle
1 (HIGH)	0	0	0	Will not occur
1	0	0	1	I/O read
1	0	1	0	I/O write
1	0	1	1	None; not a status cycle
1	1	0	0	Will not occur
1	1	0	1	Memory instruction read
1	1	1	0	Will not occur
1	1	1	1	None; not a status cycle

M/$\overline{IO}$	O	**MEMORY I/O SELECT** distinguishes memory access from I/O access. If HIGH during T_s, a memory cycle or a halt/shutdown cycle is in progress. If LOW, an I/O cycle or an interrupt acknowledge cycle is in progress. M/$\overline{IO}$ floats to 3-state OFF during bus hold acknowledge.
COD/$\overline{INTA}$	O	**CODE/INTERRUPT ACKNOWLEDGE** distinguishes instruction fetch cycles from memory data read cycles. Also distinguishes interrupt acknowledge cycles from I/O cycles. COD/$\overline{INTA}$ floats to 3-state OFF during bus hold acknowledge. Its timing is the same as M/$\overline{IO}$.
$\overline{LOCK}$	O	**BUS LOCK** indicates that other system bus masters are not to gain control of the system bus for the current and the following bus cycle. The $\overline{LOCK}$ signal may be activated explicitly by the "LOCK" instruction prefix or automatically by 80286 hardware during memory XCHG instructions, interrupt acknowledge, or descriptor table access. $\overline{LOCK}$ is active LOW and floats to 3-state OFF during bus hold acknowledge.
$\overline{READY}$	I	**BUS READY** terminates a bus cycle. Bus cycles are extended without limit until terminated by $\overline{READY}$ LOW. $\overline{READY}$ is an active LOW synchronous input requiring setup and hold times relative to the system clock be met for correct operation. $\overline{READY}$ is ignored during bus hold acknowledge.

Table 1. Pin Description (Continued)

Symbol	Type	Name and Function
HOLD HLDA	I O	**BUS HOLD REQUEST AND HOLD ACKNOWLEDGE** control ownership of the 80286 local bus. The HOLD input allows another local bus master to request control of the local bus. When control is granted, the 80286 will float its bus drivers to 3-state OFF and then activate HLDA, thus entering the bus hold acknowledge condition. The local bus will remain granted to the requesting master until HOLD becomes inactive which results in the 80286 deactivating HLDA and regaining control of the local bus. This terminates the bus hold acknowledge condition. HOLD may be asynchronous to the system clock. These signals are active HIGH.
INTR	I	**INTERRUPT REQUEST** requests the 80286 to suspend its current program execution and service a pending external request. Interrupt requests are masked whenever the interrupt enable bit in the flag word is cleared. When the 80286 responds to an interrupt request, it performs two interrupt acknowledge bus cycles to read an 8-bit interrupt vector that identifies the source of the interrupt. To assure program interruption, INTR must remain active until the first interrupt acknowledge cycle is completed. INTR is sampled at the beginning of each processor cycle and must be active HIGH at least two processor cycles before the current instruction ends in order to interrupt before the next instruction. INTR is level sensitive, active HIGH, and may be asynchronous to the system clock.
NMI	I	**NON-MASKABLE INTERRUPT REQUEST** interrupts the 80286 with an internally supplied vector value of 2. No interrupt acknowledge cycles are performed. The interrupt enable bit in the 80286 flag word does not affect this input. The NMI input is active HIGH, may be asynchronous to the system clock, and is edge triggered after internal synchronization. For proper recognition, the input must have been previously LOW for at least four system clock cycles and remain HIGH for at least four system clock cycles.
PEREQ $\overline{\text{PEACK}}$	I O	**PROCESSOR EXTENSION OPERAND REQUEST AND ACKNOWLEDGE** extend the memory management and protection capabilities of the 80286 to processor extensions. The PEREQ input requests the 80286 to perform a data operand transfer for a processor extension. The $\overline{\text{PEACK}}$ output signals the processor extension when the requested operand is being transferred. PEREQ is active HIGH and floats to 3-state OFF during bus hold acknowledge. $\overline{\text{PEACK}}$ may be asynchronous to the system clock. $\overline{\text{PEACK}}$ is active LOW.
$\overline{\text{BUSY}}$ $\overline{\text{ERROR}}$	I I	**PROCESSOR EXTENSION BUSY AND ERROR** indicate the operating condition of a processor extension to the 80286. An active $\overline{\text{BUSY}}$ input stops 80286 program execution on WAIT and some ESC instructions until $\overline{\text{BUSY}}$ becomes inactive (HIGH). The 80286 may be interrupted while waiting for $\overline{\text{BUSY}}$ to become inactive. An active $\overline{\text{ERROR}}$ input causes the 80286 to perform a processor extension interrupt when executing WAIT or some ESC instructions. These inputs are active LOW and may be asynchronous to the system clock. These inputs have internal pull-up resistors.

Table 1. Pin Description (Continued)

Symbol	Type	Name and Function
RESET	I	**SYSTEM RESET** clears the internal logic of the 80286 and is active HIGH. The 80286 may be reinitialized at any time with a LOW to HIGH transition on RESET which remains active for more than 16 system clock cycles. During RESET active, the output pins of the 80286 enter the state shown below:
		80286 Pin State During Reset
		Pin Value / **Pin Names** 1 (HIGH) / $\overline{S0}$, $\overline{S1}$, $\overline{PEACK}$, A23–A0, $\overline{BHE}$, $\overline{LOCK}$ 0 (LOW) / M/$\overline{IO}$, COD/$\overline{INTA}$, HLDA (Note 1) 3-state OFF / D_{15}–D_0
		Operation of the 80286 begins after a HIGH to LOW transition on RESET. The HIGH to LOW transition of RESET must be synchronous to the system clock. Approximately 38 CLK cycles from the trailing edge of RESET are required by the 80286 for internal initialization before the first bus cycle, to fetch code from the power-on execution address, occurs. A LOW to HIGH transition of RESET synchronous to the system clock will end a processor cycle at the second HIGH to LOW transition of the system clock. The LOW to HIGH transition of RESET may be asynchronous to the system clock; however, in this case it cannot be predetermined which phase of the processor clock will occur during the next system clock period. Synchronous LOW to HIGH transitions of RESET are required only for systems where the processor clock must be phase synchronous to another clock.
V_{SS}	I	**SYSTEM GROUND:** 0 Volts.
V_{CC}	I	**SYSTEM POWER:** +5 Volt Power Supply.
CAP	I	**SUBSTRATE FILTER CAPACITOR:** a 0.047 μF ± 20% 12V capacitor must be connected between this pin and ground. This capacitor filters the output of the internal substrate bias generator. A maximum DC leakage current of 1 μA is allowed through the capacitor. For correct operation of the 80286, the substrate bias generator must charge this capacitor to its operating voltage. The capacitor chargeup time is 5 milliseconds (max.) after V_{CC} and CLK reach their specified AC and DC parameters. RESET may be applied to prevent spurious activity by the CPU during this time. After this time, the 80286 processor clock can be synchronized to another clock by pulsing RESET LOW synchronous to the system clock.

NOTE:
1. HLDA is only Low if HOLD is inactive (Low).

FUNCTIONAL DESCRIPTION

Introduction

The 80286 is an advanced, high-performance microprocessor with specially optimized capabilities for multiple user and multi-tasking systems. Depending on the application, a 12 MHz 80286's performance is up to six times faster than the standard 5 MHz 8086's, while providing complete upward software compatibility with Intel's 8086, 88, and 186 family of CPU's.

The 80286 operates in two modes: 8086 real address mode and protected virtual address mode. Both modes execute a superset of the 8086 and 88 instruction set.

In 8086 real address mode programs use real addresses with up to one megabyte of address space. Programs use virtual addresses in protected virtual address mode, also called protected mode. In protected mode, the 80286 CPU automatically maps 1 gigabyte of virtual addresses per task into a 16 megabyte real address space. This mode also provides memory protection to isolate the operating system and ensure privacy of each tasks' programs and data. Both modes provide the same base instruction set, registers, and addressing modes.

The following Functional Description describes first, the base 80286 architecture common to both modes, second, 8086 real address mode, and third, protected mode.

80286 BASE ARCHITECTURE

The 8086, 88, 186, and 286 CPU family all contain the same basic set of registers, instructions, and addressing modes. The 80286 processor is upward compatible with the 8086, 8088, and 80186 CPU's.

Register Set

The 80286 base architecture has fifteen registers as shown in Figure 3. These registers are grouped into the following four categories:

General Registers: Eight 16-bit general purpose registers used to contain arithmetic and logical operands. Four of these (AX, BX, CX, and DX) can be used either in their entirety as 16-bit words or split into pairs of separate 8-bit registers.

Segment Registers: Four 16-bit special purpose registers select, at any given time, the segments of memory that are immediately addressable for code, stack, and data. (For usage, refer to Memory Organization.)

Base and Index Registers: Four of the general purpose registers may also be used to determine offset addresses of operands in memory. These registers may contain base addresses or indexes to particular locations within a segment. The addressing mode determines the specific registers used for operand address calculations.

Status and Control Registers: The 3 16-bit special purpose registers in figure 3A record or control certain aspects of the 80286 processor state including the Instruction Pointer, which contains the offset address of the next sequential instruction to be executed.

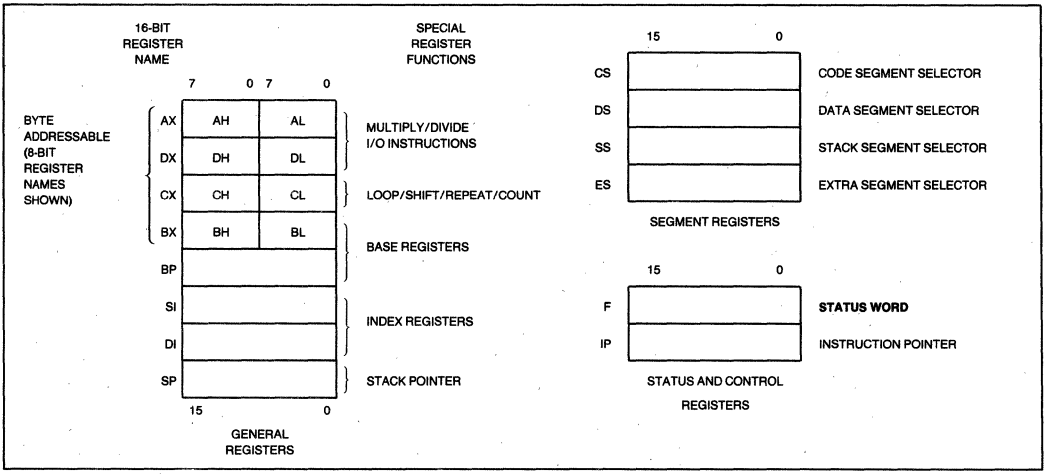


Figure 3. Register Set

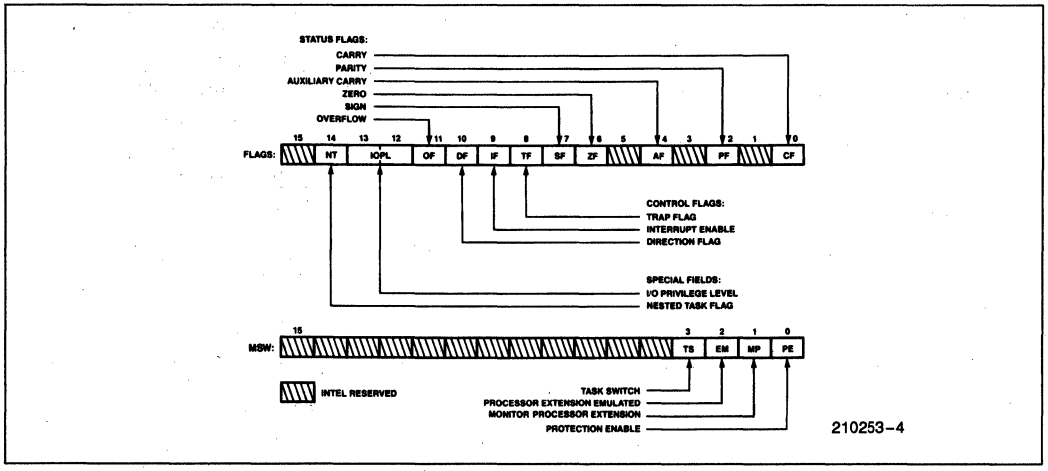


Figure 3a. Status and Control Register Bit Functions

Flags Word Description

The Flags word (Flags) records specific characteristics of the result of logical and arithmetic instructions (bits 0, 2, 4, 6, 7, and 11) and controls the operation of the 80286 within a given operating mode (bits 8 and 9). Flags is a 16-bit register. The function of the flag bits is given in Table 2.

Table 2. Flags Word Bit Functions

Bit Position	Name	Function
0	CF	Carry Flag—Set on high-order bit carry or borrow; cleared otherwise
2	PF	Parity Flag—Set if low-order 8 bits of result contain an even number of 1-bits; cleared otherwise
4	AF	Set on carry from or borrow to the low order four bits of AL; cleared otherwise
6	ZF	Zero Flag—Set if result is zero; cleared otherwise
7	SF	Sign Flag—Set equal to high-order bit of result (0 if positive, 1 if negative)
11	OF	Overflow Flag—Set if result is a too-large positive number or a too-small negative number (excluding sign-bit) to fit in destination operand; cleared otherwise
8	TF	Single Step Flag—Once set, a single step interrupt occurs after the next instruction executes. TF is cleared by the single step interrupt.
9	IF	Interrupt-enable Flag—When set, maskable interrupts will cause the CPU to transfer control to an interrupt vector specified location.
10	DF	Direction Flag—Causes string instructions to auto decrement the appropriate index registers when set. Clearing DF causes auto increment.

Instruction Set

The instruction set is divided into seven categories: data transfer, arithmetic, shift/rotate/logical, string manipulation, control transfer, high level instructions, and processor control. These categories are summarized in Figure 4.

An 80286 instruction can reference zero, one, or two operands; where an operand resides in a register, in the instruction itself, or in memory. Zero-operand instructions (e.g. NOP and HLT) are usually one byte long. One-operand instructions (e.g. INC and DEC) are usually two bytes long but some are encoded in only one byte. One-operand instructions may reference a register or memory location. Two-operand instructions permit the following six types of instruction operations:

—Register to Register
—Memory to Register
—Immediate to Register
—Memory to Memory
—Register to Memory
—Immediate to Memory

Two-operand instructions (e.g. MOV and ADD) are usually three to six bytes long. Memory to memory operations are provided by a special class of string instructions requiring one to three bytes. For detailed instruction formats and encodings refer to the instruction set summary at the end of this document.

For detailed operation and usage of each instruction, see Appendix of 80286 Programmer's Reference Manual (Order No. 210498)

GENERAL PURPOSE	
MOV	Move byte or word
PUSH	Push word onto stack
POP	Pop word off stack
PUSHA	Push all registers on stack
POPA	Pop all registers from stack
XCHG	Exchange byte or word
XLAT	Translate byte
INPUT/OUTPUT	
IN	Input byte or word
OUT	Output byte or word
ADDRESS OBJECT	
LEA	Load effective address
LDS	Load pointer using DS
LES	Load pointer using ES
FLAG TRANSFER	
LAHF	Load AH register from flags
SAHF	Store AH register in flags
PUSHF	Push flags onto stack
POPF	Pop flags off stack

Figure 4a. Data Transfer Instructions

MOVS	Move byte or word string
INS	Input bytes or word string
OUTS	Output bytes or word string
CMPS	Compare byte or word string
SCAS	Scan byte or word string
LODS	Load byte or word string
STOS	Store byte or word string
REP	Repeat
REPE/REPZ	Repeat while equal/zero
REPNE/REPNZ	Repeat while not equal/not zero

Figure 4c. String Instructions

ADDITION	
ADD	Add byte or word
ADC	Add byte or word with carry
INC	Increment byte or word by 1
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition
SUBTRACTION	
SUB	Subtract byte or word
SBB	Subtract byte or word with borrow
DEC	Decrement byte or word by 1
NEG	Negate byte or word
CMP	Compare byte or word
AAS	ASCII adjust for subtraction
DAS	Decimal adjust for subtraction
MULTIPLICATION	
MUL	Multiple byte or word unsigned
IMUL	Integer multiply byte or word
AAM	ASCII adjust for multiply
DIVISION	
DIV	Divide byte or word unsigned
IDIV	Integer divide byte or word
AAD	ASCII adjust for division
CBW	Convert byte to word
CWD	Convert word to doubleword

Figure 4b. Arithmetic Instructions

LOGICALS	
NOT	"Not" byte or word
AND	"And" byte or word
OR	"Inclusive or" byte or word
XOR	"Exclusive or" byte or word
TEST	"Test" byte or word
SHIFTS	
SHL/SAL	Shift logical/arithmetic left byte or word
SHR	Shift logical right byte or word
SAR	Shift arithmetic right byte or word
ROTATES	
ROL	Rotate left byte or word
ROR	Rotate right byte or word
RCL	Rotate through carry left byte or word
RCR	Rotate through carry right byte or word

Figure 4d. Shift/Rotate Logical Instructions

CONDITIONAL TRANSFERS	
JA/JNBE	Jump if above/not below nor equal
JAE/JNB	Jump if above or equal/not below
JB/JNAE	Jump if below/not above nor equal
JBE/JNA	Jump if below or equal/not above
JC	Jump if carry
JE/JZ	Jump if equal/zero
JG/JNLE	Jump if greater/not less nor equal
JGE/JNL	Jump if greater or equal/not less
JL/JNGE	Jump if less/not greater nor equal
JLE/JNG	Jump if less or equal/not greater
JNC	Jump if not carry
JNE/JNZ	Jump if not equal/not zero
JNO	Jump if not overflow
JNP/JPO	Jump if not parity/parity odd
JNS	Jump if not sign
JO	Jump if overflow
JP/JPE	Jump if parity/parity even
JS	Jump if sign

UNCONDITIONAL TRANSFERS	
CALL	Call procedure
RET	Return from procedure
JMP	Jump
ITERATION CONTROLS	
LOOP	Loop
LOOPE/LOOPZ	Loop if equal/zero
LOOPNE/LOOPNZ	Loop if not equal/not zero
JCXZ	Jump if register CX = 0
INTERRUPTS	
INT	Interrupt
INTO	Interrupt if overflow
IRET	Interrupt return

Figure 4e. Program Transfer Instructions

FLAG OPERATIONS	
STC	Set carry flag
CLC	Clear carry flag
CMC	Complement carry flag
STD	Set direction flag
CLD	Clear direction flag
STI	Set interrupt enable flag
CLI	Clear interrupt enable flag
EXTERNAL SYNCHRONIZATION	
HLT	Halt until interrupt or reset
WAIT	Wait for $\overline{\text{BUSY}}$ not active
ESC	Escape to extension processor
LOCK	Lock bus during next instruction
NO OPERATION	
NOP	No operation
EXECUTION ENVIRONMENT CONTROL	
LMSW	Load machine status word
SMSW	Store machine status word

Figure 4f. Processor Control Instructions

ENTER	Format stack for procedure entry
LEAVE	Restore stack for procedure exit
BOUND	Detects values outside prescribed range

Figure 4g. High Level Instructions

Memory Organization

Memory is organized as sets of variable length segments. Each segment is a linear contiguous sequence of up to 64K (2^{16}) 8-bit bytes. Memory is addressed using a two component address (a pointer) that consists of a 16-bit segment selector, and a 16-bit offset. The segment selector indicates the desired segment in memory. The offset component indicates the desired byte address within the segment.

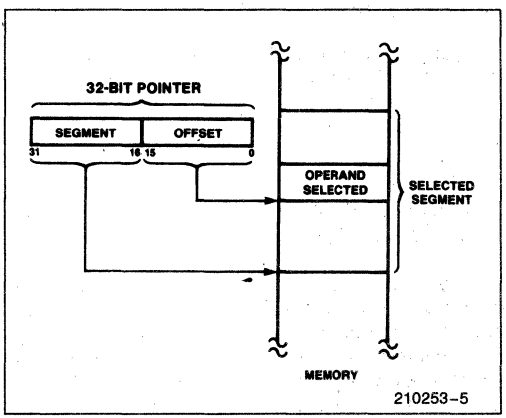


Figure 5. Two Component Address

Table 3. Segment Register Selection Rules

Memory Reference Needed	Segment Register Used	Implicit Segment Selection Rule
Instructions	Code (CS)	Automatic with instruction prefetch
Stack	Stack (SS)	All stack pushes and pops. Any memory reference which uses BP as a base register.
Local Data	Data (DS)	All data references except when relative to stack or string destination
External (Global) Data	Extra (ES)	Alternate data segment and destination of string operation

All instructions that address operands in memory must specify the segment and the offset. For speed and compact instruction encoding, segment selectors are usually stored in the high speed segment registers. An instruction need specify only the desired segment register and an offset in order to address a memory operand.

Most instructions need not explicitly specify which segment register is used. The correct segment register is automatically chosen according to the rules of Table 3. These rules follow the way programs are written (see Figure 6) as independent modules that require areas for code and data, a stack, and access to external data areas.

Special segment override instruction prefixes allow the implicit segment register selection rules to be overridden for special cases. The stack, data, and extra segments may coincide for simple programs. To access operands not residing in one of the four immediately available segments, a full 32-bit pointer or a new segment selector must be loaded.

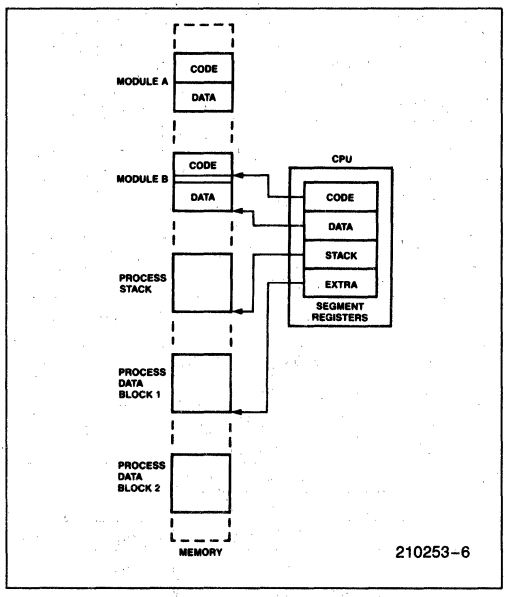


Figure 6. Segmented Memory Helps Structure Software

Addressing Modes

The 80286 provides a total of eight addressing modes for instructions to specify operands. Two addressing modes are provided for instructions that operate on register or immediate operands:

Register Operand Mode: The operand is located in one of the 8 or 16-bit general registers.

Immediate Operand Mode: The operand is included in the instruction.

Six modes are provided to specify the location of an operand in a memory segment. A memory operand address consists of two 16-bit components: segment selector and offset. The segment selector is supplied by a segment register either implicitly chosen by the addressing mode or explicitly chosen by a segment override prefix. The offset is calculated by summing any combination of the following three address elements:

the **displacement** (an 8 or 16-bit immediate value contained in the instruction)

the **base** (contents of either the BX or BP base registers)

the **index** (contents of either the SI or DI index registers)

Any carry out from the 16-bit addition is ignored. Eight-bit displacements are sign extended to 16-bit values.

Combinations of these three address elements define the six memory addressing modes, described below.

Direct Mode: The operand's offset is contained in the instruction as an 8 or 16-bit displacement element.

Register Indirect Mode: The operand's offset is in one of the registers SI, DI, BX, or BP.

Based Mode: The operand's offset is the sum of an 8 or 16-bit displacement and the contents of a base register (BX or BP).

Indexed Mode: The operand's offset is the sum of an 8 or 16-bit displacement and the contents of an index register (SI or DI).

Based Indexed Mode: The operand's offset is the sum of the contents of a base register and an index register.

Based Indexed Mode with Displacement: The operand's offset is the sum of a base register's contents, an index register's contents, and an 8 or 16-bit displacement.

Data Types

The 80286 directly supports the following data types:

Integer: A signed binary numeric value contained in an 8-bit byte or a 16-bit word. All operations assume a 2's complement representation. Signed 32 and 64-bit integers are supported using the Numeric Data Processor, the 80287.

Ordinal: An unsigned binary numeric value contained in an 8-bit byte or 16-bit word.

Pointer: A 32-bit quantity, composed of a segment selector component and an offset component. Each component is a 16-bit word.

String: A contiguous sequence of bytes or words. A string may contain from 1 byte to 64K bytes.

ASCII: A byte representation of alphanumeric and control characters using the ASCII standard of character representation.

BCD: A byte (unpacked) representation of the decimal digits 0–9.

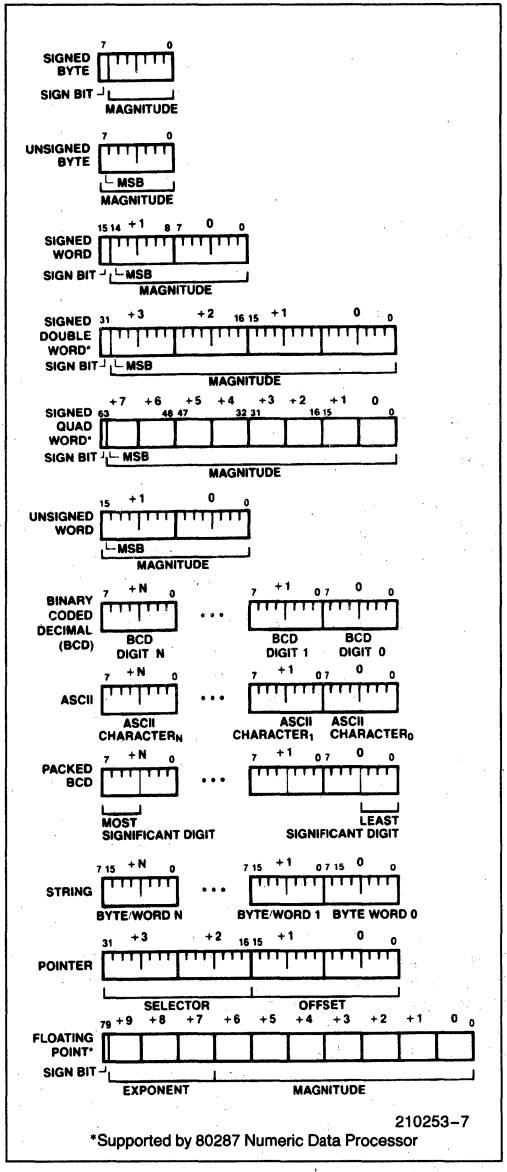
Packed BCD: A byte (packed) representation of two decimal digits 0–9 storing one digit in each nibble of the byte.

Floating Point: A signed 32, 64, or 80-bit real number representation. (Floating point operands are supported using the 80287 Numeric Processor).

Figure 7 graphically represents the data types supported by the 80286.

I/O Space

The I/O space consists of 64K 8-bit or 32K 16-bit ports. I/O instructions address the I/O space with either an 8-bit port address, specified in the instruction, or a 16-bit port address in the DX register. 8-bit port addresses are zero extended such that A_{15}–A_8 are LOW. I/O port addresses 00F8(H) through 00FF(H) are reserved.



210253-7

*Supported by 80287 Numeric Data Processor

Figure 7. 80286 Supported Data Types

Table 4. Interrupt Vector Assignments

Function	Interrupt Number	Related Instructions	Does Return Address Point to Instruction Causing Exception?
Divide error exception	0	DIV, IDIV	Yes
Single step interrupt	1	All	
NMI interrupt	2	INT 2 or NMI pin	
Breakpoint interrupt	3	INT 3	
INTO detected overflow exception	4	INTO	No
BOUND range exceeded exception	5	BOUND	Yes
Invalid opcode exception	6	Any undefined opcode	Yes
Processor extension not available exception	7	ESC or WAIT	Yes
Intel reserved–do not use	8-15		
Processor extension error interrupt	16	ESC or WAIT	
Intel reserved–do not use	17-31		
User defined	32-255		

Interrupts

An interrupt transfers execution to a new program location. The old program address (CS:IP) and machine state (Flags) are saved on the stack to allow resumption of the interrupted program. Interrupts fall into three classes: hardware initiated, INT instructions, and instruction exceptions. Hardware initiated interrupts occur in response to an external input and are classified as non-maskable or maskable. Programs may cause an interrupt with an INT instruction. Instruction exceptions occur when an unusual condition, which prevents further instruction processing, is detected while attempting to execute an instruction. The return address from an exception will always point at the instruction causing the exception and include any leading instruction prefixes.

A table containing up to 256 pointers defines the proper interrupt service routine for each interrupt. Interrupts 0–31, some of which are used for instruction exceptions, are reserved. For each interrupt, an 8-bit vector must be supplied to the 80286 which identifies the appropriate table entry. Exceptions supply the interrupt vector internally. INT instructions contain or imply the vector and allow access to all 256 interrupts. Maskable hardware initiated interrupts supply the 8-bit vector to the CPU during an interrupt acknowledge bus sequence. Non-maskable hardware interrupts use a predefined internally supplied vector.

MASKABLE INTERRUPT (INTR)

The 80286 provides a maskable hardware interrupt request pin, INTR. Software enables this input by setting the interrupt flag bit (IF) in the flag word. All 224 user-defined interrupt sources can share this input, yet they can retain separate interrupt handlers. An 8-bit vector read by the CPU during the interrupt acknowledge sequence (discussed in System Interface section) identifies the source of the interrupt.

Further maskable interrupts are disabled while servicing an interrupt by resetting the IF bit as part of the response to an interrupt or exception. The saved flag word will reflect the enable status of the processor prior to the interrupt. Until the flag word is restored to the flag register, the interrupt flag will be zero unless specifically set. The interrupt return instruction includes restoring the flag word, thereby restoring the original status of IF.

NON-MASKABLE INTERRUPT REQUEST (NMI)

A non-maskable interrupt input (NMI) is also provided. NMI has higher priority than INTR. A typical use of NMI would be to activate a power failure routine. The activation of this input causes an interrupt with an internally supplied vector value of 2. No external interrupt acknowledge sequence is performed.

While executing the NMI servicing procedure, the 80286 will service neither further NMI requests, INTR requests, nor the processor extension segment overrun interrupt until an interrupt return (IRET) instruction is executed or the CPU is reset. If NMI occurs while currently servicing an NMI, its presence will be saved for servicing after executing the first IRET instruction. IF is cleared at the beginning of an NMI interrupt to inhibit INTR interrupts.

SINGLE STEP INTERRUPT

The 80286 has an internal interrupt that allows programs to execute one instruction at a time. It is called the single step interrupt and is controlled by the single step flag bit (TF) in the flag word. Once this bit is set, an internal single step interrupt will occur after the next instruction has been executed. The interrupt clears the TF bit and uses an internally supplied vector of 1. The IRET instruction is used to set the TF bit and transfer control to the next instruction to be single stepped.

Interrupt Priorities

When simultaneous interrupt requests occur, they are processed in a fixed order as shown in Table 5. Interrupt processing involves saving the flags, return address, and setting CS:IP to point at the first instruction of the interrupt handler. If other interrupts remain enabled they are processed before the first instruction of the current interrupt handler is executed. The last interrupt processed is therefore the first one serviced.

Table 5. Interrupt Processing Order

Order	Interrupt
1	Instruction exception
2	Single step
3	NMI
4	Processor extension segment overrun
5	INTR
6	INT instruction

Initialization and Processor Reset

Processor initialization or start up is accomplished by driving the RESET input pin HIGH. RESET forces the 80286 to terminate all execution and local bus activity. No instruction or bus activity will occur as long as RESET is active. After RESET becomes inactive and an internal processing interval elapses, the 80286 begins execution in real address mode with the instruction at physical location FFFFF0(H). RESET also sets some registers to predefined values as shown in Table 6.

Table 6. 80286 Initial Register State after RESET

Flag word	0002(H)
Machine Status Word	FFF0(H)
Instruction pointer	FFF0(H)
Code segment	F000(H)
Data segment	0000(H)
Extra segment	0000(H)
Stack segment	0000(H)

HOLD must not be active during the time from the leading edge of RESET to 34 CLKs after the trailing edge of RESET.

Machine Status Word Description

The machine status word (MSW) records when a task switch takes place and controls the operating mode of the 80286. It is a 16-bit register of which the lower four bits are used. One bit places the CPU into protected mode, while the other three bits, as shown in Table 7, control the processor extension interface. After RESET, this register contains FFF0(H) which places the 80286 in 8086 real address mode.

Table 7. MSW Bit Functions

Bit Position	Name	Function
0	PE	Protected mode enable places the 80286 into protected mode and cannot be cleared except by RESET.
1	MP	Monitor processor extension allows WAIT instructions to cause a processor extension not present exception (number 7).
2	EM	Emulate processor extension causes a processor extension not present exception (number 7) on ESC instructions to allow emulating a processor extension.
3	TS	Task switched indicates the next instruction using a processor extension will cause exception 7, allowing software to test whether the current processor extension context belongs to the current task.

The LMSW and SMSW instructions can load and store the MSW in real address mode. The recommended use of TS, EM, and MP is shown in Table 8.

Table 8. Recommended MSW Encodings For Processor Extension Control

TS	MP	EM	Recommended Use	Instructions Causing Exception 7
0	0	0	Initial encoding after RESET. 80286 operation is identical to 8086, 88.	None
0	0	1	No processor extension is available. Software will emulate its function.	ESC
1	0	1	No processor extension is available. Software will emulate its function. The current processor extension context may belong to another task.	ESC
0	1	0	A processor extension exists.	None
1	1	0	A processor extension exists. The current processor extension context may belong to another task. The Exception 7 on WAIT allows software to test for an error pending from a previous processor extension operation.	ESC or WAIT

Halt

The HLT instruction stops program execution and prevents the CPU from using the local bus until restarted. Either NMI, INTR with IF = 1, or RESET will force the 80286 out of halt. If interrupted, the saved CS:IP will point to the next instruction after the HLT.

8086 REAL ADDRESS MODE

The 80286 executes a fully upward-compatible superset of the 8086 instruction set in real address mode. In real address mode the 80286 is object code compatible with 8086 and 8088 software. The real address mode architecture (registers and addressing modes) is exactly as described in the 80286 Base Architecture section of this Functional Description.

Memory Size

Physical memory is a contiguous array of up to 1,048,576 bytes (one megabyte) addressed by pins A_0 through A_{19} and $\overline{BHE}$. A_{20} through A_{23} should be ignored.

Memory Addressing

In real address mode physical memory is a contiguous array of up to 1,048,576 bytes (one megabyte) addressed by pins A_0 through A_{19} and $\overline{BHE}$. Address bits A_{20}–A_{23} may not always be zero in real mode. A_{20}–A_{23} should not be used by the system while the 80286 is operating in Real Mode.

The selector portion of a pointer is interpreted as the upper 16 bits of a 20-bit segment address. The lower four bits of the 20-bit segment address are always zero. Segment addresses, therefore, begin on multiples of 16 bytes. See Figure 8 for a graphic representation of address information.

All segments in real address mode are 64K bytes in size and may be read, written, or executed. An exception or interrupt can occur if data operands or instructions attempt to wrap around the end of a segment (e.g. a word with its low order byte at offset FFFF(H) and its high order byte at offset 0000(H). If, in real address mode, the information contained in a segment does not use the full 64K bytes, the unused end of the segment may be overlayed by another segment to reduce physical memory requirements.

Reserved Memory Locations

The 80286 reserves two fixed areas of memory in real address mode (see Figure 9); system initialization area and interrupt table area. Locations from addresses FFFF0(H) through FFFFF(H) are reserved for system initialization. Initial execution begins at location FFFF0(H). Locations 00000(H) through 003FF(H) are reserved for interrupt vectors.

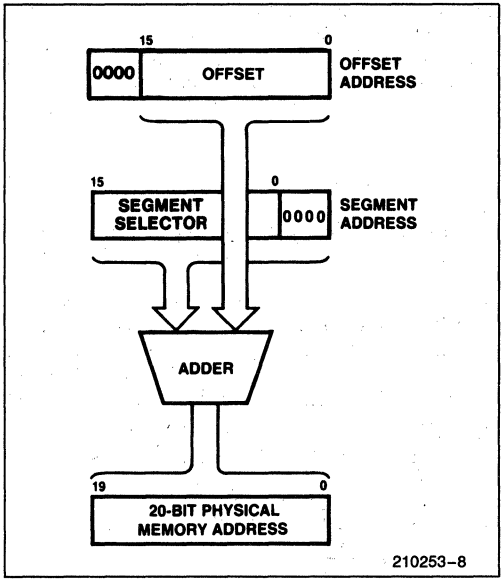


Figure 8. 8086 Real Address Mode Address Calculation

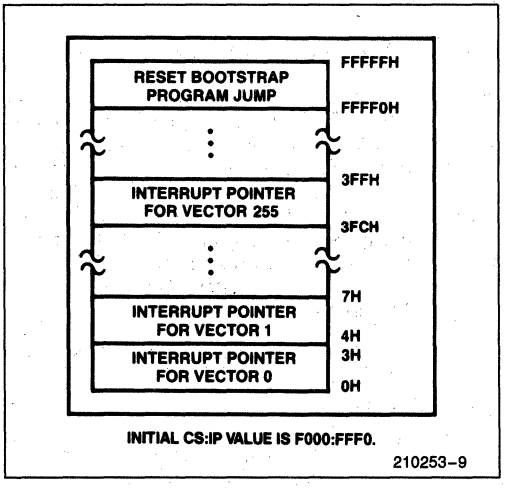


Figure 9. 8086 Real Address Mode Initially Reserved Memory Locations

Table 9. Real Address Mode Addressing Interrupts

Function	Interrupt Number	Related Instructions	Return Address Before Instruction?
Interrupt table limit too small exception	8	INT vector is not within table limit	Yes
Processor extension segment overrun interrupt	9	ESC with memory operand extending beyond offset FFFF(H)	No
Segment overrun exception	13	Word memory reference with offset = FFFF(H) or an attempt to execute past the end of a segment	Yes

Interrupts

Table 9 shows the interrupt vectors reserved for exceptions and interrupts which indicate an addressing error. The exceptions leave the CPU in the state existing before attempting to execute the failing instruction (except for PUSH, POP, PUSHA, or POPA). Refer to the next section on protected mode initialization for a discussion on exception 8.

Protected Mode Initialization

To prepare the 80286 for protected mode, the LIDT instruction is used to load the 24-bit interrupt table base and 16-bit limit for the protected mode interrupt table. This instruction can also set a base and limit for the interrupt vector table in real address mode. After reset, the interrupt table base is initialized to 000000(H) and its size set to 03FF(H). These values are compatible with 8086, 88 software. LIDT should only be executed in preparation for protected mode.

Shutdown

Shutdown occurs when a severe error is detected that prevents further instruction processing by the CPU. Shutdown and halt are externally signalled via a halt bus operation. They can be distinguished by A_1 HIGH for halt and A_1 LOW for shutdown. In real address mode, shutdown can occur under two conditions:

- Exceptions 8 or 13 happen and the IDT limit does not include the interrupt vector.
- A CALL INT or PUSH instruction attempts to wrap around the stack segment when SP is not even.

An NMI input can bring the CPU out of shutdown if the IDT limit is at least 000F(H) and SP is greater than 0005(H), otherwise shutdown can only be exited via the RESET input.

PROTECTED VIRTUAL ADDRESS MODE

The 80286 executes a fully upward-compatible superset of the 8086 instruction set in protected virtual address mode (protected mode). Protected mode also provides memory management and protection mechanisms and associated instructions.

The 80286 enters protected virtual address mode from real address mode by setting the PE (Protection Enable) bit of the machine status word with the Load Machine Status Word (LMSW) instruction. Protected mode offers extended physical and virtual memory address space, memory protection mechanisms, and new operations to support operating systems and virtual memory.

All registers, instructions, and addressing modes described in the 80286 Base Architecture section of this Functional Description remain the same. Programs for the 8086, 88, 186, and real address mode 80286 can be run in protected mode; however, embedded constants for segment selectors are different.

Memory Size

The protected mode 80286 provides a 1 gigabyte virtual address space per task mapped into a 16 megabyte physical address space defined by the address pin A_{23}-A_0 and $\overline{BHE}$. The virtual address space may be larger than the physical address space since any use of an address that does not map to a physical memory location will cause a restartable exception.

Memory Addressing

As in real address mode, protected mode uses 32-bit pointers, consisting of 16-bit selector and offset components. The selector, however, specifies an index into a memory resident table rather than the upper 16-bits of a real memory address. The 24-bit

base address of the desired segment is obtained from the tables in memory. The 16-bit offset is added to the segment base address to form the physical address as shown in Figure 10. The tables are automatically referenced by the CPU whenever a segment register is loaded with a selector. All 80286 instructions which load a segment register will reference the memory based tables without additional software. The memory based tables contain 8 byte values called descriptors.

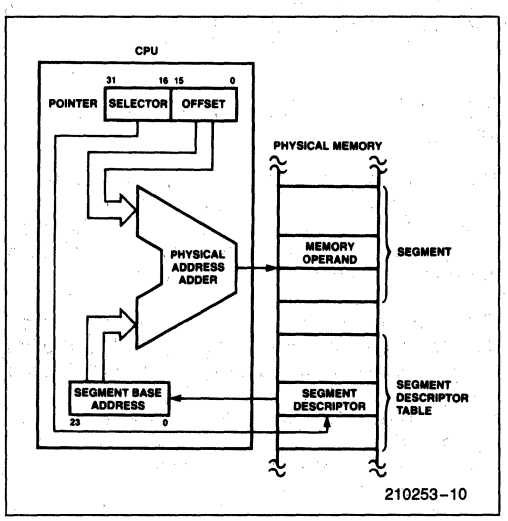


Figure 10. Protected Mode Memory Addressing

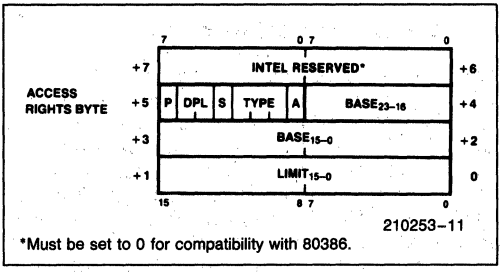
DESCRIPTORS

Descriptors define the use of memory. Special types of descriptors also define new functions for transfer of control and task switching. The 80286 has segment descriptors for code, stack and data segments, and system control descriptors for special system data segments and control transfer operations. Descriptor accesses are performed as locked bus operations to assure descriptor integrity in multi-processor systems.

CODE AND DATA SEGMENT DESCRIPTORS (S = 1)

Besides segment base addresses, code and data descriptors contain other segment attributes including segment size (1 to 64K bytes), access rights (read only, read/write, execute only, and execute/read), and presence in memory (for virtual memory systems) (See Figure 11). Any segment usage violating a segment attribute indicated by the segment descriptor will prevent the memory cycle and cause an exception or interrupt.

Code or Data Segment Descriptor



*Must be set to 0 for compatibility with 80386.

Access Rights Byte Definition

	Bit Position	Name	Function		
Type Field Definition	7	Present (P)	P = 1	Segment is mapped into physical memory.	
			P = 0	No mapping to physical memory exits, base and limit are not used.	
	6–5	Descriptor Privilege Level (DPL)		Segment privilege attribute used in privilege tests.	
	4	Segment Descriptor (S)	S = 1	Code or Data (includes stacks) segment descriptor	
			S = 0	System Segment Descriptor or Gate Descriptor	
	3	Executable (E)	E = 0	Data segment descriptor type is:	If Data Segment (S = 1, E = 0)
	2	Expansion Direction (ED)	ED = 0	Expand up segment, offsets must be ≤ limit.	
			ED = 1	Expand down segment, offsets must be > limit.	
	1	Writeable (W)	W = 0	Data segment may not be written into.	
			W = 1	Data segment may be written into.	
	3	Executable (E)	E = 1	Code Segment Descriptor type is:	If Code Segment (S = 1, E = 1)
	2	Conforming (C)	C = 1	Code segment may only be executed when CPL ≥ DPL and CPL remains unchanged.	
	1	Readable (R)	R = 0	Code segment may not be read	
			R = 1	Code segment may be read.	
	0	Accessed (A)	A = 0	Segment has not been accessed.	
			A = 1	Segment selector has been loaded into segment register or used by selector test instructions.	

Figure 11. Code and Data Segment Descriptor Formats

Code and data (including stack data) are stored in two types of segments: code segments and data segments. Both types are identified and defined by segment descriptors (S = 1). Code segments are identified by the executable (E) bit set to 1 in the descriptor access rights byte. The access rights byte of both code and data segment descriptor types have three fields in common: present (P) bit, Descriptor Privilege Level (DPL), and accessed (A) bit. If P = 0, any attempted use of this segment will cause a not-present exception. DPL specifies the privilege level of the segment descriptor. DPL controls when the descriptor may be used by a task (refer to privilege discussion below). The A bit shows whether the segment has been previously accessed for usage profiling, a necessity for virtual memory systems. The CPU will always set this bit when accessing the descriptor.

Data segments (S = 1, E = 0) may be either read-only or read-write as controlled by the W bit of the access rights byte. Read-only (W = 0) data segments may not be written into. Data segments may grow in two directions, as determined by the Expansion Direction (ED) bit: upwards (ED = 0) for data segments, and downwards (ED = 1) for a segment containing a stack. The limit field for a data segment descriptor is interpreted differently depending on the ED bit (see Figure 11).

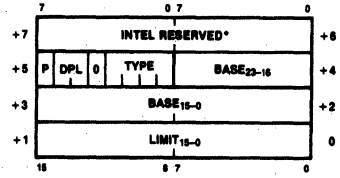
A code segment (S = 1, E = 1) may be execute-only or execute/read as determined by the Readable (R) bit. Code segments may never be written into and execute-only code segments (R = 0) may not be read. A code segment may also have an attribute called conforming (C). A conforming code segment may be shared by programs that execute at different privilege levels. The DPL of a conforming code segment defines the range of privilege levels at which the segment may be executed (refer to privilege discussion below). The limit field identifies the last byte of a code segment.

SYSTEM SEGMENT DESCRIPTORS (S = 0, TYPE = 1-3)

In addition to code and data segment descriptors, the protected mode 80286 defines System Segment Descriptors. These descriptors define special system data segments which contain a table of descriptors (Local Descriptor Table Descriptor) or segments which contain the execution state of a task (Task State Segment Descriptor).

Figure 12 gives the formats for the special system data segment descriptors. The descriptors contain a 24-bit base address of the segment and a 16-bit limit. The access byte defines the type of descriptor, its state and privilege level. The descriptor contents are valid and the segment is in physical memory if P = 1. If P = 0, the segment is not valid. The DPL field is only used in Task State Segment descriptors and indicates the privilege level at which the descriptor may be used (see Privilege). Since the Local Descriptor Table descriptor may only be used by a special privileged instruction, the DPL field is not used. Bit 4 of the access byte is 0 to indicate that it is a system control descriptor. The type field specifies the descriptor type as indicated in Figure 12.

System Segment Descriptor



210253-12

*Must be set to 0 for compatibility with 80386.

System Segment Descriptor Fields

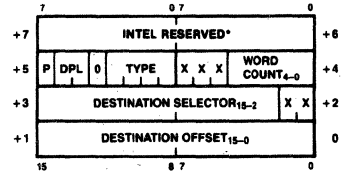
Name	Value	Description
TYPE	1 2 3	Available Task State Segment (TSS) Local Descriptor Table Busy Task State Segment (TSS)
P	0 1	Descriptor contents are not valid Descriptor contents are valid
DPL	0-3	Descriptor Privilege Level
BASE	24-bit number	Base Address of special system data segment in real memory
LIMIT	16-bit number	Offset of last byte in segment

Figure 12. System Segment Descriptor Format

GATE DESCRIPTORS (S = 0, TYPE = 4-7)

Gates are used to control access to entry points within the target code segment. The gate descriptors are call gates, task gates, interrupt gates and trap gates. Gates provide a level of indirection between the source and destination of the control transfer. This indirection allows the CPU to automatically perform protection checks and control entry point of the destination. Call gates are used to change privilege levels (see Privilege), task gates are used to perform a task switch, and interrupt and trap gates are used to specify interrupt service routines. The interrupt gate disables interrupts (resets IF) while the trap gate does not.

Gate Descriptor



210253-13

*Must be set to 0 for compatibility with 80386 (X is don't care)

Gate Descriptor Fields

Name	Value	Description
TYPE	4 5 6 7	–Call Gate –Task Gate –Interrupt Gate –Trap Gate
P	0 1	–Descriptor Contents are not valid –Descriptor Contents are valid
DPL	0–3	Descriptor Privilege Level
WORD COUNT	0–31	Number of words to copy from callers stack to called procedures stack. Only used with call gate.
DESTINATION SELECTOR	16-bit selector	Selector to the target code segment (Call, Interrupt or Trap Gate) Selector to the target task state segment (Task Gate)
DESTINATION OFFSET	16-bit offset	Entry point within the target code segment

Figure 13. Gate Descriptor Format

Figure 13 shows the format of the gate descriptors. The descriptor contains a destination pointer that points to the descriptor of the target segment and the entry point offset. The destination selector in an interrupt gate, trap gate, and call gate must refer to a code segment descriptor. These gate descriptors contain the entry point to prevent a program from constructing and using an illegal entry point. Task gates may only refer to a task state segment. Since task gates invoke a task switch, the destination offset is not used in the task gate.

Exception 13 is generated when the gate is used if a destination selector does not refer to the correct descriptor type. The word count field is used in the call gate descriptor to indicate the number of parameters (0–31 words) to be automatically copied from the caller's stack to the stack of the called routine when a control transfer changes privilege levels. The word count field is not used by any other gate descriptor.

The access byte format is the same for all gate descriptors. P = 1 indicates that the gate contents are valid. P = 0 indicates the contents are not valid and causes exception 11 if referenced. DPL is the descriptor privilege level and specifies when this descriptor may be used by a task (refer to privilege discussion below). Bit 4 must equal 0 to indicate a system control descriptor. The type field specifies the descriptor type as indicated in Figure 13.

SEGMENT DESCRIPTOR CACHE REGISTERS

A segment descriptor cache register is assigned to each of the four segment registers (CS, SS, DS, ES). Segment descriptors are automatically loaded (cached) into a segment descriptor cache register (Figure 14) whenever the associated segment register is loaded with a selector. Only segment descriptors may be loaded into segment descriptor cache registers. Once loaded, all references to that segment of memory use the cached descriptor information instead of reaccessing the descriptor. The descriptor cache registers are not visible to programs. No instructions exist to store their contents. They only change when a segment register is loaded.

SELECTOR FIELDS

A protected mode selector has three fields: descriptor entry index, local or global descriptor table indicator (TI), and selector privilege (RPL) as shown in Figure 15. These fields select one of two memory based tables of descriptors, select the appropriate table entry and allow highspeed testing of the selector's privilege attribute (refer to privilege discussion below).

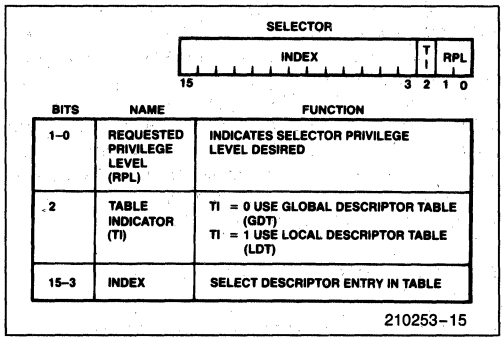


Figure 15. Selector Fields

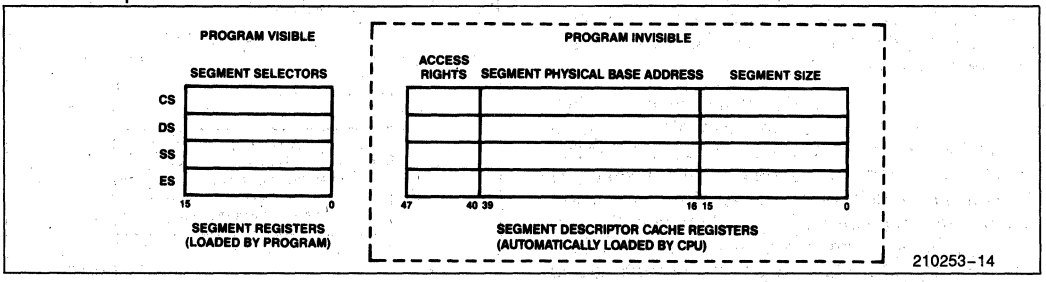


Figure 14. Descriptor Cache Registers

LOCAL AND GLOBAL DESCRIPTOR TABLES

Two tables of descriptors, called descriptor tables, contain all descriptors accessible by a task at any given time. A descriptor table is a linear array of up to 8192 descriptors. The upper 13 bits of the selector value are an index into a descriptor table. Each table has a 24-bit base register to locate the descriptor table in physical memory and a 16-bit limit register that confine descriptor access to the defined limits of the table as shown in Figure 16. A restartable exception (13) will occur if an attempt is made to reference a descriptor outside the table limits.

One table, called the Global Descriptor table (GDT), contains descriptors available to all tasks. The other table, called the Local Descriptor Table (LDT), contains descriptors that can be private to a task. Each task may have its own private LDT. The GDT may contain all descriptor types except interrupt and trap descriptors. The LDT may contain only segment, task gate, and call gate descriptors. A segment cannot be accessed by a task if its segment descriptor does not exist in either descriptor table at the time of access.

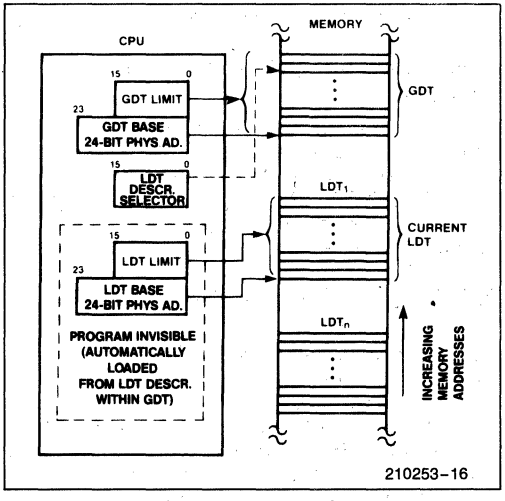
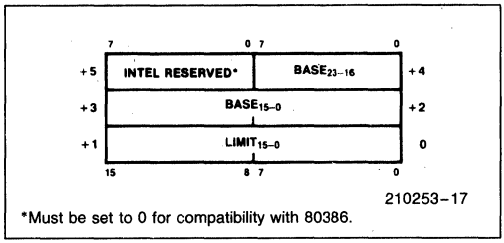


Figure 16. Local and Global Descriptor Table Definition

The LGDT and LLDT instructions load the base and limit of the global and local descriptor tables. LGDT and LLDT are privileged, i.e. they may only be executed by trusted programs operating at level 0. The LGDT instruction loads a six byte field containing the 16-bit table limit and 24-bit physical base address of the Global Descriptor Table as shown in Figure 17. The LLDT instruction loads a selector which refers to a Local Descriptor Table descriptor containing the base address and limit for an LDT, as shown in Figure 12.



*Must be set to 0 for compatibility with 80386.

Figure 17. Global Descriptor Table and Interrupt Descriptor Table Data Type

INTERRUPT DESCRIPTOR TABLE

The protected mode 80286 has a third descriptor table, called the Interrupt Descriptor Table (IDT) (see Figure 18), used to define up to 256 interrupts. It may contain only task gates, interrupt gates and trap gates. The IDT (Interrupt Descriptor Table) has a 24-bit physical base and 16-bit limit register in the CPU. The privileged LIDT instruction loads these registers with a six byte value of identical form to that of the LGDT instruction (see Figure 17 and Protected Mode Initialization).

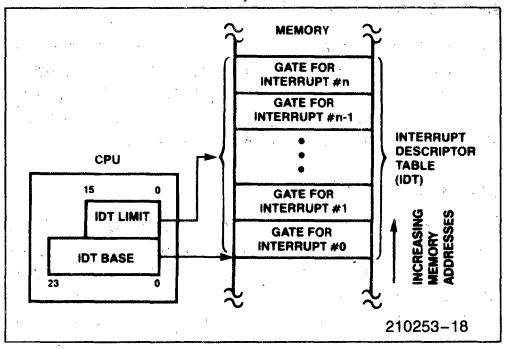
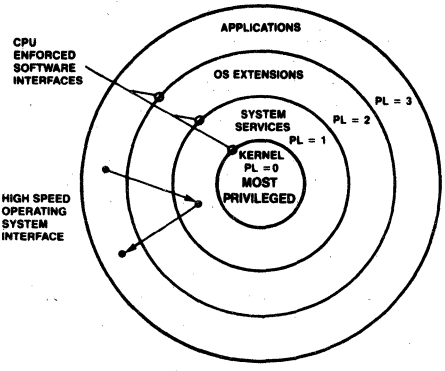


Figure 18. Interrupt Descriptor Table Definition

References to IDT entries are made via INT instructions, external interrupt vectors, or exceptions. The IDT must be at least 256 bytes in size to allocate space for all reserved interrupts.

Privilege

The 80286 has a four-level hierarchical privilege system which controls the use of privileged instructions and access to descriptors (and their associated segments) within a task. Four-level privilege, as shown in Figure 19, is an extension of the user/supervisor mode commonly found in minicomputers. The privilege levels are numbered 0 through 3. Level 0 is the



most privileged level. Privilege levels provide protection within a task. (Tasks are isolated by providing private LDT's for each task.) Operating system routines, interrupt handlers, and other system software can be included and protected within the virtual address space of each task using the four levels of privilege. Each task in the system has a separate stack for each of its privilege levels.

Tasks, descriptors, and selectors have a privilege level attribute that determines whether the descriptor may be used. Task privilege effects the use of instructions and descriptors. Descriptor and selector privilege only effect access to the descriptor.

TASK PRIVILEGE

A task always executes at one of the four privilege levels. The task privilege level at any specific instant is called the Current Privilege Level (CPL) and is defined by the lower two bits of the CS register. CPL cannot change during execution in a single code segment. A task's CPL may only be changed by control transfers through gate descriptors to a new code segment (See Control Transfer). Tasks begin executing at the CPL value specified by the code segment selector within TSS when the task is initiated via a task switch operation (See Figure 20). A task executing at Level 0 can access all data segments defined in the GDT and the task's LDT and is considered the most trusted level. A task executing a Level 3 has the most restricted access to data and is considered the least trusted level.

DESCRIPTOR PRIVILEGE

Descriptor privilege is specified by the Descriptor Privilege Level (DPL) field of the descriptor access byte. DPL specifies the least trusted task privilege level (CPL) at which a task may access the descriptor. Descriptors with DPL = 0 are the most protected. Only tasks executing at privilege level 0 (CPL = 0) may access them. Descriptors with DPL = 3 are the least protected (i.e. have the least restricted access) since tasks can access them when CPL = 0, 1, 2, or 3. This rule applies to all descriptors, except LDT descriptors.

SELECTOR PRIVILEGE

Selector privilege is specified by the Requested Privilege Level (RPL) field in the least significant two bits of a selector. Selector RPL may establish a less trusted privilege level than the current privilege level for the use of a selector. This level is called the task's effective privilege level (EPL). RPL can only reduce the scope of a task's access to data with this selector. A task's effective privilege is the numeric maximum of RPL and CPL. A selector with RPL = 0 imposes no additional restriction on its use while a selector with RPL = 3 can only refer to segments at privilege Level 3 regardless of the task's CPL. RPL is generally used to verify that pointer parameters passed to a more trusted procedure are not allowed to use data at a more privileged level than the caller (refer to pointer testing instructions).

Descriptor Access and Privilege Validation

Determining the ability of a task to access a segment involves the type of segment to be accessed, the instruction used, the type of descriptor used and CPL, RPL, and DPL. The two basic types of segment accesses are control transfer (selectors loaded into CS) and data (selectors loaded into DS, ES or SS).

DATA SEGMENT ACCESS

Instructions that load selectors into DS and ES must refer to a data segment descriptor or readable code segment descriptor. The CPL of the task and the RPL of the selector must be the same as or more privileged (numerically equal to or lower than) than the descriptor DPL. In general, a task can only access data segments at the same or less privileged levels than the CPL or RPL (whichever is numerically higher) to prevent a program from accessing data it cannot be trusted to use.

An exception to the rule is a readable conforming code segment. This type of code segment can be read from any privilege level.

If the privilege checks fail (e.g. DPL is numerically less than the maximum of CPL and RPL) or an incorrect type of descriptor is referenced (e.g. gate de-

scriptor or execute only code segment) exception 13 occurs. If the segment is not present, exception 11 is generated.

Instructions that load selectors into SS must refer to data segment descriptors for writable data segments. The descriptor privilege (DPL) and RPL must equal CPL. All other descriptor types or a privilege level violation will cause exception 13. A not present fault causes exception 12.

CONTROL TRANSFER

Four types of control transfer can occur when a selector is loaded into CS by a control transfer operation (see Table 10). Each transfer type can only occur if the operation which loaded the selector references the correct descriptor type. Any violation of these descriptor usage rules (e.g. JMP through a call gate or RET to a Task State Segment) will cause exception 13.

The ability to reference a descriptor for control transfer is also subject to rules of privilege. A CALL or JUMP instruction may only reference a code segment descriptor with DPL equal to the task CPL or a conforming segment with DPL of equal or greater privilege than CPL. The RPL of the selector used to reference the code descriptor must have as much privilege as CPL.

RET and IRET instructions may only reference code segment descriptors with descriptor privilege equal to or less privileged than the task CPL. The selector loaded into CS is the return address from the stack. After the return, the selector RPL is the task's new CPL. If CPL changes, the old stack pointer is popped after the return address.

When a JMP or CALL references a Task State Segment descriptor, the descriptor DPL must be the same or less privileged than the task's CPL. Reference to a valid Task State Segment descriptor causes a task switch (see Task Switch Operation). Reference to a Task State Segment descriptor at a more privileged level than the task's CPL generates exception 13.

When an instruction or interrupt references a gate descriptor, the gate DPL must have the same or less privilege than the task CPL. If DPL is at a more privileged level than CPL, exeception 13 occurs. If the destination selector contained in the gate references a code segment descriptor, the code segment descriptor DPL must be the same or more privileged than the task CPL. If not, Exception 13 is issued. After the control transfer, the code segment descriptors DPL is the task's new CPL. If the destination selector in the gate references a task state segment, a task switch is automatically performed (see Task Switch Operation).

The privilege rules on control transfer require:

- JMP or CALL direct to a code segment (code segment descriptor) can only be to a conforming segment with DPL of equal or greater privilege than CPL or a non-conforming segment at the same privilege level.
- interrupts within the task or calls that may change privilege levels, can only transfer control through a gate at the same or a less privileged level than CPL to a code segment at the same or more privileged level than CPL.
- return instructions that don't switch tasks can only return control to a code segment at the same or less privileged level.
- task switch can be performed by a call, jump or interrupt which references either a task gate or task state segment at the same or less privileged level.

Table 10. Descriptor Types Used for Control Transfer

Control Transfer Types	Operation Types	Descriptor Referenced	Descriptor Table
Intersegment within the same privilege level	JMP, CALL, RET, IRET*	Code Segment	GDT/LDT
Intersegment to the same or higher privilege level Interrupt within task may change CPL.	CALL	Call Gate	GDT/LDT
	Interrupt Instruction, Exception, External Interrupt	Trap or Interrupt Gate	IDT
Intersegment to a lower privilege level (changes task CPL)	RET, IRET*	Code Segment	GDT/LDT
	CALL, JMP	Task State Segment	GDT
Task Switch	CALL, JMP	Task Gate	GDT/LDT
	IRET** Interrupt Instruction, Exception, External Interrupt	Task Gate	IDT

*NT (Nested Task bit of flag word) = 0
**NT (Nested Task bit of flag word) = 1

PRIVILEGE LEVEL CHANGES

Any control transfer that changes CPL within the task, causes a change of stacks as part of the operation. Initial values of SS:SP for privilege levels 0, 1, and 2 are kept in the task state segment (refer to Task Switch Operation). During a JMP or CALL control transfer, the new stack pointer is loaded into the SS and SP registers and the previous stack pointer is pushed onto the new stack.

When returning to the original privilege level, its stack is restored as part of the RET or IRET instruction operation. For subroutine calls that pass parameters on the stack and cross privilege levels, a fixed number of words, as specified in the gate, are copied from the previous stack to the current stack. The inter-segment RET instruction with a stack adjustment value will correctly restore the previous stack pointer upon return.

Protection

The 80286 includes mechanisms to protect critical instructions that affect the CPU execution state (e.g. HLT) and code or data segments from improper usage. These protection mechanisms are grouped into three forms:

Restricted *usage* of segments (e.g. no write allowed to read-only data segments). The only segments available for use are defined by descriptors in the Local Descriptor Table (LDT) and Global Descriptor Table (GDT).

Restricted *access* to segments via the rules of privilege and descriptor usage.

Privileged instructions or operations that may only be executed at certain privilege levels as determined by the CPL and I/O Privilege Level (IOPL). The IOPL is defined by bits 14 and 13 of the flag word.

These checks are performed for all instructions and can be split into three categories: segment load checks (Table 11), operand reference checks (Table 12), and privileged instruction checks (Table 13). Any violation of the rules shown will result in an exception. A not-present exception related to the stack segment causes exception 12.

The IRET and POPF instructions do not perform some of their defined functions if CPL is not of sufficient privilege (numerically small enough). Precisely these are:

- The IF bit is not changed if CPL > IOPL.
- The IOPL field of the flag word is not changed if CPL > 0.

No exceptions or other indication are given when these conditions occur.

Table 11
Segment Register Load Checks

Error Description	Exception Number
Descriptor table limit exceeded	13
Segment descriptor not-present	11 or 12
Privilege rules violated	13
Invalid descriptor/segment type segment register load: —Read only data segment load to SS —Special Control descriptor load to DS, ES, SS —Execute only segment load to DS, ES, SS —Data segment load to CS —Read/Execute code segment load to SS	13

Table 12. Operand Reference Checks

Error Description	Exception Number
Write into code segment	13
Read from execute-only code segment	13
Write to read-only data segment	13
Segment limit exceeded[1]	12 or 13

NOTE:
Carry out in offset calculations is ignored.

Table 13. Privileged Instruction Checks

Error Description	Exception Number
CPL ≠ 0 when executing the following instructions: LIDT, LLDT, LGDT, LTR, LMSW, CTS, HLT	13
CPL > IOPL when executing the following instructions: INS, IN, OUTS, OUT, STI, CLI, LOCK	13

EXCEPTIONS

The 80286 detects several types of exceptions and interrupts, in protected mode (see Table 14). Most are restartable after the exceptional condition is removed. Interrupt handlers for most exceptions can read an error code, pushed on the stack after the return address, that identifies the selector involved (0 if none). The return address normally points to the failing instruction, including all leading prefixes. For a processor extension segment overrun exception, the return address will not point at the ESC instruction that caused the exception; however, the processor extension registers may contain the address of the failing instruction.

Table 14. Protected Mode Exceptions

Interrupt Vector	Function	Return Address At Failing Instruction?	Always Restartable?	Error Code on Stack?
8	Double exception detected	Yes	No[2]	Yes
9	Processor extension segment overrun	No	No[2]	No
10	Invalid task state segment	Yes	Yes	Yes
11	Segment not present	Yes	Yes	Yes
12	Stack segment overrun or stack segment not present	Yes	Yes[1]	Yes
13	General protection	Yes	No[2]	Yes

NOTE:
1. When a PUSHA or POPA instruction attempts to wrap around the stack segment, the machine state after the exception will not be restartable because stack segment wrap around is not permitted. This condition is identified by the value of the saved SP being either 0000(H), 0001(H), FFFE(H), or FFFF(H).
2. These exceptions indicate a violation to privilege rules or usage rules has occurred. Restart is generally not attempted under those conditions.

These exceptions indicate a violation to privilege rules or usage rules has occurred. Restart is generally not attempted under those conditions.

All these checks are performed for all instructions and can be split into three categories: segment load checks (Table 11), operand reference checks (Table 12), and privileged instruction checks (Table 13). Any violation of the rules shown will result in an exception. A not-present exception causes exception 11 or 12 and is restartable.

Special Operations

TASK SWITCH OPERATION

The 80286 provides a built-in task switch operation which saves the entire 80286 execution state (registers, address space, and a link to the previous task), loads a new execution state, and commences execution in the new task. Like gates, the task switch operation is invoked by executing an inter-segment JMP or CALL instruction which refers to a Task State Segment (TSS) or task gate descriptor in the GDT or LDT. An INT n instruction, exception, or external interrupt may also invoke the task switch operation by selecting a task gate descriptor in the associated IDT descriptor entry.

The TSS descriptor points at a segment (see Figure 20) containing the entire 80286 execution state while a task gate descriptor contains a TSS selector. The limit field of the descriptor must be >002B(H).

Each task must have a TSS associated with it. The current TSS is identified by a special register in the 80286 called the Task Register (TR). This register contains a selector referring to the task state segment descriptor that defines the current TSS. A hidden base and limit register associated with TR are loaded whenever TR is loaded with a new selector.

The IRET instruction is used to return control to the task that called the current task or was interrupted. Bit 14 in the flag register is called the Nested Task (NT) bit. It controls the function of the IRET instruction. If NT = 0, the IRET instruction performs the regular current task by popping values off the stack; when NT = 1, IRET performs a task switch operation back to the previous task.

When a CALL, JMP, or INT instruction initiates a task switch, the old (except for case of JMP) and new TSS will be marked busy and the back link field of the new TSS set to the old TSS selector. The NT bit of the new task is set by CALL or INT initiated task switches. An interrupt that does not cause a task switch will clear NT. NT may also be set or cleared by POPF or IRET instructions.

The task state segment is marked busy by changing the descriptor type field from Type 1 to Type 3. Use of a selector that references a busy task state segment causes Exception 13.

PROCESSOR EXTENSION CONTEXT SWITCHING

The context of a processor extension (such as the 80287 numerics processor) is not changed by the task switch operation. A processor extension context need only be changed when a different task attempts to use the processor extension (which still contains the context of a previous task). The 80286 detects the first use of a processor extension after a task switch by causing the processor extension not present exception (7). The interrupt handler may then decide whether a context change is necessary.

Whenever the 80286 switches tasks, it sets the Task Switched (TS) bit of the MSW. TS indicates that a processor extension context may belong to a different task than the current one. The processor extension not present exception (7) will occur when attempting to execute an ESC or WAIT instruction if TS = 1 and a processor extension is present (MP = 1 in MSW).

POINTER TESTING INSTRUCTIONS

The 80286 provides several instructions to speed pointer testing and consistency checks for maintaining system integrity (see Table 15). These instructions use the memory management hardware to verify that a selector value refers to an appropriate segment without risking an exception. A condition flag (ZF) indicates whether use of the selector or segment will cause an exception.

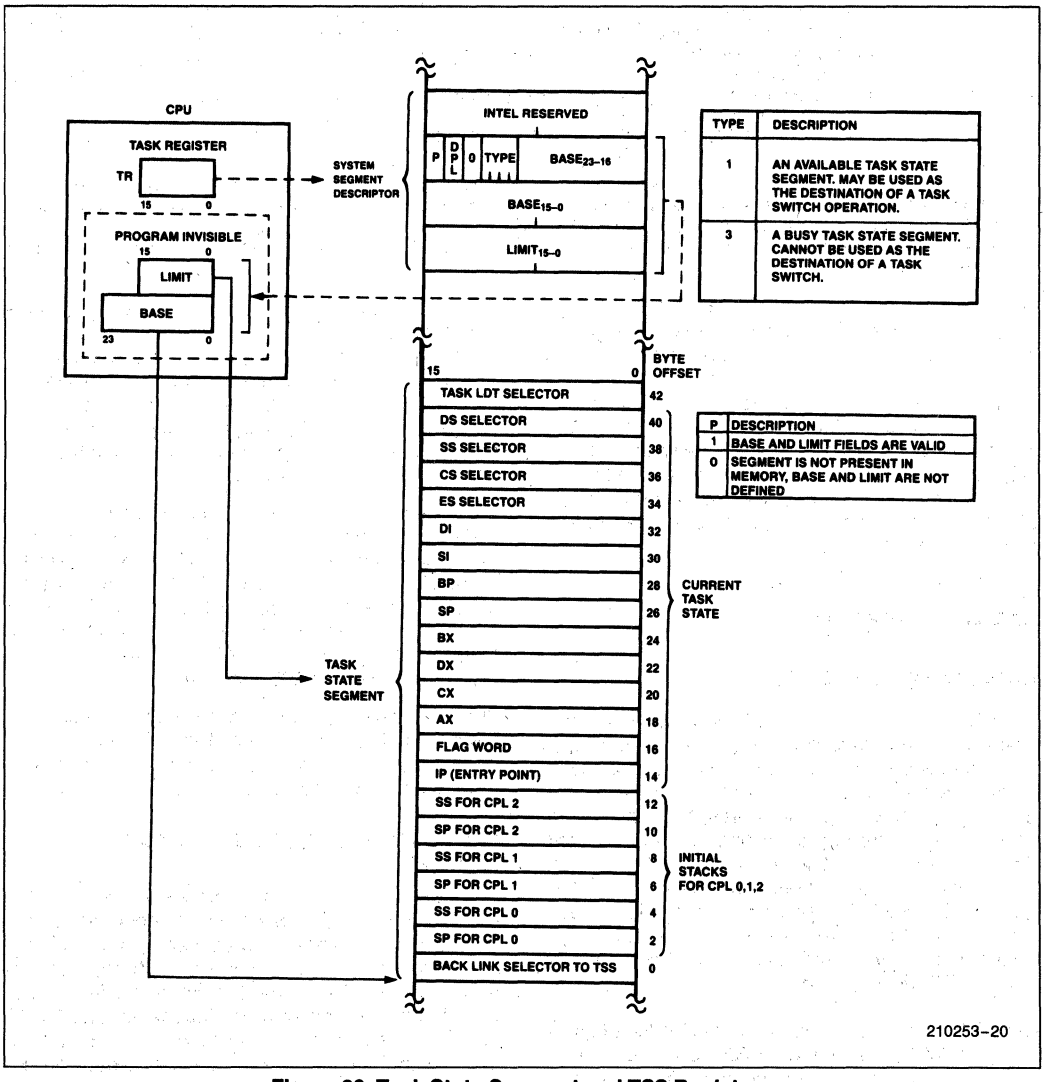


Figure 20. Task State Segment and TSS Registers

Table 15. 80286 Pointer Test Instructions

Instruction	Operands	Function
ARPL	Selector, Register	Adjust Requested Privilege Level: adjusts the RPL of the selector to the numeric maximum of current selector RPL value and the RPL value in the register. Set zero flag if selector RPL was changed by ARPL.
VERR	Selector	VERify for Read: sets the zero flag if the segment referred to by the selector can be read.
VERW	Selector	VERify for Write: sets the zero flag if the segment referred to by the selector can be written.
LSL	Register, Selector	Load Segment Limit: reads the segment limit into the register if privilege rules and descriptor type allow. Set zero flag if successful.
LAR	Register, Selector	Load Access Rights: reads the descriptor access rights byte into the register if privilege rules allow. Set zero flag if successful.

DOUBLE FAULT AND SHUTDOWN

If two separate exceptions are detected during a single instruction execution, the 80286 performs the double fault exception (8). If an execution occurs during processing of the double fault exception, the 80286 will enter shutdown. During shutdown no further instructions or exceptions are processed. Either NMI (CPU remains in protected mode) or RESET (CPU exits protected mode) can force the 80286 out of shutdown. Shutdown is externally signalled via a HALT bus operation with A_1 LOW.

PROTECTED MODE INITIALIZATION

The 80286 initially executes in real address mode after RESET. To allow initialization code to be placed at the top of physical memory, A_{23}–A_{20} will be HIGH when the 80286 performs memory references relative to the CS register until CS is changed. A_{23}–A_{20} will be zero for references to the DS, ES, or SS segments. Changing CS in real address mode will force A_{23}–A_{20} LOW whenever CS is used again. The initial CS:IP value of F000:FFF0 provides 64K bytes of code space for initialization code without changing CS.

Protected mode operation requires several registers to be initialized. The GDT and IDT base registers must refer to a valid GDT and IDT. After executing the LMSW instruction to set PE, the 80286 must immediately execute an intra-segment JMP instruction to clear the instruction queue of instructions decoded in real address mode.

To force the 80286 CPU registers to match the initial protected mode state assumed by software, execute a JMP instruction with a selector referring to the initial TSS used in the system. This will load the task register, local descriptor table register, segment registers and initial general register state. The TR should point at a valid TSS since any task switch operation involves saving the current task state.

SYSTEM INTERFACE

The 80286 system interface appears in two forms: a local bus and a system bus. The local bus consists of address, data, status, and control signals at the pins of the CPU. A system bus is any buffered version of the local bus. A system bus may also differ from the local bus in terms of coding of status and control lines and/or timing and loading of signals. The 80286 family includes several devices to generate standard system buses such as the IEEE 796 standard MULTIBUS.

Bus Interface Signals and Timing

The 80286 microsystem local bus interfaces the 80286 to local memory and I/O components. The interface has 24 address lines, 16 data lines, and 8 status and control signals.

The 80286 CPU, 82C284 clock generator, 82288 bus controller, 82289 bus arbiter, tranceivers, and latches provide a buffered and decoded system bus interface. The 82C284 generates the system clock and synchronizes $\overline{\text{READY}}$ and RESET. The 82288 converts bus operation status encoded by the 80286 into command and bus control signals. The 82289 bus arbiter generates Multibus bus arbitration signals. These components can provide the timing and electrical power drive levels required for most system bus interfaces including the Multibus.

Physical Memory and I/O Interface

A maximum of 16 megabytes of physical memory can be addressed in protected mode. One megabyte can be addressed in real address mode. Memory is accessible as bytes or words. Words consist of any two consecutive bytes addressed with the least significant byte stored in the lowest address.

Byte transfers occur on either half of the 16-bit local data bus. Even bytes are accessed over D_{7-0} while odd bytes are transferred over D_{15-8}. Even-addressed words are transferred over D_{15-0} in one bus cycle, while odd-addressed word require *two* bus operations. The first transfers data on D_{15-8}, and the second transfers data on D_{7-0}. Both byte data transfers occur automatically, transparent to software.

Two bus signals, A_0 and $\overline{BHE}$, control transfers over the lower and upper halves of the data bus. Even address byte transfers are indicated by A_0 LOW and $\overline{BHE}$ HIGH. Odd address byte transfers are indicated by A_0 HIGH and $\overline{BHE}$ LOW. Both A_0 and $\overline{BHE}$ are LOW for even address word transfers.

The I/O address space contains 64K addresses in both modes. The I/O space is accessible as either bytes or words, as is memory. Byte wide peripheral devices may be attached to either the upper or lower byte of the data bus. Byte-wide I/O devices attached to the upper data byte (D_{15-8}) are accessed with odd I/O addresses. Devices on the lower data byte are accessed with even I/O addresses. An interrupt controller such as Intel's 8259A must be connected to the lower data byte (D_{7-0}) for proper return of the interrupt vector.

Bus Operation

The 80286 uses a double frequency system clock (CLK input) to control bus timing. All signals on the local bus are measured relative to the system CLK input. The CPU divides the system clock by 2 to produce the internal processor clock, which determines bus state. Each processor clock is composed of two system clock cycles named phase 1 and phase 2. The 82C284 clock generator output (PCLK) identifies the next phase of the processor clock. (See Figure 21.)

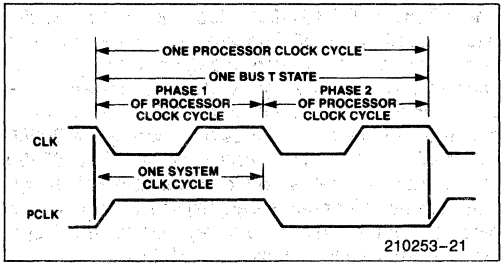


Figure 21. System and Processor Clock Relationships

Six types of bus operations are supported; memory read, memory write, I/O read, I/O write, interrupt acknowledge, and halt/shutdown. Data can be transferred at a maximum rate of one word per two processor clock cycles.

The 80286 bus has three basic states: idle (T_i), send status (T_s), and perform command (T_c). The 80286 CPU also has a fourth local bus state called hold (T_h). T_h indicates that the 80286 has surrendered control of the local bus to another bus master in response to a HOLD request.

Each bus state is one processor clock long. Figure 22 shows the four 80286 local bus states and allowed transitions.

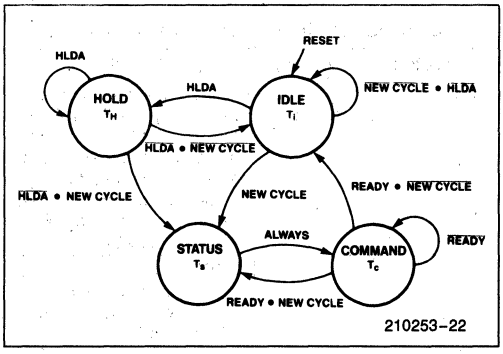


Figure 22. 80286 Bus States

Bus States

The idle (T_i) state indicates that no data transfers are in progress or requested. The first active state T_S is signaled by status line $\overline{S1}$ or $\overline{S0}$ going LOW and identifying phase 1 of the processor clock. During T_S, the command encoding, the address, and data (for a write operation) are available on the 80286 output pins. The 82288 bus controller decodes the status signals and generates Multibus compatible read/write command and local transceiver control signals.

After T_S, the perform command (T_C) state is entered. Memory or I/O devices respond to the bus operation during T_C, either transferring read data to the CPU or accepting write data. T_C states may be repeated as often as necessary to assure sufficient time for the memory or I/O device to respond. The $\overline{READY}$ signal determines whether T_C is repeated. A repeated T_C state is called a wait state.

During hold (T_h), the 80286 will float all address, data, and status output pins enabling another bus master to use the local bus. The 80286 HOLD input signal is used to place the 80286 into the T_h state. The 80286 HLDA output signal indicates that the CPU has entered T_h.

Pipelined Addressing

The 80286 uses a local bus interface with pipelined timing to allow as much time as possible for data access. Pipelined timing allows a new bus operation to be initiated every two processor cycles, while allowing each individual bus operation to last for three processor cycles.

The timing of the address outputs is pipelined such that the address of the next bus operation becomes available during the current bus operation. Or in other words, the first clock of the next bus operation is overlapped with the last clock of the current bus operation. Therefore, address decode and routing logic can operate in advance of the next bus operation.

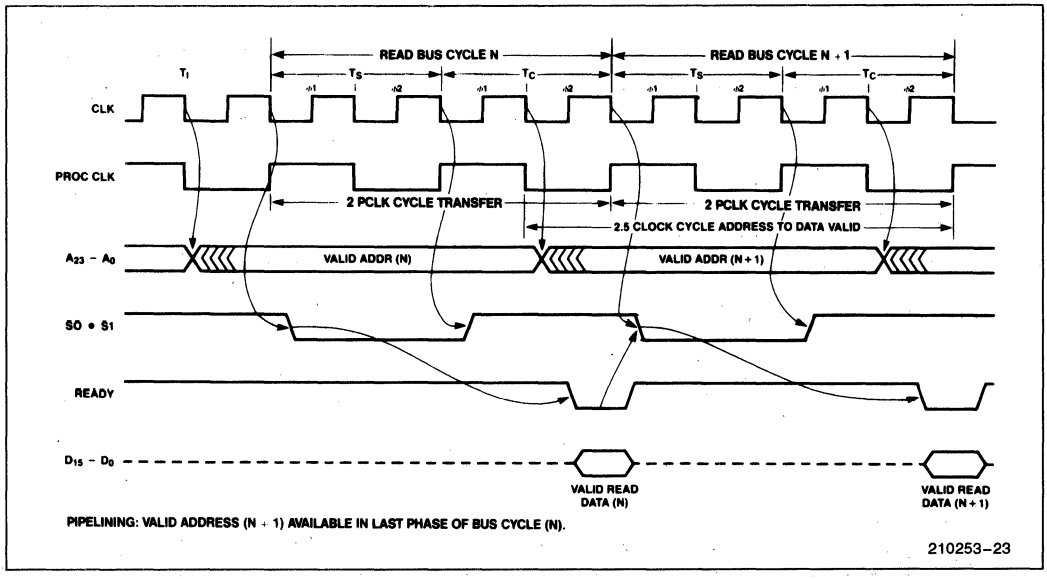


Figure 23. Basic Bus Cycle

External address latches may hold the address stable for the entire bus operation, and provide additional AC and DC buffering.

The 80286 does not maintain the address of the current bus operation during all T_C states. Instead, the address for the next bus operation may be emitted during phase 2 of any T_C. The address remains valid during phase 1 of the first T_C to guarantee hold time, relative to ALE, for the address latch inputs.

Bus Control Signals

The 82288 bus controller provides control signals; address latch enable (ALE), Read/Write commands, data transmit/receive (DT/$\overline{R}$), and data enable (DEN) that control the address latches, data transceivers, write enable, and output enable for memory and I/O systems.

The Address Latch Enable (ALE) output determines when the address may be latched. ALE provides at least one system CLK period of address hold time from the end of the previous bus operation until the address for the next bus operation appears at the latch outputs. This address hold time is required to support MULTIBUS and common memory systems.

The data bus transceivers are controlled by 82288 outputs Data Enable (DEN) and Data Transmit/Receive (DT/$\overline{R}$). DEN enables the data transceivers; while DT/$\overline{R}$ controls tranceiver direction. DEN and DT/$\overline{R}$ are timed to prevent bus contention between the bus master, data bus transceivers, and system data bus transceivers.

Command Timing Controls

Two system timing customization options, command extension and command delay, are provided on the 80286 local bus.

Command extension allows additional time for external devices to respond to a command and is analogous to inserting wait states on the 8086. External logic can control the duration of any bus operation such that the operation is only as long as necessary. The $\overline{READY}$ input signal can extend any bus operation for as long as necessary.

Command delay allows an increase of address or write data setup time to system bus command active for any bus operation by delaying when the system bus command becomes active. Command delay is controlled by the 82288 CMDLY input. After T_S, the bus controller samples CMDLY at each failing edge of CLK. If CMDLY is HIGH, the 82288 will not activate the command signal. When CMDLY is LOW, the 82288 will activate the command signal. After the command becomes active, the CMDLY input is not sampled.

When a command is delayed, the available response time from command active to return read data or accept write data is less. To customize system bus timing, an address decoder can determine which bus operations require delaying the command. The CMDLY input does not affect the timing of ALE, DEN, or DT/$\overline{R}$.

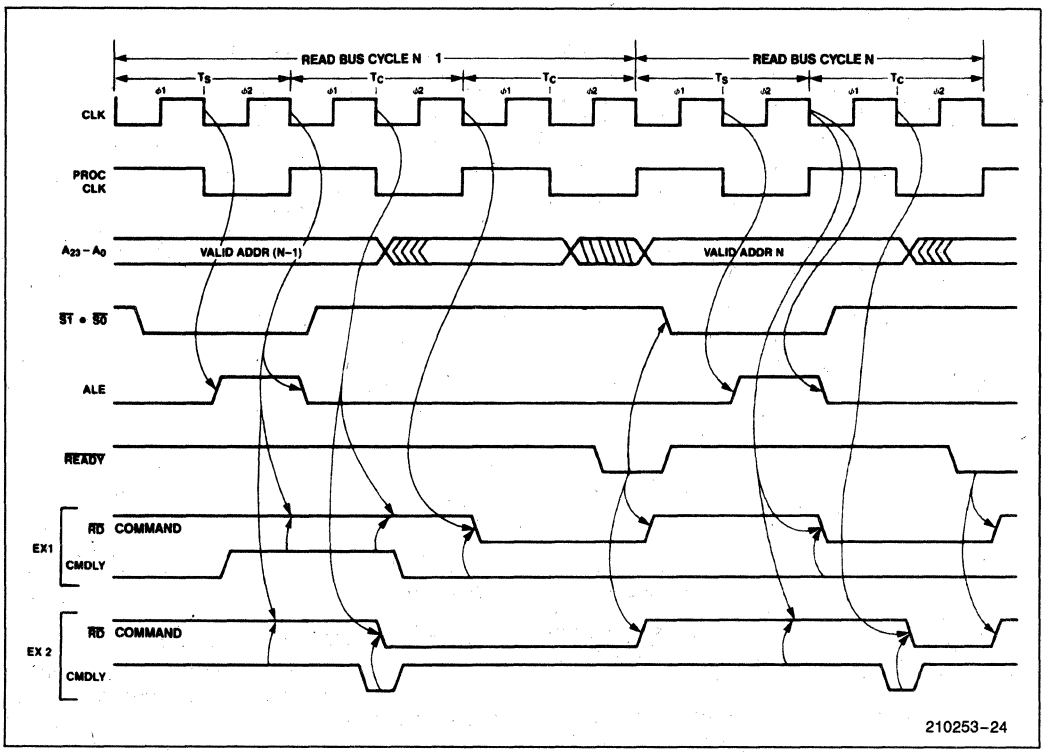


Figure 24. CMDLY Controls the Leading Edge of Command Signal

Figure 24 illustrates four uses of CMDLY. Example 1 shows delaying the read command two system CLKs for cycle N-1 and no delay for cycle N, and example 2 shows delaying the read command one system CLK for cycle N-1 and one system CLK delay for cycle N.

Bus Cycle Termination

At maximum transfer rates, the 80286 bus alternates between the status and command states. The bus status signals become inactive after T_S so that they may correctly signal the start of the next bus operation after the completion of the current cycle. No external indication of T_C exists on the 80286 local bus. The bus master and bus controller enter T_C directly after T_S and continue executing T_C cycles until terminated by $\overline{READY}$.

$\overline{READY}$ Operation

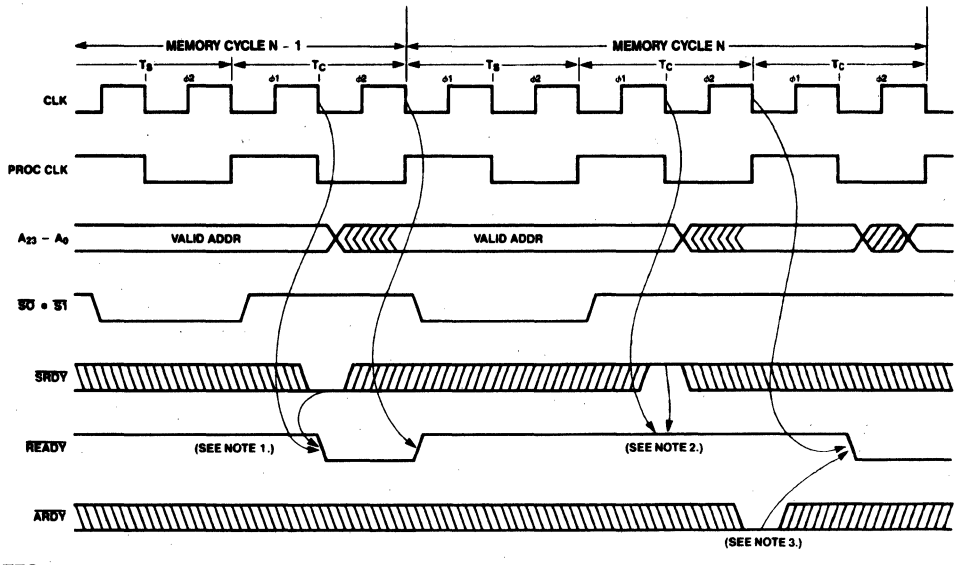
The current bus master and 82288 bus controller terminate each bus operation simultaneously to achieve maximum bus operation bandwidth. Both are informed in advance by $\overline{READY}$ active (open-collector output from 82C284) which identifies the last T_C cycle of the current bus operation. The bus master and bus controller must see the same sense of the $\overline{READY}$ signal, thereby requiring $\overline{READY}$ be synchronous to the system clock.

Synchronous Ready

The 82C284 clock generator provides $\overline{READY}$ synchronization from both synchronous and asynchronous sources (see Figure 25). The synchronous ready input ($\overline{SRDY}$) of the clock generator is sampled with the falling edge of CLK at the end of phase 1 of each T_C. The state of $\overline{SRDY}$ is then broadcast to the bus master and bus controller via the $\overline{READY}$ output line.

Asynchronous Ready

Many systems have devices or subsystems that are asynchronous to the system clock. As a result, their ready outputs cannot be guaranteed to meet the 82C284 $\overline{SRDY}$ setup and hold time requirements. But the 82C284 asynchronous ready input ($\overline{ARDY}$) is designed to accept such signals. The $\overline{ARDY}$ input is sampled at the beginning of each T_C cycle by 82C284 synchronization logic. This provides one system CLK cycle time to resolve its value before broadcasting it to the bus master and bus controller.



NOTES:
1. $\overline{\text{SRDYEN}}$ is active low.
2. If $\overline{\text{SRDYEN}}$ is high, the state of $\overline{\text{SRDY}}$ will no affect $\overline{\text{READY}}$.
3. $\overline{\text{ARDYEN}}$ is active low.

210253-25

Figure 25. Synchronous and Asynchronous Ready

$\overline{\text{ARDY}}$ or $\overline{\text{ARDYEN}}$ must be HIGH at the end of T_S. $\overline{\text{ARDY}}$ cannot be used to terminate bus cycle with no wait states.

Each ready input of the 82C284 has an enable pin ($\overline{\text{SRDYEN}}$ and $\overline{\text{ARDYEN}}$) to select whether the current bus operation will be terminated by the synchronous or asynchronous ready. Either of the ready inputs may terminate a bus operation. These enable inputs are active low and have the same timing as their respective ready inputs. Address decode logic usually selects whether the current bus operation should be terminated by $\overline{\text{ARDY}}$ or $\overline{\text{SRDY}}$.

Data Bus Control

Figures 26, 27, and 28 show how the DT/$\overline{\text{R}}$, DEN, data bus, and address signals operate for different combinations of read, write, and idle bus operations. DT/$\overline{\text{R}}$ goes active (LOW) for a read operation. DT/$\overline{\text{R}}$ remains HIGH before, during, and between write operations.

The data bus is driven with write data during the second phase of T_S. The delay in write data timing allows the read data drivers, from a previous read cycle, sufficient time to enter 3-state OFF before the 80286 CPU begins driving the local data bus for write operations. Write data will always remain valid for one system clock past the last T_C to provide sufficient hold time for Multibus or other similar memory or I/O systems. During write-read or write-idle sequences the data bus enters 3-state OFF during the second phase of the processor cycle after the last T_C. In a write-write sequence the data bus does not enter 3-state OFF between T_C and T_S.

Bus Usage

The 80286 local bus may be used for several functions: instruction data transfers, data transfers by other bus masters, instruction fetching, processor extension data transfers, interrupt acknowledge, and halt/shutdown. This section describes local bus activities which have special signals or requirements.

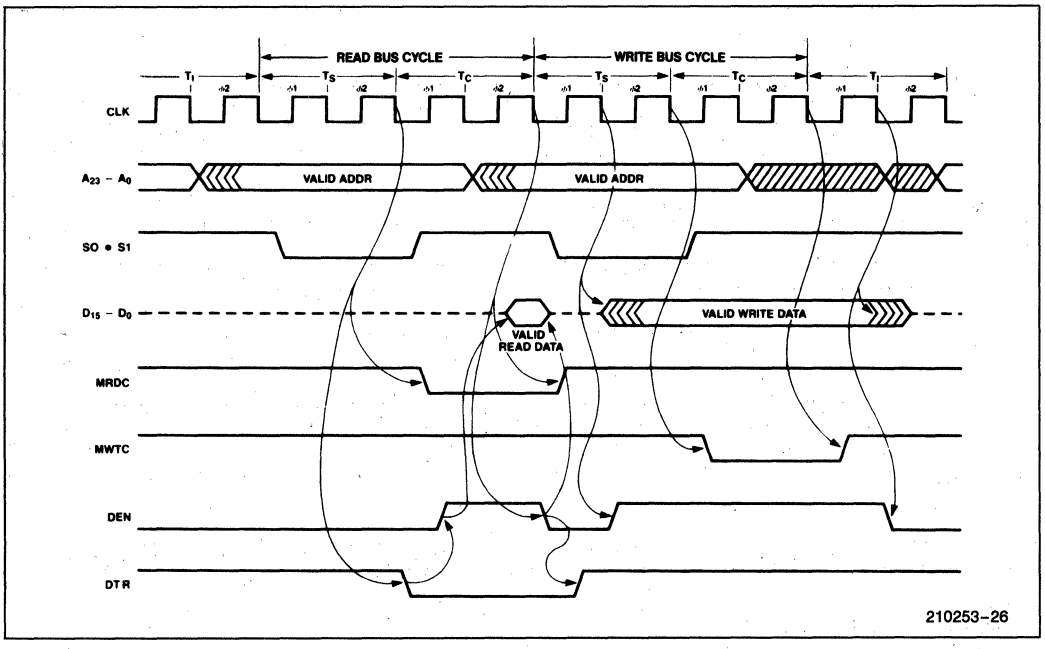


Figure 26. Back to Back Read-Write Cycles

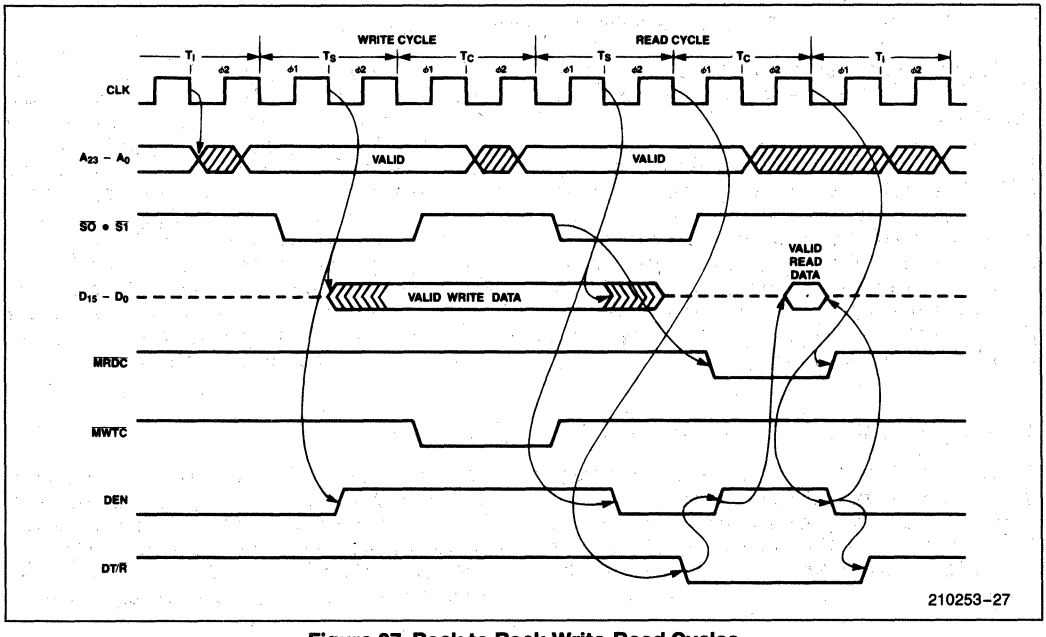


Figure 27. Back to Back Write-Read Cycles

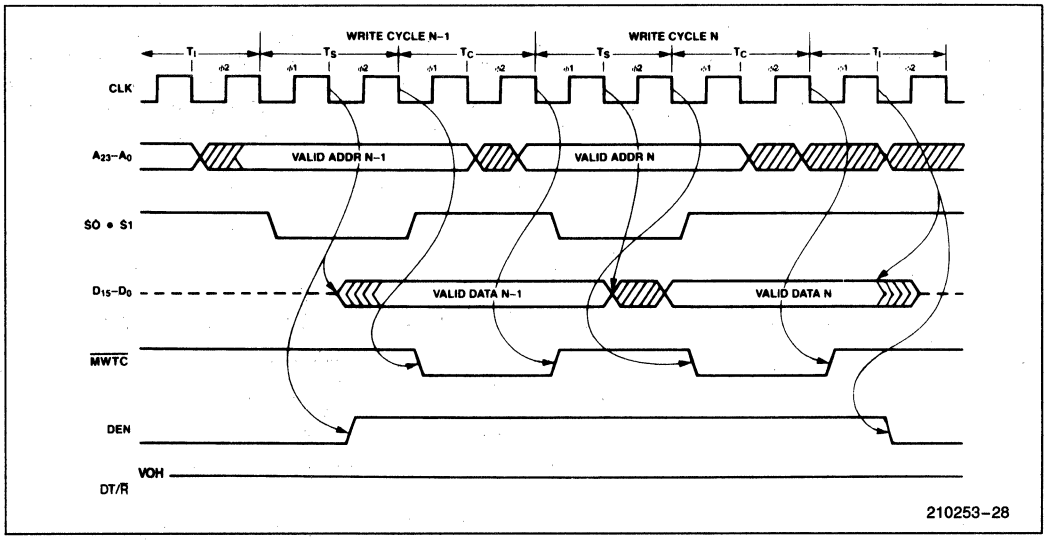


Figure 28. Back to Back Write-Write Cycles

HOLD and HLDA

HOLD AND HLDA allow another bus master to gain control of the local bus by placing the 80286 bus into the T_h state. The sequence of events required to pass control between the 80286 and another local bus master are shown in Figure 29.

In this example, the 80286 is initially in the T_h state as signaled by HLDA being active. Upon leaving T_h, as signaled by HLDA going inactive, a write operation is started. During the write operation another local bus master requests the local bus from the 80286 as shown by the HOLD signal. After completing the write operation, the 80286 performs one T_i bus cycle, to guarantee write data hold time, then enters T_h as signaled by HLDA going active.

The CMDLY signal and $\overline{\text{ARDY}}$ ready are used to start and stop the write bus command, respectively. Note that $\overline{\text{SRDY}}$ must be inactive or disabled by $\overline{\text{SRDYEN}}$ to guarantee $\overline{\text{ARDY}}$ will terminate the cycle.

HOLD must not be active during the time from the leading edge of RESET until 34 CLKs following the trailing edge of RESET.

Lock

The CPU asserts an active lock signal during Interrupt-Acknowledge cycles, the XCHG instruction, and during some descriptor accesses. Lock is also asserted when the LOCK prefix is used. The LOCK prefix may be used with the following ASM-286 assembly instructions; MOVS, INS, and OUTS. For bus cycles other than Interrupt-Acknowledge cycles, Lock will be active for the first and subsequent cycles of a series of cycles to be locked. Lock will not be shown active during the last cycle to be locked. For the next-to-last cycle, Lock will become inactive at the end of the first T_c regardless of the number of wait-states inserted. For Interrupt-Acknowledge cycles, Lock will be active for each cycle, and will become inactive at the end of the first T_c for each cycle regardless of the number of wait-states inserted.

Instruction Fetching

The 80286 Bus Unit (BU) will fetch instructions ahead of the current instruction being executed. This activity is called prefetching. It occurs when the local bus would otherwise be idle and obeys the following rules:

A prefetch bus operation starts when at least two bytes of the 6-byte prefetch queue are empty.

The prefetcher normally performs word prefetches independent of the byte alignment of the code segment base in physical memory.

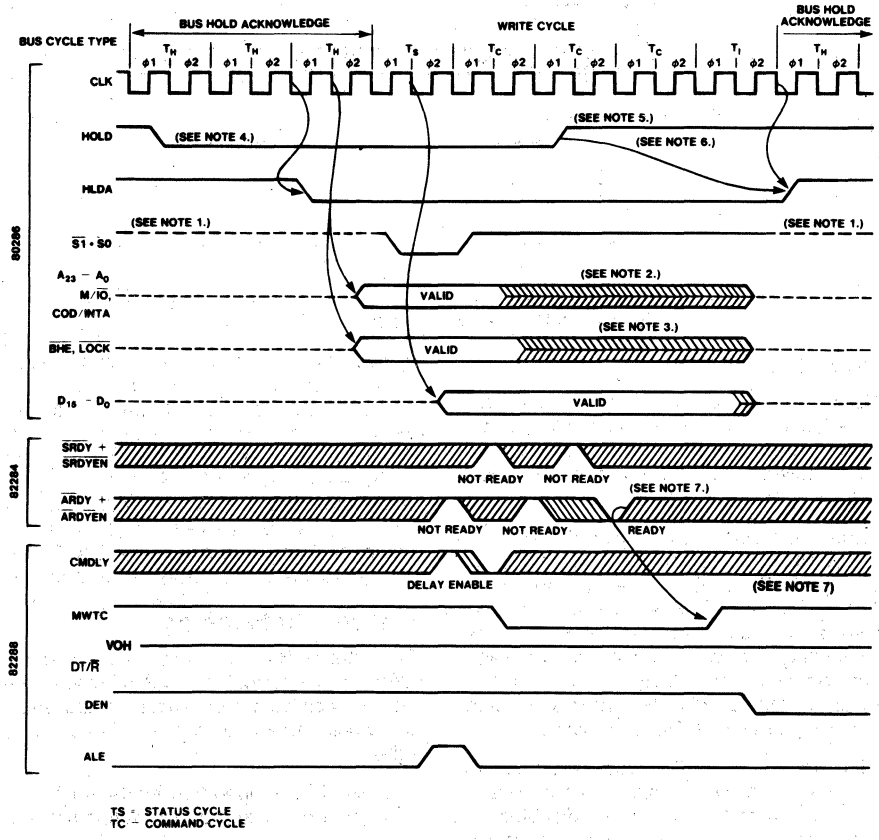
The prefetcher will perform only a byte code fetch operation for control transfers to an instruction beginning on a numerically odd physical address.

Prefetching stops whenever a control transfer or HLT instruction is decoded by the IU and placed into the instruction queue.

In real address mode, the prefetcher may fetch up to 6 bytes beyond the last control transfer or HLT instruction in a code segment.

In protected mode, the prefetcher will never cause a segment overrun exception. The prefetcher stops at the last physical memory word of the code segment. Exception 13 will occur if the program attempts to execute beyond the last full instruction in the code segment.

If the last byte of a code segment appears on an even physical memory address, the prefetcher will read the next physical byte of memory (perform a word code fetch). The value of this byte is ignored and any attempt to execute it causes exception 13.



NOTES:
1. Status lines are not driven by 80286, yet remain high due to pullup resistors in 82288 and 82289 during HOLD state.
2. Address, M/$\overline{IO}$ and COD/$\overline{INTA}$ may start floating during any T_C depending on when internal 80286 bus arbiter decides to release bus to external HOLD. The float starts in $\phi 2$ of T_C.
3. $\overline{BHE}$ and $\overline{LOCK}$ may start floating after the end of any T_C depending on when internal 80286 bus arbiter decides to release bus to external HOLD. The float starts in $\phi 1$ of T_C.
4. The minimum HOLD to HLDA time is shown. Maximum is one T_H longer.
5. The earliest HOLD time is shown. It will always allow a subsequent memory cycle if pending is shown.
6. The minimum HOLD to HLDA time is shown. Maximum is a function of the instruction, type of bus cycle and other machine state (i.e., Interrupts, Waits, Lock, etc.).
7. Asynchronous ready allows termination of the cycle. Synchronous ready does not signal ready in this example. Synchronous ready state is ignored after ready is signaled via the asynchronous input.

Figure 29. MULTIBUS® Write Terminated by Asynchronous Ready with Bus Hold

Processor Extension Transfers

The processor extension interface uses I/O port addresses 00F8(H), 00FA(H), and 00FC(H) which are part of the I/O port address range reserved by Intel. An ESC instruction with Machine Status Word bits EM = 0 and TS = 0 will perform I/O bus operations to one or more of these I/O port addresses independent of the value of IOPL and CPL.

ESC instructions with memory references enable the CPU to accept PEREQ inputs for processor extension operand transfers. The CPU will determine the operand starting address and read/write status of the instruction. For each operand transfer, two or three bus operations are performed, one word transfer with I/O port address 00FA(H) and one or two bus operations with memory. Three bus operations are required for each word operand aligned on an odd byte address.

NOTE:
Odd-aligned numerics operands should be avoided when using an 80286 system running six or more memory-write wait states. The 80286 can generate an incorrect numerics address if all the following conditions are met:

— Two floating point (FP) instructions are fetched and in the 80286 queue.

— The first FP instruction is any floating point store except FSTSW AX.

— The second FP instruction accesses memory.

— The operand of the first instruction is aligned on an odd memory address.

— Six or more wait states are inserted during either of the last two memory write (odd aligned operands are transferred as two bytes) transfers of the first instruction.

The second FP operand's address will be incremented by one if these conditions are met. These conditions are most likely to occur in a multi-master system. For a hardware solution, contact your local Intel representative.

Commands to the numerics coprocessor should not be delayed by nine or more T-states. Excessive (nine or more) command-delays can cause the 80286 and 80287 to lose synchronization.

Interrupt Acknowledge Sequence

Figure 30 illustrates an interrupt acknowledge sequence performed by the 80286 in response to an INTR input. An interrupt acknowledge sequence consists of two INTA bus operations. The first allows a master 8259A Programmable Interrupt Controller (PIC) to determine which if any of its slaves should return the interrupt vector. An eight bit vector is read on D0–D7 of the 80286 during the second INTA bus operation to select an interrupt handler routine from the interrupt table.

The Master Cascade Enable (MCE) signal of the 82288 is used to enable the cascade address drivers, during INTA bus operations (See Figure 30), onto the local address bus for distribution to slave interrupt controllers via the system address bus. The 80286 emits the $\overline{LOCK}$ signal (active LOW) during T_S of the first INTA bus operation. A local bus "hold" request will not be honored until the end of the second INTA bus operation.

Three idle processor clocks are provided by the 80286 between INTA bus operations to allow for the minimum INTA to INTA time and CAS (cascade address) out delay of the 8259A. The second INTA bus operation must always have at least one extra T_C state added via logic controlling $\overline{READY}$. This is needed to meet the 8259A minimum INTA pulse width.

Local Bus Usage Priorities

The 80286 local bus is shared among several internal units and external HOLD requests. In case of simultaneous requests, their relative priorities are:

(Highest) Any transfers which assert $\overline{LOCK}$ either explicitly (via the LOCK instruction prefix) or implicitly (i.e. some segment descriptor accesses, interrupt acknowledge sequence, or an XCHG with memory).

The second of the two byte bus operations required for an odd aligned word operand.

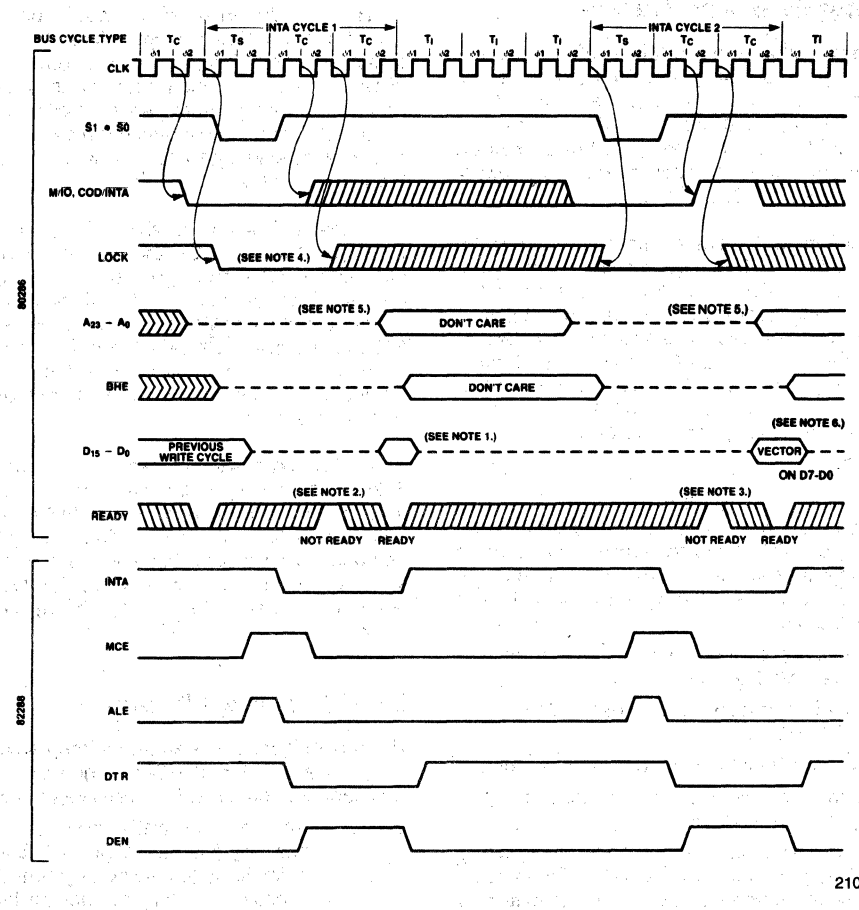
The second or third cycle of a processor extension data transfer.

Local bus request via HOLD input.

Processor extension data operand transfer via PEREQ input.

Data transfer performed by EU as part of an instruction.

(Lowest) An instruction prefetch request from BU. The EU will inhibit prefetching two processor clocks in advance of any data transfers to minimize waiting by EU for a prefetch to finish.



NOTES:
1. Data is ignored, upper data bus, D_8–D_{15}, should not change state during this time.
2. First INTA cycle should have at least one wait state inserted to meet 8259A minimum INTA pulse width.
3. Second INTA cycle should have at least one wait state inserted to meet 8259A minimum INTA pulse width.
4. $\overline{LOCK}$ is active for the first INTA cycle to prevent the 82289 from releasing the bus between INTA cycles in a multi-master system. $\overline{LOCK}$ is also active for the second INTA cycle.
5. A_{23}–A_0 exits 3-state OFF during $\phi 2$ of the second T_C in the INTA cycle.
6. Upper data bus should not change state during this time.

Figure 30. Interrupt Acknowledge Sequence

Halt or Shutdown Cycles

The 80286 externally indicates halt or shutdown conditions as a bus operation. These conditions occur due to a HLT instruction or multiple protection exceptions while attempting to execute one instruction. A halt or shutdown bus operation is signalled when $\overline{S1}$, $\overline{S0}$ and COD/$\overline{INTA}$ are LOW and M/$\overline{IO}$ is HIGH. A_1 HIGH indicates halt, and A_1 LOW indicates shutdown. The 82288 bus controller does not issue ALE, nor is $\overline{READY}$ required to terminate a halt or shutdown bus operation.

During halt or shutdown, the 80286 may service PEREQ or HOLD requests. A processor extension segment overrun exception during shutdown will inhibit further service of PEREQ. Either NMI or RESET will force the 80286 out of either halt or shutdown. An INTR, if interrupts are enabled, or a processor extension segment overrun exception will also force the 80286 out of halt.

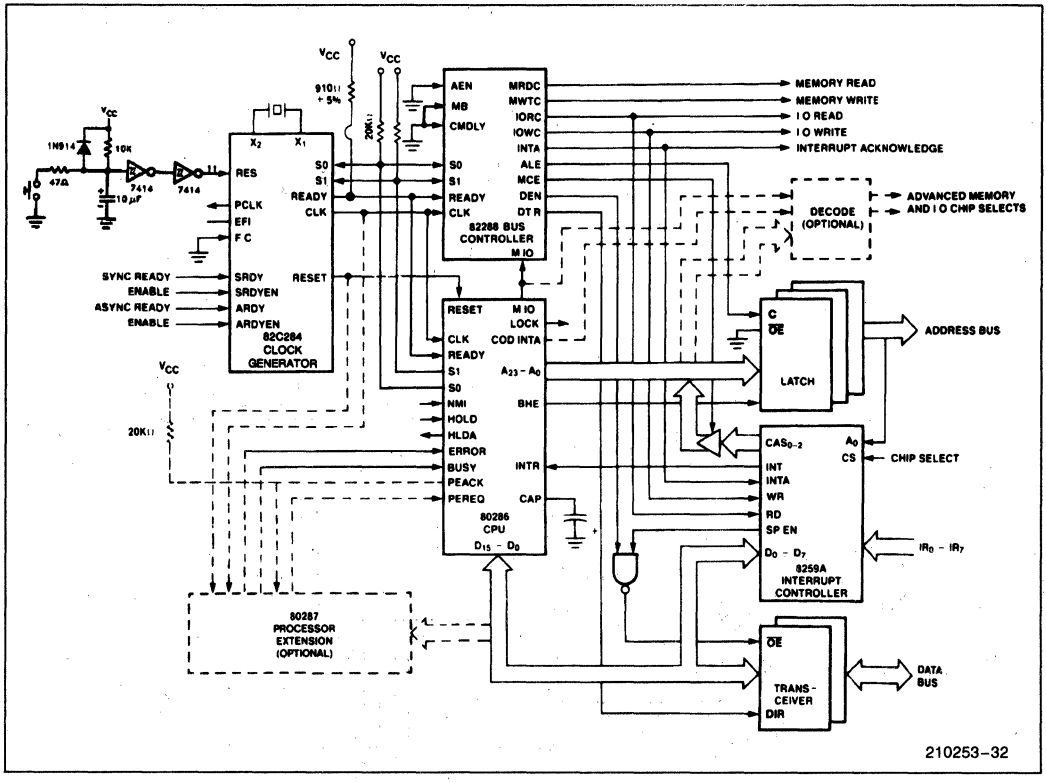


Figure 31. Basic 80286 System Configuration

SYSTEM CONFIGURATIONS

The versatile bus structure of the 80286 microsystem, with a full complement of support chips, allows flexible configuration of a wide range of systems. The basic configuration, shown in Figure 31, is similar to an 8086 maximum mode system. It includes the CPU plus an 8259A interrupt controller, 82C284 clock generator, and the 82288 Bus Controller.

As indicated by the dashed lines in Figure 31, the ability to add processor extensions is an integral feature of 80286 microsystems. The processor extension interface allows external hardware to perform special functions and transfer data concurrent with CPU execution of other instructions. Full system integrity is maintained because the 80286 supervises all data transfers and instruction execution for the processor extension.

The 80287 has all the instructions and data types of an 8087. The 80287 NPX can perform numeric calculations and data transfers concurrently with CPU program execution. Numerics code and data have the same integrity as all other information protected by the 80286 protection mechanism.

The 80286 can overlap chip select decoding and address propagation during the data transfer for the previous bus operation. This information is latched by ALE during the middle of a T_S cycle. The latched chip select and address information remains stable during the bus operation while the next cycle's address is being decoded and propagated into the system. Decode logic can be implemented with a high speed bipolar PROM.

The optional decode logic shown in Figure 31 takes advantage of the overlap between address and data of the 80286 bus cycle to generate advanced memory and IO-select signals. This minimizes system

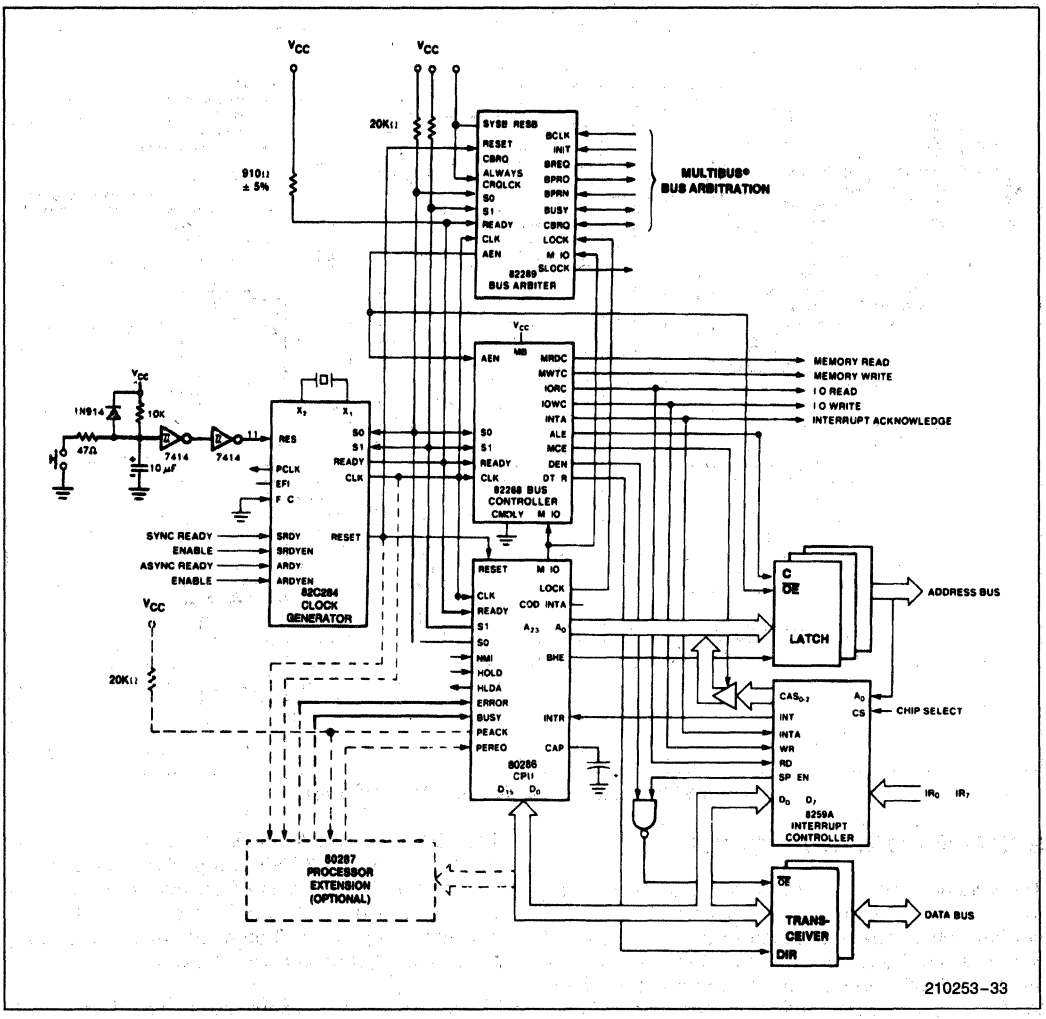


Figure 32. MULTIBUS® System Bus Interface

performance degradation caused by address propagation and decode delays. In addition to selecting memory and I/O, the advanced selects may be used with configurations supporting local and system buses to enable the appropriate bus interface for each bus cycle. The COD/$\overline{INTA}$ and M/$\overline{IO}$ signals are applied to the decode logic to distinguish between interrupt, I/O, code and data bus cycles.

By adding the 82289 bus arbiter chip, the 80286 provides a MULTIBUS system bus interface as shown in Figure 32. The ALE output of the 82288 for the MULTIBUS bus is connected to its CMDLY input to delay the start of commands one system CLK as required to meet MULTIBUS address and write data setup times. This arrangement will add at least one extra T_C state to each bus operation which uses the MULTIBUS.

A second 82288 bus controller and additional latches and transceivers could be added to the local bus of Figure 32. This configuration allows the 80286 to support an on-board bus for local memory and peripherals, and the MULTIBUS for system bus interfacing.

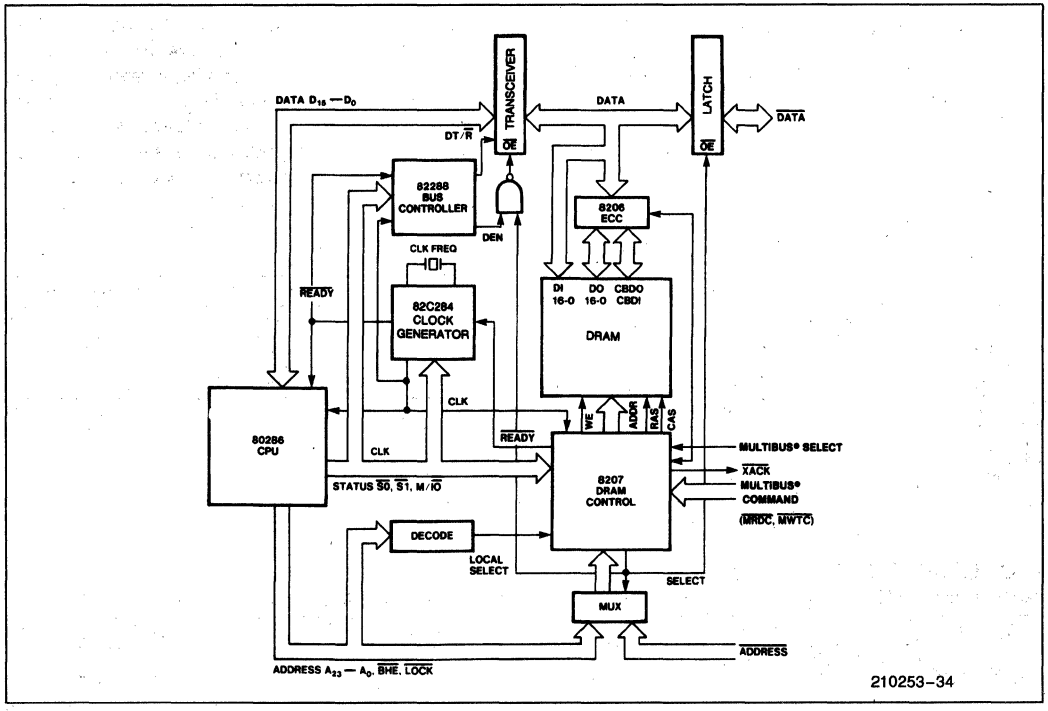


Figure 33. 80286 System Configuration with Dual-Ported Memory

Figure 33 shows the addition of dual ported dynamic memory between the MULTIBUS system bus and the 80286 local bus. The dual port interface is provided by the 8207 Dual Port DRAM Controller. The 8207 runs synchronously with the CPU to maximize throughput for local memory references. It also arbitrates between requests from the local and system buses and performs functions such as refresh, initialization of RAM, and read/modify/write cycles. The 8207 combined with the 8206 Error Checking and Correction memory controller provide for single bit error correction. The dual-ported memory can be combined with a standard MULTIBUS system bus interface to maximize performance and protection in multiprocessor system configurations.

Table 16. 80286 Systems Recommended Pull Up Resistor Values

80286 Pin and Name	Pullup Value	Purpose
4—$\overline{S1}$	20 KΩ ±10%	Pull $\overline{S0}$, $\overline{S1}$, and $\overline{PEACK}$ inactive during 80286 hold periods(1)
5—$\overline{S0}$		
6—$\overline{PEACK}$		
63—$\overline{READY}$	910Ω ±5%	Pull $\overline{READY}$ inactive within required minimum time (C_L = 150 pF, $I_R \leq 7$ mA)

NOTE:
1. Pull-up resistors are not required on $\overline{S0}$ and $\overline{S1}$ when the corresponding pins of the 82C284 are connected to $\overline{S0}$ and $\overline{S1}$.

I²ICE™-286 System Design Considerations

One of the advantages of using the 80286 is that full in-circuit emulation debugging support is provided through the I²ICE system 80286 probe. To utilize this powerful tool it is necessary that the system designer be aware of a few minor parametric and functional differences between the 80286 and I²ICE system 80286 probe. The I²ICE data sheet (I²ICE Integrated Instrumentation and In-Circuit Emulation System, order #210469) contains a detailed description of these design considerations. It is recommended that this document be reviewed by the 80286 system designer to determine whether or not these differences affect his design.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ....0°C to +70°C
Storage Temperature .....−65°C to +150°C
Voltage on Any Pin with
Respect to Ground.....−1.0V to +7V
Power Dissipation.....3.3W

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS (V_{CC} = 5V ±5%, T_{CASE} = 0°C to +85°C)*

Symbol	Parameter	Min	Max	Unit	Test Condition
I_{CC}	Supply Current (0°C Turn On)		600	mA	(Note 1)
C_{CLK}	CLK Input Capacitance		20	pF	(Note 2)
C_{IN}	Other Input Capacitance		10	pF	(Note 2)
C_O	Input/Output Capacitance		20	pF	(Note 2)

NOTES:
1. Tested at worst case load and maximum frequency.
2. These are not tested. They are guaranteed by design characterization.

D.C. CHARACTERISTICS

(V_{CC} = 5V ±5%, T_{CASE} = 0°C to +85°C)* Tested at the minimum operating frequency of the part.

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{IL}	Input LOW Voltage	−0.5	0.8	V	
V_{IH}	Input HIGH Voltage	2.0	V_{CC} +0.5	V	
V_{ILC}	CLK Input LOW Voltage	−0.5	0.6	V	
V_{IHC}	CLK Input HIGH Voltage	3.8	V_{CC} +0.5	V	
V_{OL}	Output LOW Voltge		0.45	V	I_{OL} = 2.0 mA
V_{OH}	Output HIGH Voltage	2.4		V	I_{OH} = −400.0 μA
I_{LI}	Input Leakage Current		±10	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{LCR}	Input CLK, RESET Leakage Current		±10	μA	0.45 ≤ V_{IN} ≤ V_{CC}
I_{LCR}	Input CLK, RESET Leakage Current		±1	mA	0 ≤ V_{IN} < 0.45
I_{IL}	Input Sustaining Current on $\overline{BUSY}$ and $\overline{ERROR}$ Pins	30	500	μA	V_{IN} = 0V
I_{LO}	Output Leakage Current		±10	μA	0.45 ≤ V_{OUT} ≤ V_{CC}
I_{LO}	Output Leakage Current		±1	mA	0 ≤ V_{OUT} < 0.45

*T_A is guaranteed from 0°C to +55°C as long as T_{CASE} is not exceeded.

A.C. CHARACTERISTICS (V_{CC} = 5V ±5%, T_{CASE} = 0°C to +85°C)*

AC timings are referenced to 0.8V and 2.0V points of signals as illustrated in datasheet waveforms, unless otherwise noted.

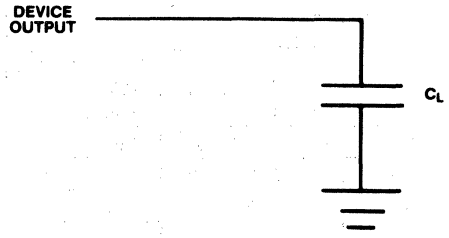
Symbol	Parameter	8 MHz		10 MHz		12.5 MHz (Preliminary)		Unit	Test Condition
		-8 Min	-8 Max	-10 Min	-10 Max	-12 Min	-12 Max		
1	System Clock (CLK) Period	62	250	50	250	40	250	ns	
2	System Clock (CLK) LOW Time	15	225	12	232	11	237	ns	at 1.0V
3	System Clock (CLK) HIGH Time	25	235	16	239	13	239	ns	at 3.6V
17	System Clock (CLK) Rise Time		10		8	—	8	ns	1.0V to 3.6V, (Note 7)
18	System Clock (CLK) Fall Time		10		8	—	8	ns	3.6V to 1.0V, (Note 7)
4	Asynch. Inputs Setup Time	20		20		15		ns	(Note 1)
5	Asynch. Inputs Hold Time	20		20		15		ns	(Note 1)
6	RESET Setup Time	28		23		18		ns	
7	RESET Hold Time	5		5		5		ns	
8	Read Data Setup Time	10		8		5		ns	
9	Read Data Hold Time	8		8		6		ns	
10	$\overline{\text{READY}}$ Setup Time	38		26		22		ns	
11	$\overline{\text{READY}}$ Hold Time	25		25		20		ns	
12	Status/$\overline{\text{PEACK}}$ Valid Delay	1	40	—	—	—	—	ns	(Notes 2, 3)
12a1	Status Active Delay	—	—	1	22	3	18	ns	(Notes 2, 3)
12a2	$\overline{\text{PEACK}}$ Active Delay	—	—	1	22	3	20	ns	(Notes 2, 3)
12b	Status/$\overline{\text{PEACK}}$ Inactive Delay	—	—	1	30	3	22	ns	(Notes 2, 3)
13	Address Valid Delay	1	60	1	35	1	32	ns	(Notes 2, 3)
14	Write Data Valid Delay	0	50	0	30	0	30	ns	(Notes 2, 3)
15	Address/Status/Data Float Delay	0	50	0	47	0	32	ns	(Notes 2, 4, 7)
16	HLDA Valid Delay	0	50	0	47	0	27	ns	(Notes 2, 3)
19	Address Valid To Status Valid Setup Time	38		27		22		ns	(Notes 3, 5, 6)

*T_A is guaranteed from 0°C to +55°C as long as T_{CASE} is not exceeded.

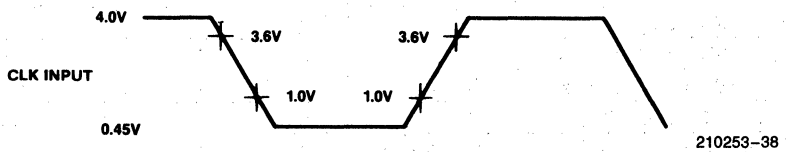
NOTES:

1. Asynchronous inputs are INTR, NMI, HOLD, PEREQ, $\overline{\text{ERROR}}$, and $\overline{\text{BUSY}}$. This specification is given only for testing purposes, to assure recognition at a specific CLK edge.
2. Delay from 1.0V on the CLK, to 0.8V or 2.0V or float on the output as appropriate for valid or floating condition.
3. Output load: C_L = 100 pF.
4. Float condition occurs when output current is less than I_{LO} in magnitude.
5. Delay measured from address either reaching 0.8V or 2.0V (valid) to status going active reaching 2.0V or status going inactive reaching 0.8V.
6. For load capacitance of 10 pF or more on STATUS/$\overline{\text{PEACK}}$ lines, subtract typically 7 ns for 8 MHz, 10 MHz and 12.5 MHz spec.
7. These are not tested. They are guaranteed by design characterization.

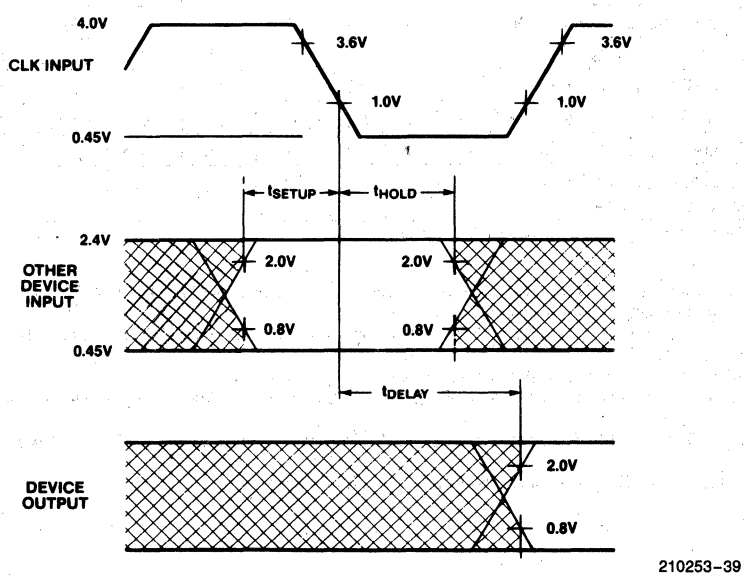
A.C. CHARACTERISTICS (Continued)



NOTE 8:
AC Test Loading on Outputs



NOTE 9:
AC Drive and Measurement Points—CLK Input



NOTE 10:
AC Setup, Hold and Delay Time Measurement—General

A.C. CHARACTERISTICS (Continued)

82C284 Timing Requirements

Symbol	Parameter	82C284-8		82C284-10		82C284-12		Units	Test Conditions
		Min	Max	Min	Max	Min	Max		
11	$\overline{SRDY}/\overline{SRDYEN}$ Setup Time	17		15		15		ns	
12	$\overline{SRDY}/\overline{SRDYEN}$ Hold Time	0		2		2		ns	
13	$\overline{ARDY}/\overline{ARDYEN}$ Setup Time	0		0		0		ns	(Note 1)
14	$\overline{ARDY}/\overline{ARDYEN}$ Hold Time	30		30		25		ns	(Note 1)
19	PCLK Delay	0	45	0	35	0	23	ns	C_L = 75 pF I_{OL} = 5 mA I_{OH} = −1 mA

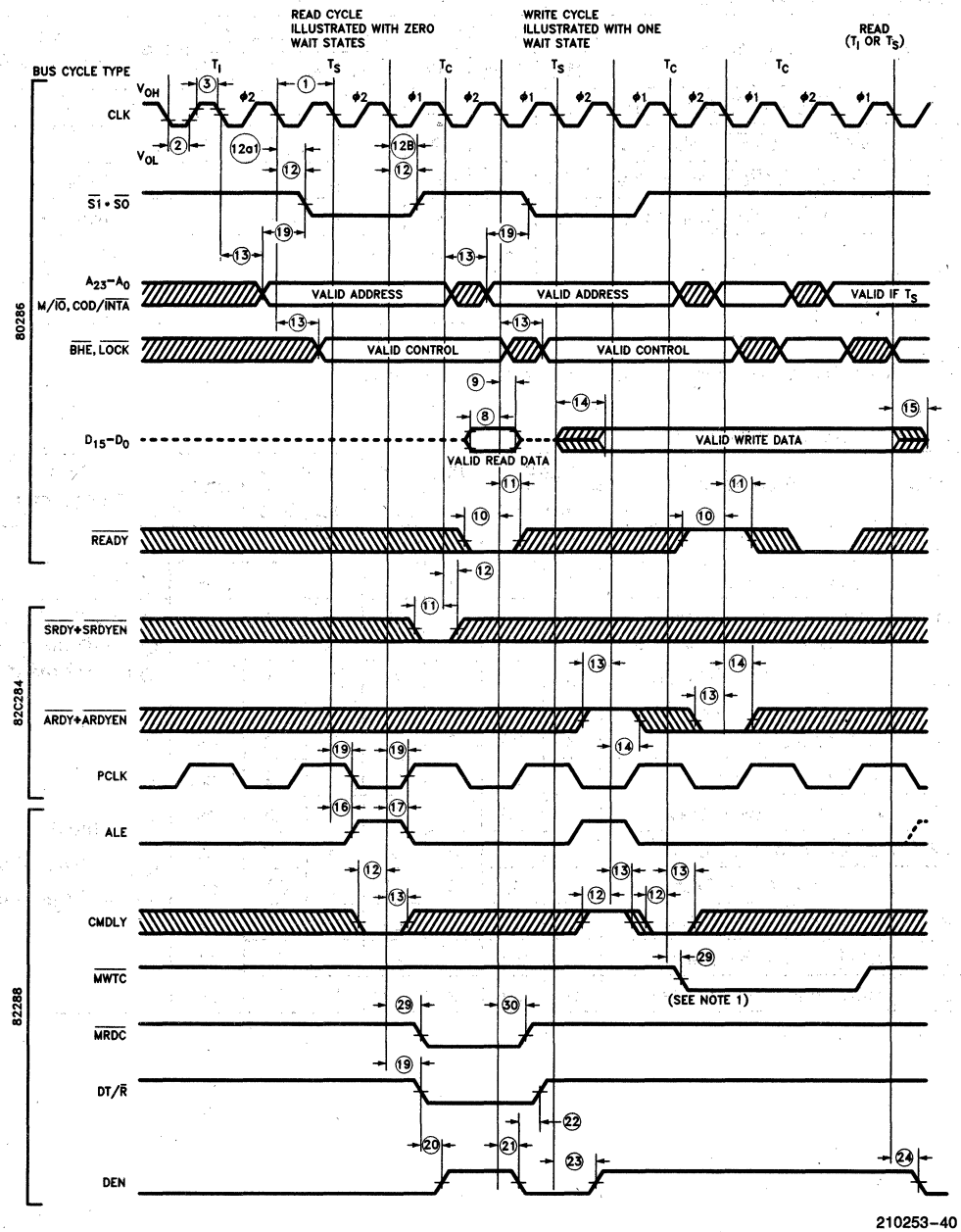
NOTE 1:
These times are given for testing purposes to assure a predetermined action.

82288 Timing Requirements

Symbol	Parameter		82288-8		82288-10		82288-12		Units	Test Conditions
			Min	Max	Min	Max	Min	Max		
12	CMDLY Setup Time		20		15		15		ns	
13	CMDLY Hold Time		1		1		1		ns	
30	Command Delay from CLK	Command Inactive	5	20	5	20	5	20	ns	C_L = 300 pF max I_{OL} = 32 mA max I_{OH} = −5 mA max
29		Command Active	3	25	3	21	3	21		
16	ALE Active Delay		3	20	3	16	3	16	ns	C_L = 150 pF I_{OL} = 16 mA max I_{OH} = −1 mA max
17	ALE Inactive Delay			25		19		19	ns	
19	DT/$\overline{R}$ Read Active Delay			25		23		23	ns	
22	DT/$\overline{R}$ Read Inactive Delay		5	35	5	20	5	18	ns	
20	DEN Read Active Delay		5	35	5	21	5	21	ns	
21	DEN Read Inactive Delay		3	35	3	21	3	19	ns	
23	DEN Write Active Delay			30		23		23	ns	
24	DEN Write Inactive Delay		3	30	3	19	3	19	ns	

WAVEFORMS

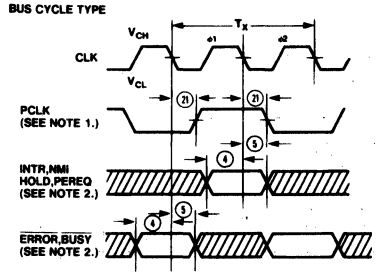
MAJOR CYCLE TIMING



NOTE:
1. The modified timing is due to the $\overline{CMDLY}$ signal being active.

WAVEFORMS (Continued)

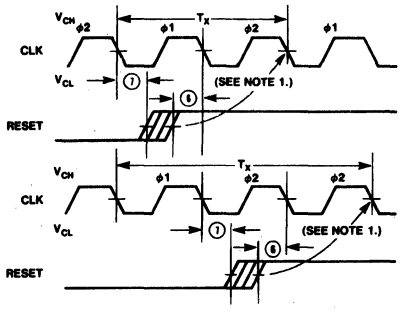
80286 ASYNCHRONOUS INPUT SIGNAL TIMING



210253-41

NOTES:
1. PCLK indicates which processor cycle phase will occur on the next CLK. PCLK may not indicate the correct phase until the first bus cycle is performed.
2. These inputs are asynchronous. The setup and hold times shown assure recognition for testing purposes.

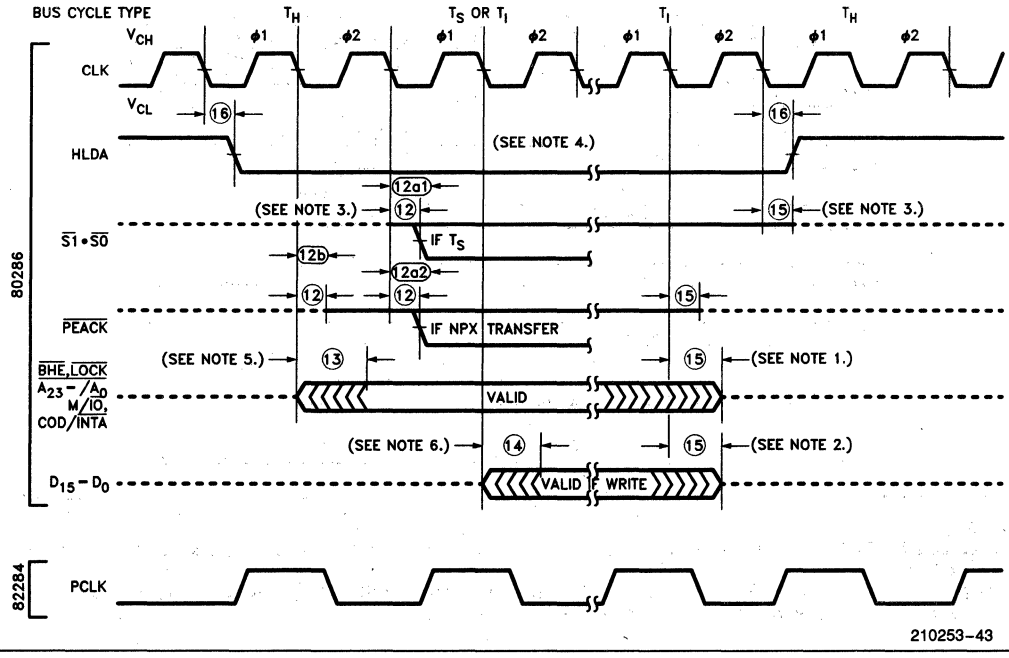
80286 RESET INPUT TIMING AND SUBSEQUENT PROCESSOR CYCLE PHASE



210253-42

NOTE:
When RESET meets the setup time shown, the next CLK will start or repeat $\phi2$ of a processor cycle.

EXITING AND ENTERING HOLD

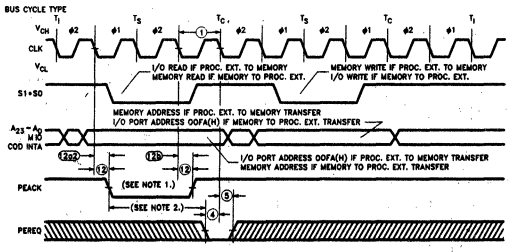


210253-43

NOTES:
1. These signals may not be driven by the 80286 during the time shown. The worst case in terms of latest float time is shown.
2. The data bus will be driven as shown if the last cycle before T_I in the diagram was a write T_C.
3. The 80286 floats its status pins during T_H. External 20 KΩ resistors keep these signals high (see Table 16).
4. For HOLD request set up to HLDA, refer to Figure 29.
5. $\overline{BHE}$ and $\overline{LOCK}$ are driven at this time but will not become valid until T_S.
6. The data bus will remain in 3-state OFF if a read cycle is performed.

WAVEFORMS (Continued)

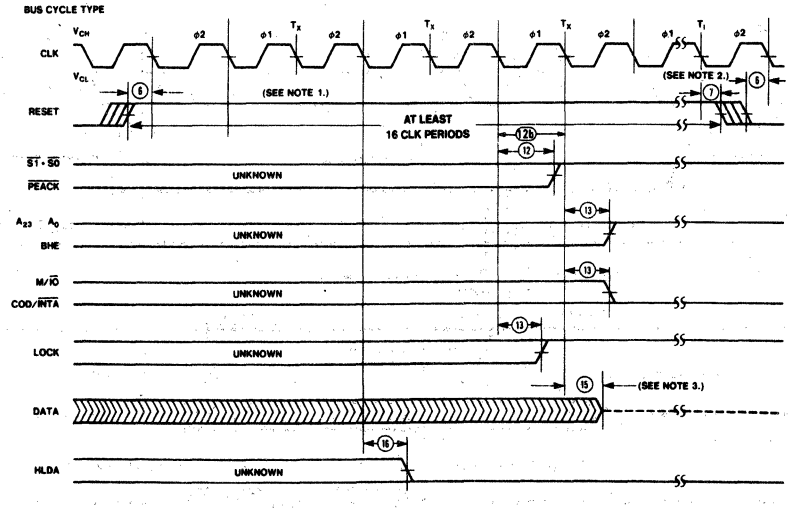
80286 PEREQ/$\overline{\text{PEACK}}$ TIMING FOR ONE TRANSFER ONLY



210253-44

NOTES:
1. $\overline{\text{PEACK}}$ always goes active during the first bus operation of a processor extension data operand transfer sequence. The first bus operation will be either a memory read at operand address or I/O read at port address OOFA(H).
2. To prevent a second processor extension data operand transfer, the worst case maximum time (Shown above) is: $3\times$ ① $-12a_{2max.}-$ ④$_{min.}$. The actual, configuration dependent, maximum time is: $3\times$ ① $-12a_{2max.}-$ ④$_{min.}$ $+ A\times 2\times$ ①.
A is the number of extra T_C states added to either the first or second bus operation of the processor extension data operand transfer sequence.

INITIAL 80286 PIN STATE DURING RESET



210253-45

NOTES:
1. Setup time for RESET ↑ may be violated with the consideration that $\phi 1$ of the processor clock may begin one system CLK period later.
2. Setup and hold times for RESET ↓ must be met for proper operation, but RESET ↓ may occur during $\phi 1$ or $\phi 2$.
3. The data bus is only guaranteed to be in 3-state OFF at the time shown.

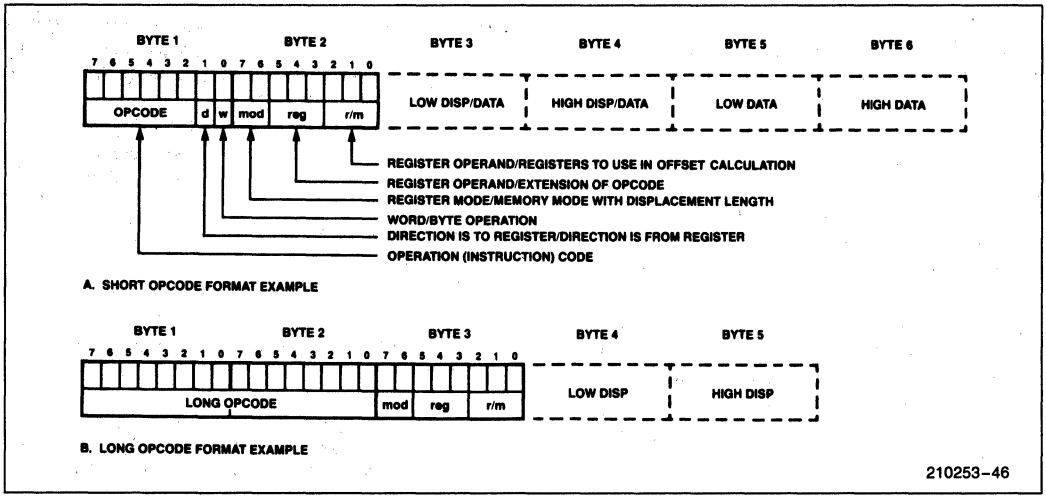


Figure 35. 80286 Instruction Format Examples

80286 INSTRUCTION SET SUMMARY

Instruction Timing Notes

The instruction clock counts listed below establish the maximum execution rate of the 80286. With no delays in bus cycles, the actual clock count of an 80286 program will average 5% more than the calculated clock count, due to instruction sequences which execute faster than they can be fetched from memory.

To calculate elapsed times for instruction sequences, multiply the sum of all instruction clock counts, as listed in the table below, by the processor clock period. An 8 MHz processor clock has a clock period of 125 nanoseconds and requires an 80286 system clock (CLK input) of 16 MHz.

Instruction Clock Count Assumptions

1. The instruction has been prefetched, decoded, and is ready for execution. Control transfer instruction clock counts include all time required to fetch, decode, and prepare the next instruction for execution.
2. Bus cycles do not require wait states.
3. There are no processor extension data transfer or local bus HOLD requests.
4. No exceptions occur during instruction execution.

Instruction Set Summary Notes

Addressing displacements selected by the MOD field are not shown. If necessary they appear after the instruction fields shown.

Above/below refers to unsigned value

Greater refers to positive signed value

Less refers to less positive (more negative) signed values

if d = 1 then to register; if d = 0 then from register

if w = 1 then word instruction; if w = 0 then byte instruction

if s = 0 then 16-bit immediate data form the operand

if s = 1 then an immediate data byte is sign-extended to form the 16-bit operand

x don't care

z used for string primitives for comparison with ZF FLAG

If two clock counts are given, the smaller refers to a register operand and the larger refers to a memory operand

* = add one clock if offset calculation requires summing 3 elements

n = number of times repeated

m = number of bytes of code in next instruction

Level (L)—Lexical nesting level of the procedure

The following comments describe possible exceptions, side effects, and allowed usage for instructions in both operating modes of the 80286.

REAL ADDRESS MODE ONLY

1. This is a protected mode instruction. Attempted execution in real address mode will result in an undefined opcode exception (6).
2. A segment overrun exception (13) will occur if a word operand reference at offset FFFF(H) is attempted.
3. This instruction may be executed in real address mode to initialize the CPU for protected mode.
4. The IOPL and NT fields will remain 0.
5. Processor extension segment overrun interrupt (9) will occur if the operand exceeds the segment limit.

EITHER MODE

6. An exception may occur, depending on the value of the operand.
7. $\overline{\text{LOCK}}$ is automatically asserted regardless of the presence or absence of the LOCK instruction prefix.
8. $\overline{\text{LOCK}}$ does not remain active between all operand transfers.

PROTECTED VIRTUAL ADDRESS MODE ONLY

9. A general protection exception (13) will occur if the memory operand cannot be used due to either a segment limit or access rights violation. If a stack segment limit is violated, a stack segment overrun exception (12) occurs.
10. For segment load operations, the CPL, RPL, and DPL must agree with privilege rules to avoid an exception. The segment must be present to avoid a not-present exception (11). If the SS register is the destination, and a segment not-present violation occurs, a stack exception (12) occurs.
11. All segment descriptor accesses in the GDT or LDT made by this instruction will automatically assert $\overline{\text{LOCK}}$ to maintain descriptor integrity in multiprocessor systems.
12. JMP, CALL, INT, RET, IRET instructions referring to another code segment will cause a general protection exception (13) if any privilege rule is violated.
13. A general protection exception (13) occurs if CPL $\neq$ 0.
14. A general protection exception (13) occurs if CPL > IOPL.
15. The IF field of the flag word is not updated if CPL > IOPL. The IOPL field is updated only if CPL = 0.
16. Any violation of privilege rules as applied to the selector operand do not cause a protection exception; rather, the instruction does not return a result and the zero flag is cleared.
17. If the starting address of the memory operand violates a segment limit, or an invalid access is attempted, a general protection exception (13) will occur before the ESC instruction is executed. A stack segment overrun exception (12) will occur if the stack limit is violated by the operand's starting address. If a segment limit is violated during an attempted data transfer then a processor extension segment overrun exception (9) occurs.
18. The destination of an INT, JMP, CALL, RET or IRET instruction must be in the defined limit of a code segment or a general protection exception (13) will occur.

80286 INSTRUCTION SET SUMMARY

FUNCTION	FORMAT	CLOCK COUNT: Real Address Mode	CLOCK COUNT: Protected Virtual Address Mode	COMMENTS: Real Address Mode	COMMENTS: Protected Virtual Address Mode
DATA TRANSFER					
MOV = Move:					
Register to Register/Memory	1000100w | mod reg r/m	2,3*	2,3*	2	9
Register/memory to register	1000101w | mod reg r/m	2,5*	2,5*	2	9
Immediate to register/memory	1100011w | mod 000 r/m | data | data if w = 1	2,3*	2,3*	2	9
Immediate to register	1011w reg | data | data if w=1	2	2		
Memory to accumulator	1010000w | addr-low | addr-high	5	5	2	9
Accumulator to memory	1010001w | addr-low | addr-high	3	3	2	9
Register/memory to segment register	10001110 | mod 0 reg r/m	2,5*	17,19*	2	9,10,11
Segment register to register/memory	10001100 | mod 0 reg r/m	2,3*	2,3*	2	9
PUSH = Push:					
Memory	11111111 | mod 1 1 0 r/m	5*	5*	2	9
Register	01010 reg	3	3	2	9
Segment register	000reg 1 1 0	3	3	2	9
Immediate	011010s0 | data | data if s=0	3	3	2	9
PUSHA = Push All	01100000	17	17	2	9
POP = Pop:					
Memory	10001111 | mod 000 r/m	5*	5*	2	9
Register	01011 reg	5	5	2	9
Segment register	000reg 1 1 1 (reg≠01)	5	20	2	9,10,11
POPA = Pop All	01100001	19	19	2	9
XCHG = Exhcange:					
Register/memory with register	1000011w | mod reg r/m	3,5*	3,5*	2,7	7,9
Register with accumulator	10010 reg	3	3		
IN = Input from:					
Fixed port	1110010w | port	5	5		14
Variable port	1110110w	5	5		14
OUT = Output to:					
Fixed port	1110011w | port	3	3		14
Variable port	1110111w	3	3		14
XLAT = Translate byte to AL	11010111	5	5		9
LEA = Load EA to register	10001101 | mod reg r/m	3*	3*		
LDS = Load pointer to DS	11000101 | mod reg r/m (mod≠11)	7*	21*	2	9,10,11
LES = Load pointer to ES	11000100 | mod reg r/m (mod≠1)	7*	21*	2	9,10,11

Shaded areas indicate instructions not available in 8086, 88 microsystems.

80286 INSTRUCTION SET SUMMARY (Continued)

FUNCTION	FORMAT	CLOCK COUNT: Real Address Mode	CLOCK COUNT: Protected Virtual Address Mode	COMMENTS: Real Address Mode	COMMENTS: Protected Virtual Address Mode
DATA TRANSFER (Continued)					
LAHF Load AH with flags	10011111	2	2		
SAHF = Store AH into flags	10011110	2	2		
PUSHF = Push flags	10011100	3	3	2	9
POPF = Pop flags	10011101	5	5	2,4	9,15
ARITHMETIC					
ADD = **Add:**					
Reg/memory with register to either	000000dw | mod reg r/m	2,7*	2,7*	2	9
Immediate to register/memory	100000sw | mod 000 r/m | data | data if s w = 01	3,7*	3,7*	2	9
Immediate to accumulator	0000010w | data | data if w=1	3	3		
ADC = **Add with carry:**					
Reg/memory with register to either	000100dw | mod reg r/m	2,7*	2,7*	2	9
Immediate to register/memory	100000sw | mod 010 r/m | data | data if s w = 01	3,7*	3,7*	2	9
Immediate to accumulator	0001010w | data | data if w=1	3	3		
INC = **Increment:**					
Register/memory	1111111w | mod 000 r/m	2,7*	2,7*	2	9
Register	01000reg	2	2		
SUB = **Subtract:**					
Reg/memory and register to either	001010dw | mod reg r/m	2,7*	2,7*	2	9
Immediate from register/memory	100000sw | mod 101 r/m | data | data if s w = 01	3,7*	3,7*	2	9
Immediate from accumulator	0010110w | data | data if w=1	3	3		
SBB = **Subtract with borrow:**					
Reg/memory and register to either	000110dw | mod reg r/m	2,7*	2,7*	2	9
Immediate from register/memory	100000sw | mod 011 r/m | data | data if s w=01	3,7*	3,7*	2	9
Immediate from accumulator	0001110w | data | data if w=1	3	3		
DEC = **Decrement**					
Register/memory	1111111w | mod 001 r/m	2,7*	2,7*	2	9
Register	01001 reg	2	2		
CMP = **Compare**					
Register/memory with register	0011101w | mod reg r/m	2,6*	2,6*	2	9
Register with register/memory	0011100w | mod reg r/m	2,7*	2,7*	2	9
Immediate with register/memory	100000sw | mod 111 r/m | data | data if s w=01	3,6*	3,6*	2	9
Immediate with accumulator	0011110w | data | data if w=1	3	3		
NEG = Change sign	1111011w | mod 011 r/m	2	7*	2	9
AAA = ASCII adjust for add	00110111	3	3		
DAA = Decimal adjust for add	00100111	3	3		

80286 INSTRUCTION SET SUMMARY (Continued)

FUNCTION	FORMAT	CLOCK COUNT: Real Address Mode	CLOCK COUNT: Protected Virtual Address Mode	COMMENTS: Real Address Mode	COMMENTS: Protected Virtual Address Mode
ARITHMETIC (Continued)					
AAS = ASCII adjust for subtract	00111111	3	3		
DAS = Decimal adjust for subtract	00101111	3	3		
MUL = Multiply (unsigned):	1111011w mod 100 r/m				
Register-Byte		13	13		
Register-Word		21	21		
Memory-Byte		16*	16*	2	9
Memory-Word		24*	24*	2	9
IMUL = Integer multiply (signed):	1111011w mod 101 r/m				
Register-Byte		13	13		
Register-Word		21	21		
Memory-Byte		16*	16*	2	9
Memory-Word		24*	24*	2	9
IMUL = Integer immediate multiply (signed)	011010s1 mod reg r/m data data if s = 0	21,24*	21,24*	2	9
DIV = Divide (unsigned)	1111011w mod 110 r/m				
Register-Byte		14	14	6	6
Register-Word		22	22	6	6
Memory-Byte		17*	17*	2,6	6,9
Memory-Word		25*	25*	2,6	6,9
IDIV = Integer divide (signed)	1111011w mod 111 r/m				
Register-Byte		17	17	6	6
Register-Word		25	25	6	6
Memory-Byte		20*	20*	2,6	6,9
Memory-Word		28*	28*	2,6	6,9
AAM = ASCII adjust for multiply	11010100 00001010	16	16		
AAD = ASCII adjust for divide	11010101 00001010	14	14		
CBW = Convert byte to word	10011000	2	2		
CWD = Convert word to double word	10011001	2	2		
LOGIC					
Shift/Rotate Instructions:					
Register/Memory by 1	1101000w mod TTT r/m	2,7*	2,7*	2	9
Register/Memory by CL	1101001w mod TTT r/m	5+n,8+n*	5+n,8+n*	2	9
Register/Memory by Count	1100000w mod TTT r/m count	5+n,8+n*	5+n,8+n*	2	9

TTT	Instruction
000	ROL
001	ROR
010	RCL
011	RCR
100	SHL/SAL
101	SHR
111	SAR

Shaded areas indicate instructions not available in 8086, 88 microsystems.

80286 INSTRUCTION SET SUMMARY (Continued)

FUNCTION	FORMAT	CLOCK COUNT		COMMENTS	
		Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
ARITHMETIC (Continued)					
AND = And:					
Reg/memory and register to either	001000dw | mod reg r/m	2,7*	2,7*	2	9
Immediate to register/memory	1000000w | mod 100 r/m | data | data if w=1	3,7*	3,7*	2	9
Immediate to accumulator	0010010w | data | data if w=1	3	3		
TEST = And function to flags, no result:					
Register/memory and register	1000010w | mod reg r/m	2,6*	2,6*	2	9
Immediate data and register/memory	1111011w | mod 000 r/m | data | data if w=1	3,6*	3,6*	2	9
Immediate data and accumulator	1010100w | data | data if w=1	3	3		
OR = Or:					
Reg/memory and register to either	000010dw | mod reg r/m	2,7*	2,7*	2	9
Immediate to register/memory	1000000w | mod 001 r/m | data | data if w=1	3,7*	3,7*	2	9
Immediate to accumulator	0000110w | data | data if w=1	3	3		
XOR = Exclusive or:					
Reg/memory and register to either	001100dw | mod reg r/m	2,7*	2,7*	2	9
Immediate to register/memory	1000000w | mod 110 r/m | data | data if w = 1	3,7*	3,7*	2	9
Immediate to accumulator	0011010w | data | data if w = 1	3	3		
NOT = Invert register/memory	1111011w | mod 010 r/m	2,7*	2,7*	2	9
STRING MANIPULATION:					
MOVS = Move byte/word	1010010w	5	5	2	9
CMPS = Compare byte/word	1010011w	8	8	2	9
SCAS = Scan byte/word	1010111w	7	7	2	9
LODS = Load byte/wd to AL/AX	1010110w	5	5	2	9
STOS = Stor byte/wd from AL/A	1010101w	3	3	2	9
INS = Input byte/wd from DX port	0110110w	5	5	2	9,14
OUTS = Output byte/wd to DX port	0110111w	5	5	2	9,14
Repeated by count in CX					
MOVS = Move string	11110011 | 1010010w	5+4n	5+4n	2	9
CMPS = Compare string	1111001z | 1010011w	5+9n	5+9n	2,8	8,9
SCAS = Scan string	1111001z | 1010111w	5+8n	5+8n	2,8	8,9
LODS = Load string	11110011 | 1010110w	5+4n	5+4n	2,8	8,9
STOS = Store string	11110011 | 1010101w	4+3n	4+3n	2,8	8,9
INS = Input string	11110011 | 0110110w	5+4n	5+4n	2	9,14
OUTS = Output string	11110011 | 0110111w	5+4n	5+4n	2	9,14

Shaded areas indicate instructions not available in 8086, 88 microsystems.

80286 INSTRUCTION SET SUMMARY (Continued)

FUNCTION	FORMAT	CLOCK COUNT: Real Address Mode	CLOCK COUNT: Protected Virtual Address Mode	COMMENTS: Real Address Mode	COMMENTS: Protected Virtual Address Mode
CONTROL TRANSFER					
CALL = Call:					
Direct within segment	11101000 | disp-low | disp-high	7+m	7+m	2	18
Register/memory indirect within segment	11111111 | mod 010 r/m	7+m, 11+m*	7+m, 11+m*	2,8	8,9,18
Direct intersegment	10011010 | segment offset	13+m	26+m	2	11,12,18
Protected Mode Only (Direct Intersegment):	| segment selector				
Via call gate to same privilege level			41+m		8,11,12,18
Via call gate to different privilege level, no parameters			82+m		8,11,12,18
Via call gate to different privilege level, x parameters			86+4x+m		8,11,12,18
Via TSS			177+m		8,11,12,18
Via task gate			182+m		8,11,12,18
Indirect intersegment	11111111 | mod 011 r/m (mod≠11)	16+m	29+m*	2	8,9,11,12,18
Protected Mode Only (Indirect Intersegment):					
Via call gate to same privilege level			44+m*		8,9,11,12,18
Via call gate to different privilege level, no parameters			83+m*		8,9,11,12,18
Via call gate to different privilege level, x parameters			90+4x+m*		8,9,11,12,18
Via TSS			180+m*		8,9,11,12,18
Via task gate			185+m*		8,9,11,12,18
JMP = Unconditional jump:					
Short/long	11101011 | disp-low	7+m	7+m		18
Direct within segment	11101001 | disp-low | disp-high	7+m	7+m		18
Register/memory indirect within segment	11111111 | mod 100 r/m	7+m, 11+m*	7+m, 11+m*	2	9,18
Direct intersegment	11101010 | segment offset	11+m	23+m		11,12,18
Protected Mode Only (Direct Intersegment):	| segment selector				
Via call gate to same privilege level			38+m		8,11,12,18
Via TSS			175+m		8,11,12,18
Via task gate			180+m		8,11,12,18
Indirect intersegment	11111111 | mod 101 r/m (mod≠11)	15+m*	26+m*	2	8,9,11,12,18
Protected Mode Only (Indirect Intersegment):					
Via call gate to same privilege level			41+m*		8,9,11,12,18
Via TSS			178+m*		8,9,11,12,18
Via task gate			183+m*		8,9,11,12,18
RET = Return from CALL:					
Within segment	11000011	11+m	11+m	2	8,9,18
Within seg adding immed to SP	11000010 | data-low | data-high	11+m	11+m	2	8,9,18
Intersegment	11001011	15+m	25+m	2	8,9,11,12,18
Intersegment adding immediate to SP	11001010 | data-low | data-high	15+m		2	8,9,11,12,18
Protected Mode Only (RET):					
To different privilege level			55+m		9,11,12,18

80286 INSTRUCTION SET SUMMARY (Continued)

FUNCTION	FORMAT	CLOCK COUNT: Real Address Mode	CLOCK COUNT: Protected Virtual Address Mode	COMMENTS: Real Address Mode	COMMENTS: Protected Virtual Address Mode
CONTROL TRANSFER (Continued)					
JE/JZ = Jump on equal zero	01110100 disp	7+m or 3	7+m or 3		18
JL/JNGE = Jump on less/not greater or equal	01111100 disp	7+m or 3	7+m or 3		18
JLE/JNG = Jump on less or equal/not greater	01111110 disp	7+m or 3	7+m or 3		18
JB/JNAE = Jump on below/not above or equal	01110010 disp	7+m or 3	7+m or 3		18
JBE/JNA = Jump on below or equal/not above	01110110 disp	7+m or 3	7+m or 3		18
JP/JPE = Jump on parity/parity even	01111010 disp	7+m or 3	7+m or 3		18
JO = Jump on overflow	01110000 disp	7 +m or 3	7+m or 3		18
JS = Jump on sign	01111000 disp	7+ m or 3	7+m or 3		18
JNE/JNZ = Jump on not equal/not zero	01110101 disp	7+m or 3	7+m or 3		18
JNL/JGE = Jump on not less/greater or equal	01111101 disp	7+m or 3	7+m or 3		18
JNLE/JG = Jump on not less or equal/greater	01111111 disp	7+m or 3	7+m or 3		18
JNB/JAE = Jump on not below/above or equal	01110011 disp	7+m or 3	7+m or 3		18
JNBE/JA = Jump on not below or equal/above	01110111 disp	7+m or 3	7+m or 3		18
JNP/JPO = Jump on not par/par odd	01111011 disp	7+m or 3	7+m or 3		18
JNO = Jump on not overflow	01110001 disp	7+m or 3	7+m or 3		18
JNS = Jump on not sign	01111001 disp	7 +m or 3	7+m or 3		18
LOOP = Loop CX times	11100010 disp	8 +m or 4	8+m or 4		18
LOOPZ/LOOPE = Loop while zero/equal	11100001 disp	8+m or 4	8+m or 4		18
LOOPNZ/LOOPNE = Loop while not zero/equal	11100000 disp	8+m or 4	8+m or 4		18
JCXZ = Jump on CX zero	11100011 disp	8 +m or 4	8+m or 4		18
ENTER = Enter Procedure	11001000 data-low data-high L			2,8	8,9
L=0		11	11	2,8	8,9
L=1		15	15	2,8	8,9
L > 1		16+4(L − 1)	16+4(L − 1)	2,8	8,9
LEAVE = Leave Procedure	11001001	5	5		
INT = Interrupt:					
Type specified	11001101 type	23+m		2,7,8	
Type 3	11001100	23+m		2,7,8	
INTO = Interrupt on overflow	11001110	24 +m or 3 (3 if no interrupt)	(3 if no interrupt)	2,6,8	

Shaded areas indicate instructions not available in 8086, 88 microsystems.

80286 INSTRUCTION SET SUMMARY (Continued)

FUNCTION	FORMAT	CLOCK COUNT		COMMENTS	
		Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
CONTROL TRANSFER (Continued)					
Protected Mode Only:					
Via interrupt or trap gate to same privilege level			40+ m		7,8,11,12,18
Via interrupt or trap gate to fit different privilege level			78+ m		7,8,11,12,18
Via Task Gate			167+m		7,8,11,12,18
IRET = Interrupt return	11001111	17+m	31+ m	2,4	8,9,11,12,15,18
Protected Mode Only:					
To different privilege level			55+m		8,9,11,12,15,18
To different task (NT = 1)			169+m		8,9,11,12,18
BOUND = Detect value out of range	01100010 mod reg r/m	13*	13* (Use INT clock count if exception 5)	2,6	6,8,9,11,12,18
PROCESSOR CONTROL					
CLC = Clear carry	11111000	2	2		
CMC = Complement carry	11110101	2	2		
STC = Set carry	11111001	2	2		
CLD = Clear direction	11111100	2	2		
STD = Set direction	11111101	2	2		
CLI = Clear interrupt	11111010	3	3		14
STI = Set interrupt	11111011	2	2		14
HLT = Halt	11110100	2	2		13
WAIT = Wait	10011011	3	3		
LOCK = Bus lock prefix	11110000	0	0		14
CTS = Clear task switched flag	00001111 00000110	2	2	3	13
ESC = Processor Extension Escape	11011TTT mod LLL r/m (TTT LLL are opcode to processor extension)	9–20*	9–20*	5,8	8,17
SEG = Segment Override Prefix	001 reg 110	0	0		
PROTECTION CONTROL					
LGDT = Load global descriptor table register	00001111 00000001 mod 010 r/m	11*	11*	2,3	9,13
SGDT = Store global descriptor table register	00001111 00000001 mod 000 r/m	11*	11*	2,3	9
LIDT = Load interrupt descriptor table register	00001111 00000001 mod 011 r/m	12*	12*	2,3	9,13
SIDT = Store interrupt descriptor table register	00001111 00000001 mod 001 r/m	12*	12*	2,3	9
LLDT = Load local descriptor table register from register memory	00001111 00000000 mod 010 r/m		17,19*	1	9,11,13
SLDT = Store local descriptor table register to register/memory	00001111 00000000 mod 000 r/m		2,3*	1	9

Shaded areas indicate instructions not available in 8086, 88 microsystems.

80286 INSTRUCTION SET SUMMARY (Continued)

FUNCTION	FORMAT	CLOCK COUNT: Real Address Mode	CLOCK COUNT: Protected Virtual Address Mode	COMMENTS: Real Address Mode	COMMENTS: Protected Virtual Address Mode
PROTECTION CONTROL (Continued)					
LTR = Local task register from register/memory	00001111 | 00000000 | mod011 r/m		17,19*	1	9,11,13
STR = Store task register to register memory	00001111 | 00000000 | mod001 r/m		2,3*	1	9
LMSW = Load machine status word from register/memory	00001111 | 00000001 | mod110 r/m	3,6*	3,6*	2,3	9,13
SMSW = Store machine status word	00001111 | 00000001 | mod100 r/m	2,3*	2,3*	2,3	9
LAR = Load access rights from register/memory	00001111 | 00000010 | mod reg r/m		14,16*	1	9,11,16
LSL = Load segment limit from register/memory	00001111 | 00000011 | mod reg r/m		14,16*	1	9,11,16
ARPL = Adjust requested privilege level: from register/memory	01100011 | mod reg r/m		10*,11*	2	8,9
VERR = Verify read access: register/memory	00001111 | 00000000 | mod100r/m		14,16*	1	9,11,16
VERR = Verify write access:	00001111 | 00000000 | mod101r/m		14,16*	1	9,11,16

Shaded areas indicate instructions not available in 8086, 88 microsystems.

Footnotes

The Effective Address (EA) of the memory operand is computed according to the mod and r/m fields:

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16 bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low

if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP

DISP follows 2nd byte of instruction (before data if required)

*except if mod = 00 and r/m = 110 then EQ = disp-high: disp-low.

SEGMENT OVERRIDE PREFIX

0	0	1	reg	1	1	0

reg is assigned according to the following:

reg	Segment Register
00	ES
01	CS
10	SS
11	DC

REG is assigned according to the following table:

16-Bit (w = 1)		8-Bit (w = 0)	
000	AX	000	AL
001	CX	001	CL
010	DX	010	DL
011	BX	011	BL
100	SP	100	AH
101	BP	101	CH
101	SI	110	DH
111	DI	111	BH

The physical addresses of all operands addressed by the BP register are computed using the SS segment register. The physical addresses of the destination operands of the string primitive operations (those addressed by the DI register) are computed using the ES segment, which may not be overridden.

DATA SHEET REVISION REVIEW

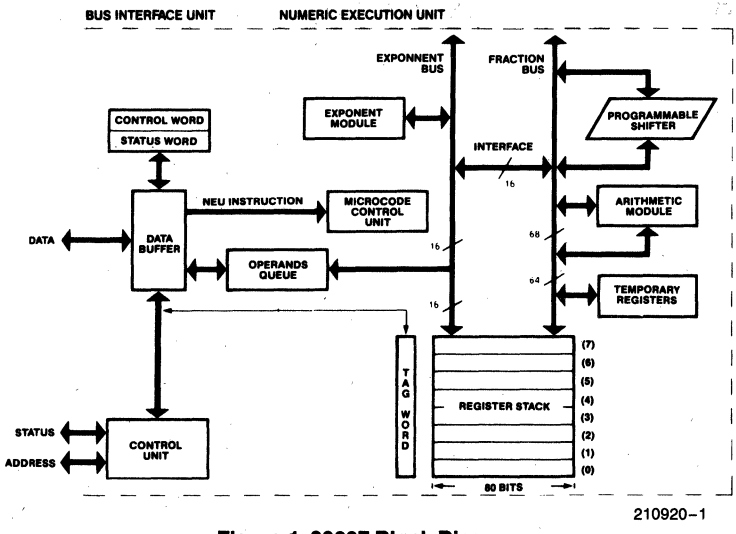
The following list represents key differences between this and the -012 data sheet. Please review this summary carefully.

1. Specifications for the 6 MHz version of the part have been deleted. Intel no longer manufactures an 80286-6.
2. The system diagrams (Figures 31 and 32) have been modified. The circuit which drives the RES input of the 82C284 has been modified in order to allow the 82C284 to correctly generate a system reset signal. See the 82C284 data sheet (Order No. 210453) for further information.

80287
80-BIT HMOS
NUMERIC PROCESSOR EXTENSION
(80287-3, 80287-6, 80287-8, 80287-10)

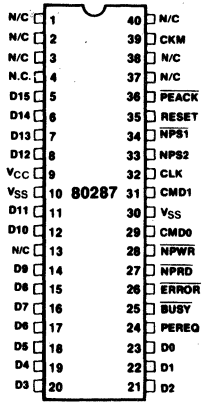
- **High Performance 80-Bit Internal Architecture**
- **Implements Proposed IEEE Floating Point Standard 754**
- **Expands 80286 Data types to Include 32-, 64-, 80-Bit Floating Point, 32-, 64-Bit Integers and 18-Digit BCD Operands**
- **Object Code Compatible with 8087**
- **Built-in Exception Handling**
- **Operates in Both Real and Protected Mode 80286 Systems**
- **8x80-Bit, Individually Addressable, Numeric Register Stack**
- **Protected Mode Operation Completely Conforms to the 80286 Memory Management and Protection Mechanisms**
- **Directly Extends 80286 Instruction Set to Trigonometric, Logarithmic, Exponential and Arithmetic Instructions for All Data types**
- **Operates with 80386 CPU without Software Modification**
- **Available in EXPRESS—Standard Temperature Range**
- **Available in 40 pin-CERDIP package**
  (see Packaging Spec: Order #231369)

The Intel 80287 is a high performance numerics processor extension that extends the 80286 architecture with floating point, extended integer and BCD data types. The 80286/80287 computing system fully conforms to the proposed IEEE Floating Point Standard. Using a numerics oriented architecture, the 80287 adds over fifty mnemonics to the 80286/80287 instruction set, making the 80286/80287 a complete solution for high performance numeric processing. The 80287 is implemented in N-channel, depletion load, silicon gate technology (HMOS) and packaged in a 40-pin cerdip package. The 80286/80287 is object code compatible with the 8086/8087 and 8088/8087.



210920-1

Figure 1. 80287 Block Diagram



210920-2

NOTE:
N/C Pins should not be connected

Figure 2.
80287 Pin Configuration

September 1987
Order Number: 210920-007

Table 1. 80287 Pin Description

Symbols	Type	Name and Functon
CLK	I	**CLOCK INPUT:** this clock provides the basic timing for internal 80287 operations. Special MOS level inputs are required. The 82284 or 8284A CLK outputs are compatible to this input.
CKM	I	**CLOCK MODE SIGNAL:** indicates whether CLK input is to be divided by 3 or used directly. A HIGH input will cause CLK to be used directly. This input must be connected to V_{CC} or V_{SS} as appropriate. This input must be either HIGH or LOW 20 CLK cycles before RESET goes LOW.
RESET	I	**SYSTEM RESET:** causes the 80287 to immediately terminate its present activity and enter a dormant state. RESET is required to be HIGH for more than 4 80287 CLK cycles. For proper initialization the HIGH-LOW transition must occur no sooner than 50 μs after V_{CC} and CLK meet their D.C. and A.C. specifications.
D15–D0	I/O	**DATA:** 1-bit bidirectional data bus. Inputs to these pins may be applied asynchronous to the 80287 clock.
$\overline{\text{BUSY}}$	O	**BUSY STATUS:** asserted by the 80287 to indicate that it is currently executing a command.
$\overline{\text{ERROR}}$	O	**ERROR STATUS:** reflects the ES bit of the status word. This signal indicates that an unmasked error condition exists.
PEREQ	O	**PROCESSOR EXTENSION DATA CHANNEL OPERAND TRANSFER REQUEST:** a HIGH on this output indicates that the 80287 is ready to transfer data. PEREQ will be disabled upon assertion of $\overline{\text{PEACK}}$ or upon actual data transfer, whichever occurs first, if no more transfers are required.
$\overline{\text{PEACK}}$	I	**PROCESSOR EXTENSION DATA CHANNEL OPERAND tRANSFER ACKNOWLEDGE:** acknowledges that the request signal (PEREQ) has been recognized. Will cause the request (PEREQ) to be withdrawn in case there are no more transfers required. $\overline{\text{PEACK}}$ may be asynchronous to the 80287 clock.
$\overline{\text{NPRD}}$	I	**NUMERIC PROCESSOR READ:** Enables transfer of data from the 80287. This input may be asynchronous to the 80287 clock.
$\overline{\text{NPWR}}$	I	**NUMERIC PROCESSOR READ:** Enables transfer of data from the 80287. This input may be asynchronous to the 80287 clock.
$\overline{\text{NPS1}}$, NPS2	I	**NUMERIC PROCESSOR SELECTS:** indicate the CPU is performing an ESCAPE instruction. Concurrent assertion of these signals (i.e., $\overline{\text{NPS1}}$ is LOW and NPS2 is HIGH) enables the 80287 to perform floating point instrucctions. No data transfers involving the 80287 will occur unless the device is selected via these lines. These inputs may be asynchronous to the 80287 clock.
CMD1, CMD0	I	**COMMAND LINES:** These, along with select inputs, allow the CPU to direct the operation of the 80287. These inputs may be asynchronous to the 80287 clock.

Table 1. 80187 Pin Description (Continued)

Symbols	Type	Name and Function
V_{SS}	I	System ground, both pins must be connected to ground.
V_{CC}	I	+5V supply

FUNCTIONAL DESCRIPTION

The 80287 Numeric Processor Extension (NPX) provides arithmetic instructions for a variety of numeric data types in 80286/80287 systems. It also executes numerous built-in transcendental functions (e.g., tangent and log functions). The 80287 executes instructions in parallel with an 80286. It effectively extends the register and instruction set of an 80286 system for existing 80286 data types and adds several new data types as well. Figure 3 presents the program visible register model of the 80286/80287. Essentially, the 80287 can be treated as an additional resource or an extension to the 80286 that can be used as a single unified system, the 80286/80287.

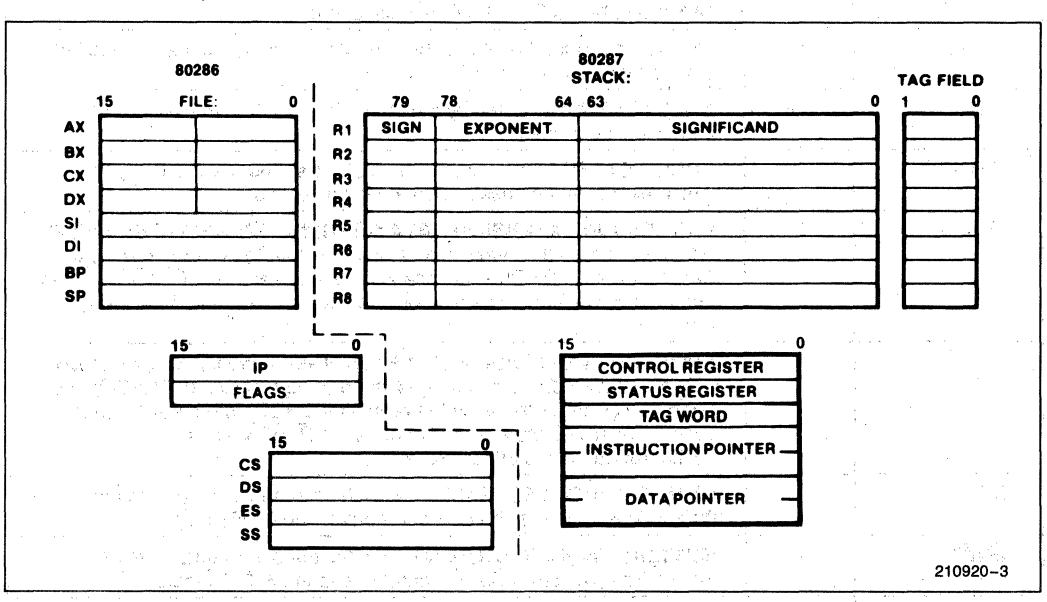


Figure 3. 80286/80287 Architecture

The 80287 has two operating modes similar to the two modes of the 80286. When reset, 80287 is in the real address mode. It can be placed in the protected virtual address mode by executing the SETPM ESC instruction. The 80287 cannot be switched back to the real address mode except by reset. In the real address mode, the 80286/80287 is completely software compatible with 8086/8087 and 8088/8087.

Once in protected mode, all references to memory for numerics data or status information, obey the 80286 memory management and protection rules giving a fully protected extension of the 80286 CPU. In the protected mode, 80286/80287 numerics software is also completely compatible with 8086/8087 and 8088/8087.

SYSTEM CONFIGURATION WITH 80286

As a processor extension to an 80286, the 80287 can be connected to the CPU as shown in Figure 4A. The data channel control signals (PEREQ, PEACK), the BUSY signal and the NPRD, NPWR signals, allow the NPX to receive instructions and data from the CPU. When in the protected mode, all information received by the NPX is validated by the 80286 memory management and protection unit. Once started, the 80287 can process in parallel with and independent of the host CPU. When the NPX detects an error or exception, it will indicate this to the CPU by asserting the ERROR signal.

The NPX uses the processor extension request and acknowledge pins of the 80286 CPU to implement data transfers with memory under the protection model of the CPU. The full virtual and physical address space of the 80286 is available. Data for the 80287 in memory is addressed and represented in the same manner as for an 8087.

The 80287 can operate either directly from the CPU clock or with a dedicated clock. For operation with the CPU clock (CKM = 0), the 80287 works at one-third the frequency of the system clock (i.e., for an 8 MHz 80286, the 16 MHz system clock is divided down to 5.3 MHz). The 80287 provides a capability to internally divide the CPU clock by three to produce the required internal clock (33% duty cycle). To use a higher performance 80287 (8 MHz), an 8284A clock driver and appropriate crystal may be used to directly drive the 80287 with a ⅓ duty cycle clock on the CLK input (CKM = 1). The following table describes the relationship between the clock speed and the 287 speed version needed as a function of the CKM state.

287 Speed Version	CLK Speed	
	CKM = 0	CKM = 1
5 MHz	12 MHz	5 MHz
6 MHz	16 MHz	6 MHz
8 MHz	20 MHz	8 MHz
10 MHz	25 MHz	10 MHz

SYSTEM CONFIGURATION WITH 80386

The 80287 can also be connected as a processor extension to the 80386 CPU as shown in Figure 4b. All software written for 8086/8087 and 80286/80287 is object code compatible with 80386/80287 and can benefit from the increased speed of the 80386 CPU.

Note that the PEACK input pin is pulled high. This is because the 80287 is not required to keep track of the number of words transferred during an operand transfer when it is connected to the 80386 CPU. Unlike the 80286 CPU, the 80386 CPU knows the exact length of the operand being transferred to/from the 80287. After an ESC instruction has been sent to the 80287, the 80386 processor extension data channel will initiate the data transfer as soon as it receives the PEREQ signal from the 80287. The transfer is automatically terminated by the 80386 CPU as soon as all the words of the operand have been transferred.

Because of the very high speed local local bus of the 80386 CPU, the 80287 cannot reside directly on the CPU local bus. A local bus controller logic is used to generate the necessary read and write cycle timings as well as the chip select timings for the 80287. The 80386 CPU uses I/O addresses 800000F8 through 800000FF to communicate with the 80287. This is beyond the normal I/O address space of the CPU and makes it easier to generate the chip select signals using A31 and M/IO. It may also be noted that the 80386 CPU automatically generates 16-bit bus cycles whenever it communicates with the 80287.

HARDWARE INTERFACE

Communication of instructions and data operands between the 80286 and 80287 is handled by the CMD0, CMD1, NPS1, NPS2, NPRD, and NPWR signals. I/O port addresses 00F8H, 00FAH, and 00FCH are used by the 80286 for this communication. When any of these addresses are used, the NPS1 input must be LOW and NPS2 input HIGH. The IORC and IOWC outputs of the 82288 identify I/O space transfers (see Figure 4A). CMD0 should be connected to latched 80286 A1 and CMD1 should be connected to latched 80286 A2.

I/O ports 00F8H to 00FFH are reserved for the 80286/80287 interface. To guarantee correct operation of the 80287, programs must not perform any I/O operations to these ports.

The PEREQ, PEACK, BUSY, and ERROR signals of the 80287 are connected to the same-named 80286 input. The data pins of the 80287 should be directly connected to the 80286 data bus. Note that all bus drivers connected to the 80286 local bus must be inhibited when the 80286 reads from the 80287. The use of M/IO in the decoder prevents INTA bus cycles from disabling the data transceivers.

PROGRAMMING INTERFACE

Table 2 lists the seven data types the 80287 supports and presents the format for each type. These

values are stored in memory with the least significant digits at the lowest memory address. Programs retrieve these values by generating the lowest address. All values should start at even addresses for maximum system performance.

Internally the 80287 holds all numbers in the temporary real format. Load instructions automatically convert operands represented in memory as 16-, 32-, or 64-bit integers, 32- or 64-bit floating point number or 18-digit packed BCD numbers into temporary real format. Store instructions perform the reverse type conversion.

80287 computations use the processor's register stack. These eight 80-bit registers provide the equivalent capacity of 40 16-bit registers. The 80287 register set can be accessed as a stack, with instructions operating on the top one or two stack elements, or as a fixed register set, with instructions operating on explicitly designated registers.

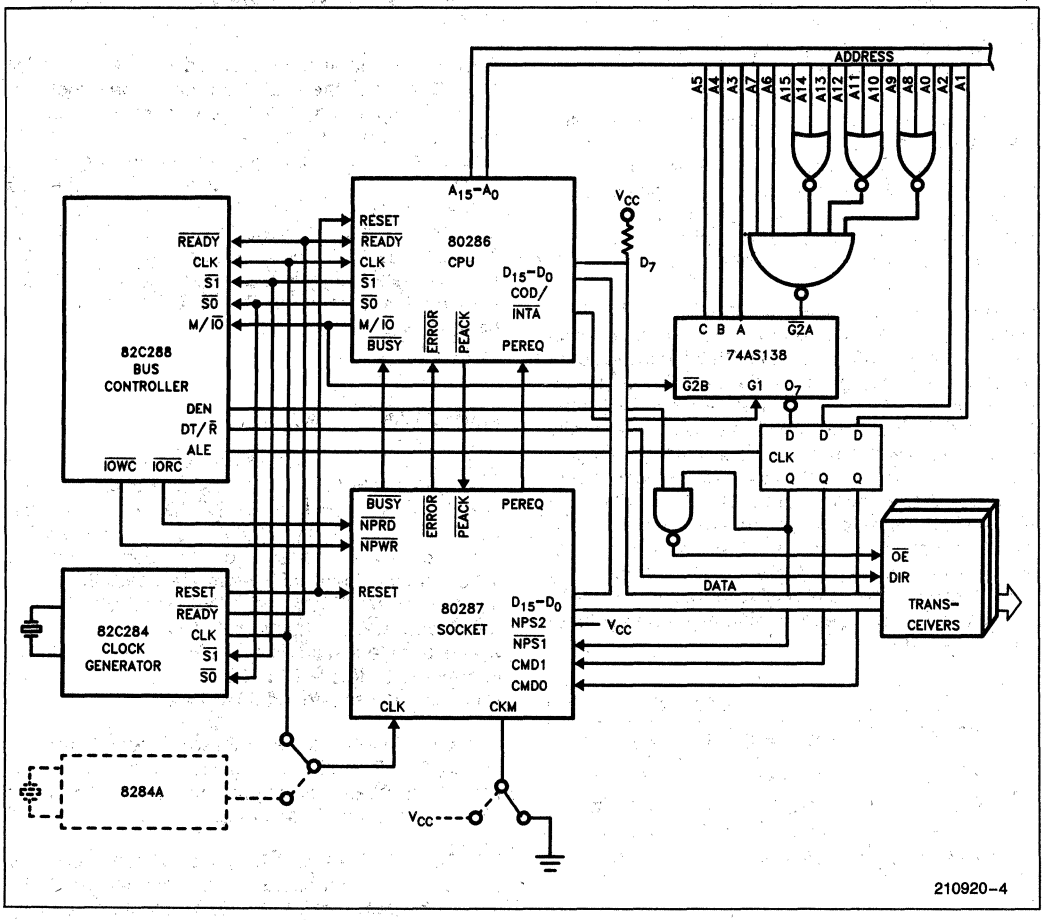


Figure 4A. 80286/80287 System Configuration

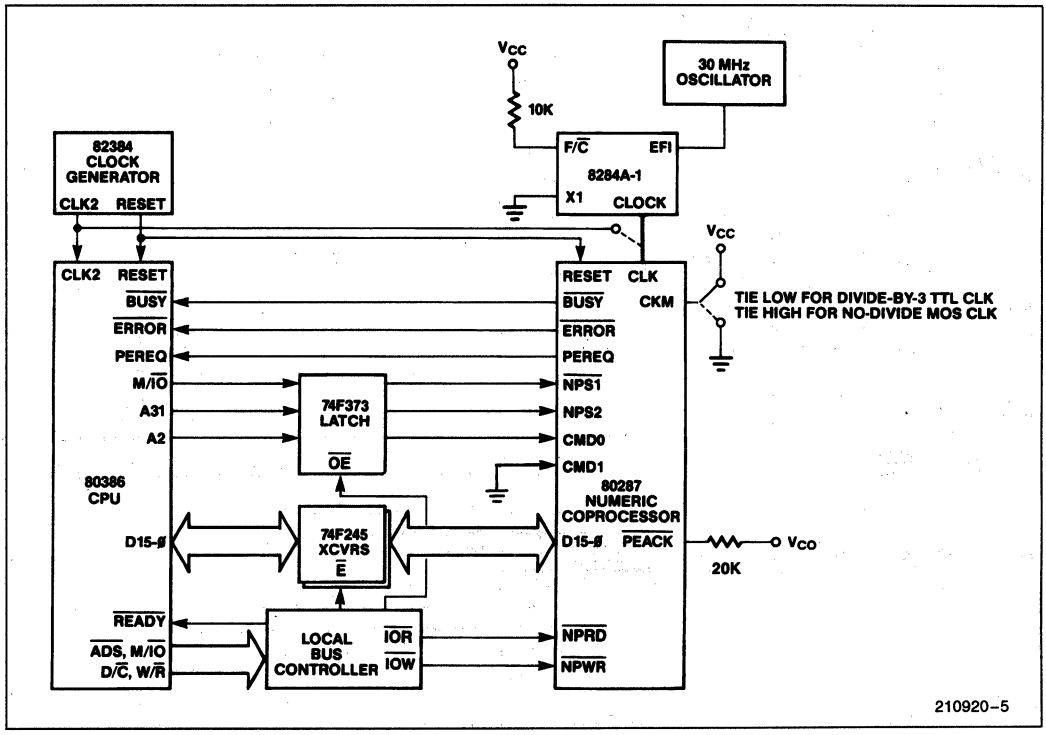
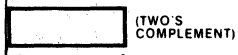
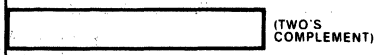
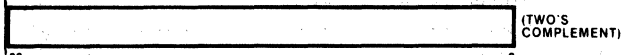
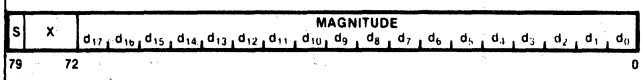
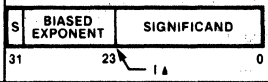
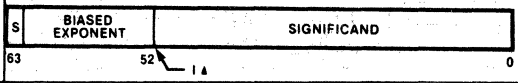
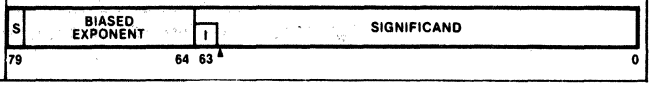


Figure 4B. 80386/80287 System Configuration

Table 2. 80287 Data Type Representation in Memory

Data Formats	Range	Precision	Most Significant Byte ... HIGHEST ADDRESSED BYTE
Word Integer	10^4	16 Bits	
Short Integer	10^9	32 Bits	
Long Integer	10^{19}	64 Bits	
Packed BCD	10^{18}	18 Digits	
Short Real	$10^{\pm 38}$	24 Bits	
Long Real	$10^{\pm 308}$	53 Bits	
Temporary Real	$10^{\pm 4932}$	64 Bits	

210920-6

NOTES:
1. S = Sign bit (0 = positive, 1 = negative)
2. d_n = Decimal digit (two per byte)
3. X = Bits have no significance; 8087 ignores when loading, zeros when storing.
4. ▲ = Position of implicit binary point
5. I = Integer bit of significant; stored in temporary real, implicit in short and long real.
6. Exponent Bias (normalized values):
   Short Real: 127 (7FH)
   Long Real: 1023 (3FFH)
   Temporary Real: 16383 (3FFFH)
7. Packed BCD: $(-1)^S (D_{17} \ldots D_0)$
8. Real: $(-1)^S (2^{E\text{-BIAS}})(F_0 F_1 \ldots)$

Table 6 lists the 80287's instructions by class. No special programming tools are necessary to use the 80287 since all new instructions and data types are directly supported by the 80286 assembler and appropriate high level languages. All 8086/8088 development tools which support the 8087 can also be used to develop software for the 80286/80287 in real address mode.

SOFTWARE INTERFACE

The 80286/80287 is programmed as a single processor. All communication between the 80286 and the 80287 is transparent to software. The CPU automatically controls the 80287 whenever a numeric instruction is executed. All memory addressing modes, physical memory, and virtual memory of the CPU are available for use by the NPX.

Since the NPX operates in parallel with the CPU, any errors detected by the NPX may be reported after the CPU has executed the ESCAPE instruction which caused it. To allow identification of the failing numeric instruction, the NPX contains two pointer registers which identify the address of the failing numeric instruction and the numeric memory operand if appropriate for the instruction encountering this error.

INTERRUPT DESCRIPTION

Several interrupts of the 80286 are used to report exceptional conditions while executing numeric programs in either real or protected mode. The interrupts and their functions are shown in Table 3.

PROCESSOR ARCHITECTURE

As shown in Figure 1, the NPX is internally divided into two processing elements, the bus interface unit (BIU) and the numeric execution unit (NEU). The NEU executes all numeric instructions, while the BIU receives and decodes instructions, requests operand transfers to and from memory and executes processor control instructions. The two units are able to operate independently of one another allowing the BIU to maintain asynchronous communication with the CPU while the NEU is busy processing a numeric instruction.

BUS INTERFACE UNIT

The BIU decodes the ESC instruction executed by the CPU. If the ESC code defines a math instruction, the BIU transmits the formatted instruction to the NEU. If the ESC code defines an administrative instruction, the BIU executes it independently of the NEU. The parallel operation of the NPX with the CPU is normally transparent to the user. The BIU generates the $\overline{\text{BUSY}}$ and $\overline{\text{ERROR}}$ signals for 80826/80287 processor synchronization and error notification, respectively.

The 80287 executes a single numeric instruction at a time. When executing most ESC instructions, the

Table 3. 80286 Interrupt Vectors Reserved for NPX

Interrupt Number	Interrupt Function
7	An ESC instruction was encountered when EM or TS of the 80286 MSW was set. EM = 1 indicates that software emulation of the instruction is required. When TS is set, either an ESC or WAIT instruction will cause interrupt 7. This indicates that the current NPX context may not belong to the current task.
9	The second or subsequent words of a numeric operand in memory exceeded a segment's limit. This interrupt occurs after executing an ESC instruction. The saved return address will not point at the numeric instruction causing this interrupt. After processing the addressing error, the 80286 program can be restarted at the return address with IRET. The address of the failing numeric instruction and numeric operand and saved in the 80287. An interrupt handler for this interrupt *must* execute FNINIT before *any* other ESC or WAIT instruction.
13	The starting address of a numeric operand is not in the segment's limit. The return address will point at the ESC instruction, including prefixes, causing this error. The 80287 has not executed this instruction. The instruction and data address is 80287 refer to a previous, correctly executed, instruction.
16	The previous numeric instruction caused an unmasked numeric error. The address of the faulty numeric instruction or numeric data operand is stored in the 80287. Only ESC or WAIT instructions can cause this interrupt. The 80286 return address will point at a WAIT or ESC instruction, including prefixes, which may be restarted after clearing the error condition in the NPX.

80286 tests the $\overline{\text{BUSY}}$ pin and waits until the 80287 indicates that it is not busy before initiating the command. Once initiated, the 80286 continues program execution while the 80287 executes the ESC instruction. In 8086/8087 systems, this synchronization is achieved by placing a WAIT instruction before an ESC instruction. For most ESC instructions, the 80287 does not require a WAIT instruction before the ESC opcode. However, the 80287 will operate correctly with these WAIT instruction. In all cases, a WAIT or ESC instruction should be inserted after any 80287 store to memory (except FSTSW and FSTCW) or load from memory (except FLDENV or FRSTOR) before the 80286 reads or changes the value to be sure the numeric value has already been wrtten or read by the NPX.

Data transfers between memory and the 80287, when needed, are controlled by the PEREQ $\overline{\text{PEACK}}$, $\overline{\text{NPRD}}$, $\overline{\text{NPWR}}$, $\overline{\text{NPS1}}$, NPS2 signals. The 80286 does the actual data transfer with memory through its processor extension data channel. Numeric data transfers with memory performed by the 80286 use the same timing as any other bus cycle. Control signal for the 80287 are generated by the 80826 as shown in Figure 4a, and meet the timing requirements shown in the AC requirements section.

NUMERIC EXECUTION UNIT

The NEU executes all instructions that involve the register stack; these include arithmetic, logical, transcendental, constant and data transfer instructions. The data path in the NEU is 84 bits wide (68 significand bits, 15 exponent bits and a sign bit) which allows internal operand transfers to be performed at very high speeds.

When the NEU begins executing an instruction, it activated the BIU $\overline{\text{BUSY}}$ signal. This signal is used in conjunction with the CPU WAIT instruction or automatically with most of the ESC instructions to synchronize both processors.

REGISTER SET

The 80287 register set is shown in Figure 5. Each of the eight data registers in the 8087's register stack

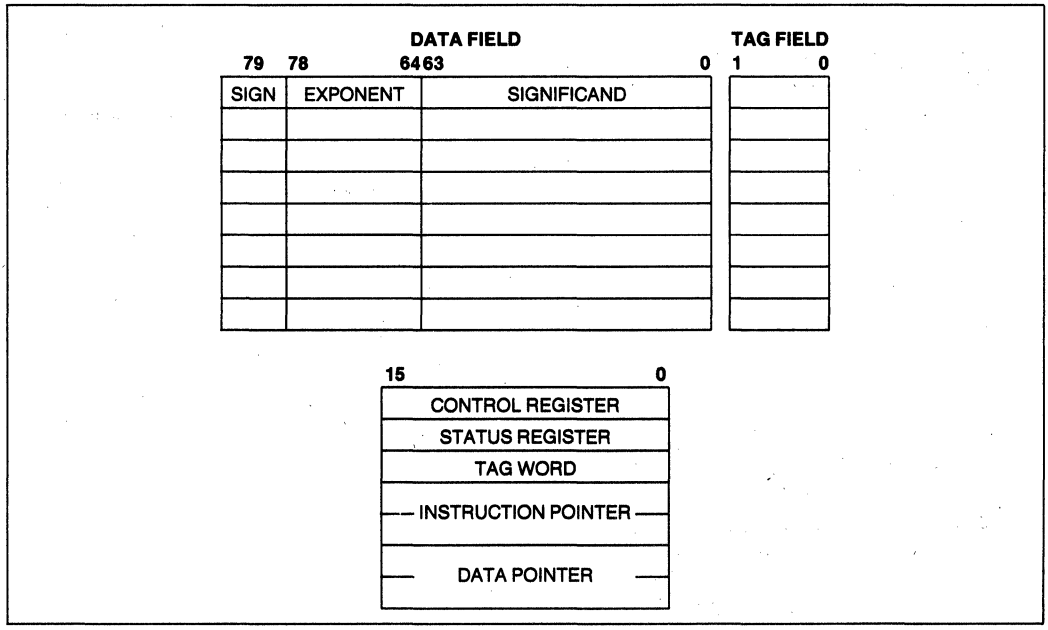


Figure 5. 80287 Register Set

is 80 bits wide and is divided into "fields" corresponding to the NPX's temporary real data type.

At a given point in time the TOP field in the status word identifies the current top-of-stack register. A "push" operation decrements TOP by 1 and loads a value into the new top register. A "pop" operation stores the value from the current top register and then increments TOP by 1. Like 80286 stacks in memory, the 80287 register stack grows "down" toward lower-addressed registers.

Instructions may address the data registers either implicitly or explicitly. Many instructions operate on the register at the TOP of the stack. These instructions implicitly address the register pointed by the TOP. Other instructions allow the programmer to explicitly specify the register which is to be used. This explicit register addressing is also "top-relative."

STATUS WORD

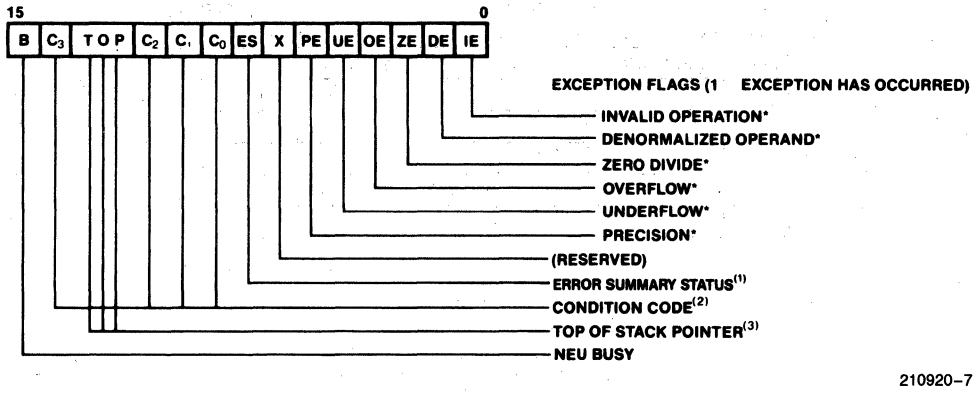
The 16-bit status word (in the status register) shown in Figure 6 reflects the overall state of the 80287. It may be read and inspected by CPU code. The busy bit (bit 15) indicates whether the NEU is executing an instruction (B = 1) or is idle (B = 0).

The instructions FSTSW, FSTSW AX, FSTENV, and FSAVE which store the status word are executed exclusively by the BIU and do not set the busy bit themselves or require the Busy bit be cleared in order to be executed.

The four numeric condition code bits (C_0–C_3) are similar to the flags in a CPU: instructions that perform arithmetic operations update these bits to reflect the outcome of NPX operations. The effect of these instructions on the condition code is summarized in Tables 4a and 4b.

Bits 14–12 of the status word point to the 80287 register that is the current top-of-stack (TOP) as described above. Figure 6 shows the six error flags in bits 5–0 of the status word. Bits 5–0 are set to indicate that the NEU has detected an exception while executing an instruction. The section on exception handling explains how they are set and used.

Bit 7 is the error summary status bit. This bit is set if any unmasked exception bit is set and cleared otherwise. If this bit is set, the ERROR signal is asserted.



NOTES:
1. ES is set if any unmasked exception bit is set, cleared otherwise.
2. See Table 5 for condition code interpretation.
3. Top Values
   000 = Register 0 is Top of Stack
   001 = Register 1 is Top of Stack
   •
   •
   •
   111 = Register 7 is Top of Stack.

*For definitions, see the section on exception handling

Figure 6. 80287 Status Word

TAG WORD

The tag word marks the content of each register as shown in Figure 7. The principal function of the tag word is to optimize the NPX's performance. The eight two-bit tags in the tag word can be used, however, to interpret the contents of 80287 registers.

INSTRUCTION AND DATA POINTERS

The instruction and data pointers (See Figures 8a and 8b) are provided for user-written error handlers. Whenever the 80287 executes a new instruction, the BIU saves the instruction address, the operand address (if present) and the instruction opcode. 80287 instructions can store this data into memory.

The instruction and data pointers appear in one of two formats depending on the operating mode of the 80287. In real mode, these values are the 20-bit physical address and 11-bit opcode formatted like the 8087. In protection mode, these values are the 32-bit virtual address used by the program which executed an ESC instruction. The same FLDENV/FSTENV/FSAVE/FRSTOR instructions as those of the 8087 are used to transfer these values between the 80287 registers and memory.

The saved instruction address in the 80287 will point at any prefixes which preceded the instruction. This is different than in the 8087 which only pointed at the ESCAPE instruction opcode.

CONTROL WORD

The NPX provides several processing options which are selected by loading a word from memory into the control word. Figure 9 shows the format and encoding of fields in the control word.

The low order byte of this control word configures the 80287 error and exception masking. Bits 5–0 of the control word contain individual masks for each of the six exceptions that the 80287 recognizes. The high order byte of the control word configures the

Table 4a. Condition Code Interpretation

Instruction Type	C_3	C_2	C_1	C_0	Interpretation
Compare, Test	0	0	X	0	ST > Source or 0 (FTST)
	0	0	X	1	ST < Source or 0 (FTST)
	1	0	X	0	ST = Source or 0 (FTST)
	1	1	X	1	ST is not comparable
Remainder	Q_1	0	Q_0	Q_2	Complete reduction with three low bits of quotient (See Table 5b)
	U	1	U	U	Incomplete Reduction
Examine	0	0	0	0	Valid, positive unnormalized
	0	0	0	1	Invalid, positive, exponent = 0
	0	0	1	0	Valid, negative, unnormalized
	0	0	1	1	Invalid, negative, exponent = 0
	0	1	0	0	Valid, positive, normalized
	0	1	0	1	Infinity, positive
	0	1	1	0	Valid, negative, normalized
	0	1	1	1	Infinity, negative
	1	0	0	0	Zero, positive
	1	0	0	1	Empty
	1	0	1	0	Zero, Negative
	1	0	1	1	Empty
	1	1	0	0	Invalid, positive, exponent = 0
	1	1	0	1	Empty
	1	1	1	0	Invalid, negative, exponent = 0
	1	1	1	1	Empty

NOTES:
1. ST = Top of Stack
2. X = value is not affected by instruction
3. U = value is undefined following instruction
4. Q_n = Quotient bit n

Table 4b. Condition Code Interpretation after FPREM (See Note 1) Instruction as a Function of Dividend Value

Dividend Range	Q_2	Q_1	Q_0
Dividend < 2 * Modulus	C_3	C_1	Q_0
Dividend < 4 * Modulus	C_3	Q_1	Q_0
Dividend ≥ 4 * Modulus	Q_2	Q_1	Q_0

NOTE:
1. Previous value of indicated bit, not affected by FPREM instruction execution.

80287 operating mode including precision, rounding, and infinity control. The precision control bits (bits 9–8) can be used to set the 80287 internal operating precision at less than the default of temporary real (80-bit) precision. This can be useful in providing compatibility with the early generation arithmetic processors of smaller precision than the 80287. The rounding control bits (bits 11–10) provide for directed rounding and true chop as well as the unbiased round to nearest even mode specified in the IEEE standard. Control over closure of the number space at infinity is also provided (either affine closure: ± ∞, or projective closure: ∞, is treated as unsigned, may be specified).

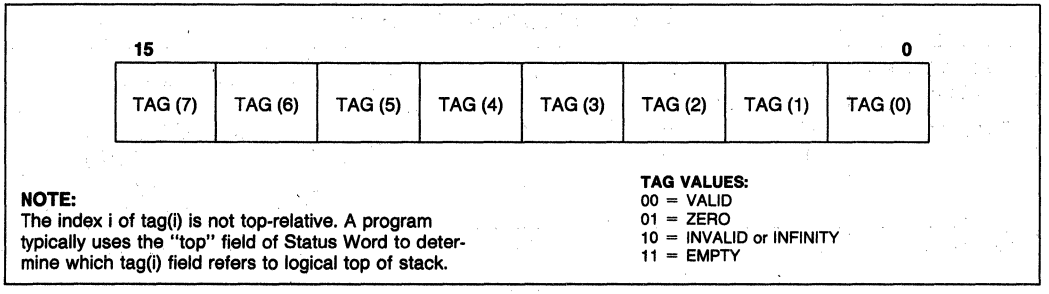


Figure 7. 80287 Tag Word

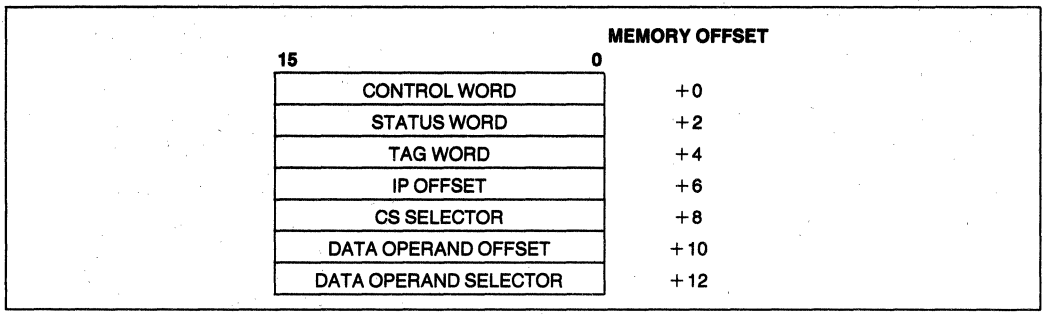


Figure 8a. Protected Mode 80287 Instruction and Data Pointer Image in Memory

EXCEPTION HANDLING

The 80287 detects six different exception conditions that can occur during instruction execution. Any or all exceptions will cause the assertion of external $\overline{\text{ERROR}}$ signal and ES bit of the Status Word if the appropriate exception masks are not set.

The exceptions that the 80287 detects and the 'default' procedures that will be carried out if the exception is masked, are as follows:

Invalid Operation: Stack overflow, stack underflow, indeterminate form (0/0, ∞, $-\infty$, etc) or the use of a Non-Number (NAN) as an operand. An exponent value of all ones and non-zero significand is reserved to identify NANs. If this exception is masked, the 80287 default response is to generate a specific NAN called INDEFINITE, or to propogate already existing NANs as the calculation result.

Overflow: The result is too large in magnitude to fit the specified format. The 80287 will generate an encoding for infinity if this exception is masked.

Zero Divisor: The divisor is zero while the dividend is a non-infinite, non-zero number. Again, the 80287 will generate an encoding for infinity if this exception is masked.

Underflow: The result in non-zero but too small in magnitude to fit in the specified format. If this exception is masked the 80287 will denormalize (shift right) the fraction until the exponent is in range. The process is called gradual underflow.

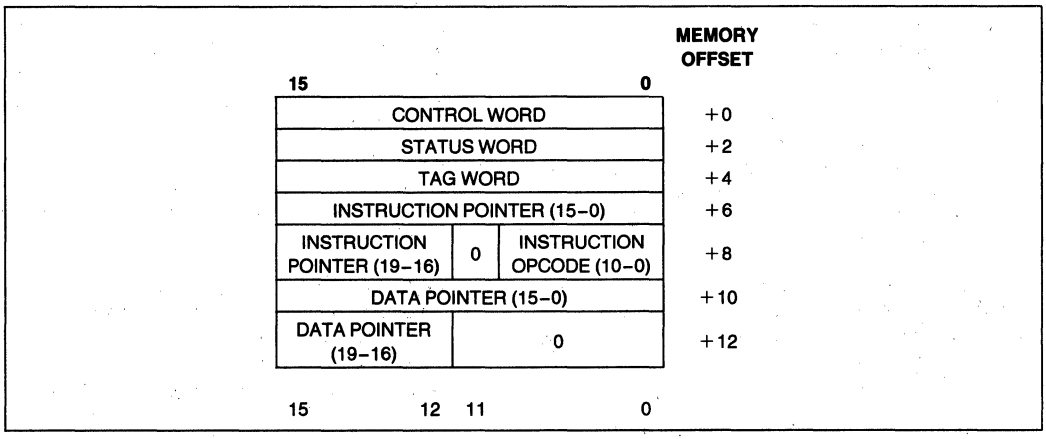


Figure 8b. Real Mode 80287 Instruction and Data Pointer Image in Memory

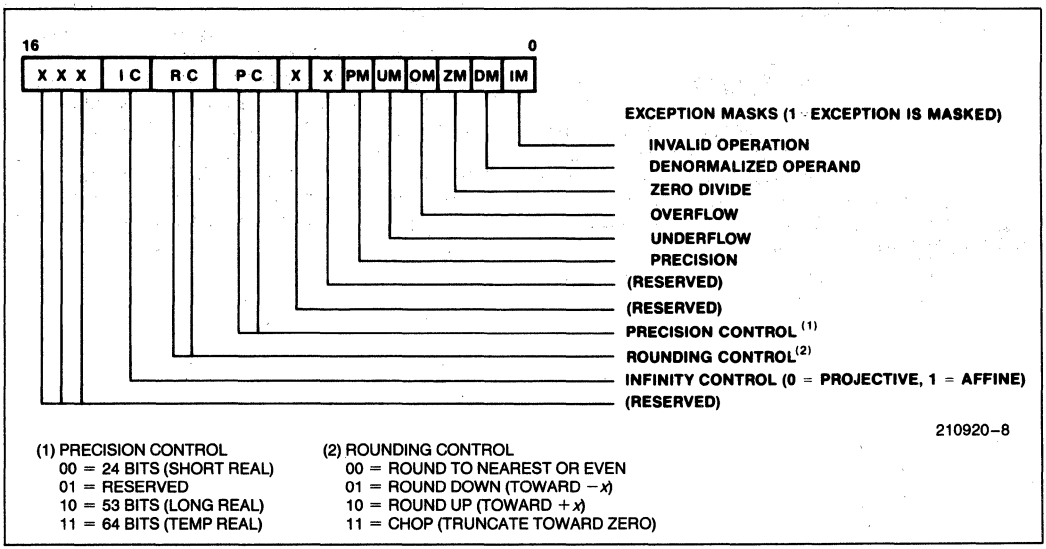


Figure 9. 80287 Control Word

Denormalized Operand: At least one of the operands is denormalized; it has the smallest exponent but a non-zero significand. Normal processing continues if this exception is masked off.

Inexact Result: The true result is not exactly representable in the specified format, the result is rounded according to the rounding mode, and this flag is set. If this exception is masked, processing will simply continue.

If the error is not masked, the corresponding error bit and the error status bit (ES) in the control word will be set, and the $\overline{\text{ERROR}}$ output signal will be asserted. If the CPU attempts to execute another ESC or WAIT instruction, exception 7 will occur.

The error condition must be resolved via an interrupt service routine. The 80287 saves the address of the floating point instruction causing the error as well as the address of the lowest memory location of any memory operand required by that instruction.

8086/8087 COMPATIBILITY:

The 80286/80287 supports portability of 8086/8087 programs when it is in the real address mode. However, because of differences in the numeric error handling techniques, error handling routines *may* need to be changed. The differences between an 80286/80287 and 8086/8087 are:

1. The NPX error signal does not pass through an interrupt controller (8087 INT signal does). Therefore, any interrupt controller oriented instructions for the 8086/8087 may have to be deleted.
2. Interrupt vector 16 must point at the numeric error handler routine.
3. The saved floating point instruction address in the 80287 includes any leading prefixes before the ESCAPE opcode. The corresponding saved address of the 8087 does not include leading prefixes.
4. In protected mode, the format of the saved instruction and operand pointers is different than for the 8087. The instruction opcode is not saved—it must be read from memory if needed.
5. Interrupt 7 will occur when executing ESC instructions with either TS or EM or MSW = 1. If TS of MSW = 1 then WAIT will also cause interrupt 7. An interrupt handler should be added to handle this situation.
6. Interrupt 9 will occur if the second or subsequent words of a floating point operand fall outside a segment's size. Interrupt 13 will occur if the starting address of a numeric operand falls outside a segment's size. An interrupt handler should be added to report these programming errors.

In the protected mode, 8086/8087 application code can be directly ported via recompilation if the 80286 memory protection rules are not violated.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature .....−65°C to +150°C
Case Temperature .....0°C to 85°C
Voltage on any Pin with Respect to Ground .....−1.0 to +7V
Power Dissipation .....3.0 Watt

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS T_A = 0°C to 70°C, T_C = 0°C to 85°C, V_{CC} = 5V ±5%

ALL SPEEDS SELECTIONS

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input LOW Voltage	−0.5	0.8	V	
V_{IH}	Input HIGH Voltage	2.0	V_{CC} +0.5	V	
V_{IHC}	Clock Input HIGH Voltage CKM = 1: CKM = 0:	 2.0 3.8	 V_{CC} +1 V_{CC} +1	 V V	
V_{ILC}	Clock Input LOW Voltage CKM = 1 CKM = 0	 −0.5 −0.5	 0.8 0.6	 V V	
V_{OL}	Output LOW Voltage		0.45	V	I_{OL} = 3.0 mA
V_{OH}	Output HIGH Voltage	2.4		V	I_{OH} = −400 μA
I_{LI}	Input Leakage Current	•	±10	μA	OV ≤ V_{IN} ≤ V_{CC}
I_{LO}	Output Leakage Current	•	±10	μA	0.45V ≤ V_{OUT} ≤ V_{CC}
I_{CC}	Power Supply Current	 •	600 475 375	mA mA mA	T_A = 0°C T_A = 25°C T_A = 70°C
C_{IN}	Input Capacitance	•	10	pF	F_C = MHz
C_O	Input/Output Capacitance (D0–D15)	•	20	pF	V_C = 1 MHz
C_{CLK}	CLK Capacitance	•	12	pF	F_C = 1 MHz

A.C. CHARACTERISTICS T_A = 0°C to 70°C, T_{CASE} = 0°C to 85°C, V_{CC} = 5V ±5%

TIMING REQUIREMENTS

A.C. timings are referenced to 0.8V and 2.0V points on signals unless otherwise noted.

Symbol	Parameter	80287-3 5 MHz		80287-6 6 MHz		80287-8 8 MHz		80287-10 10 MHz Preliminary		Units	Test Conditions
		Min	Max	Min	Max	Min	Max	Min	Max		
T_{CLCL}	CLK Period CKM = 1: CKM = 0:	 200 62.5	 500 250	 166 62.5	 500 166	 125 50	 500 166	 100 40	 500 166	 ns ns	
T_{CLCH}	CLK LOW Time CKM = 1: CKM = 0:	 118 15	 230	 100 15	 343 146	 68 15	 343 146	 62 11	 343 146	 ns ns	 At 0.8V At 0.6V
T_{CHCL}	CLK HIGH Time CKM = 1: CKM = 0:	 69 20	 235	 50 20	 230 151	 43 20	 230 151	 28 18	 230 151	 ns ns	 At 2.0V At 3.6V
T_{CH1CH2}	CLK Rise Time		10		10		10		10	ns	1.0V to 3.6V if CKM = 0
T_{CL2CL1}	CLK Fall Time		10		10		10		10	ns	3.6V to 1.0V if CKM = 0
T_{DYWH}	Data Setup to $\overline{NPWR}$ Inactive	75		75		75		75		ns	
T_{WHDX}	Data Hold from $\overline{NPWR}$ Inactive	30		30		18		18		ns	
T_{WLWH} T_{RLRH}	$\overline{NPWR}$ $\overline{NPRD}$ Active Time	95		95		90		90		ns	At 0.8V
T_{AVWL} T_{AVRL}	Command Valid to $\overline{NPWR}$ or $\overline{NPRD}$Active	0		0		0		0		ns	
T_{MHRL}	Minimum Delay from PEREQ Active to $\overline{NPRD}$ Active	130		130		130		100		ns	
T_{KLKH}	$\overline{PEAK}$ Active Time	85		85		85		60		ns	At 0.8V
T_{KHKL}	$\overline{PEAK}$ Inactive Time	250		250		250		200		ns	At 2.0V
T_{KHCH}	$\overline{PEAK}$ Inactive to $\overline{NPWR}$, $\overline{NPRD}$ Inactive	50		50		40		40		ns	
T_{CHKL}	$\overline{NPWR}$, $\overline{NPRD}$ Inactive to $\overline{PEAK}$ Active	−30		−30		−30		−30		ns	
T_{WHAX} T_{RHAX}	Command Hold from $\overline{NPWR}$, $\overline{NPRD}$ Inactive	30		30		30		22		ns	
T_{KLCL}	$\overline{PEAK}$ Active Setup to $\overline{NPWR}$ $\overline{NPRD}$ Active	50		50		40		40		ns	

A.C. CHARACTERISTICS T_A = 0°C to 70°C, T_{CASE} = 0°C to 85°C, V_{CC} = 5V ±5% (Continued)

TIMING REQUIREMENTS (Continued)

A.C. timings are referenced to 0.8V and 2.0V points on signals unless otherwise noted.

Symbol	Parameter	80287-3 5 MHz		80287-6 6 MHz		80287-8 8 MHz		80287-10 10 MHz Preliminary		Units	Test Conditions
		Min	Max	Min	Max	Min	Max	Min	Max		
T_{IVCL}	$\overline{NPWR}$, $\overline{NPRD}$ to CLK Setup Time	70		70		70		53		ns	(Note 1)
T_{CLIH}	$\overline{NPWR}$, $\overline{NPRD}$ from CLK Hold Time	45		45		45		37		ns	(Note 1)
T_{RSCL}	RESET to CLK Setup Time	20		20		20		20		ns	(Note 1)
T_{CLRS}	RESET from CLK Hold Time	20		20		20		20		ns	(Note 1)

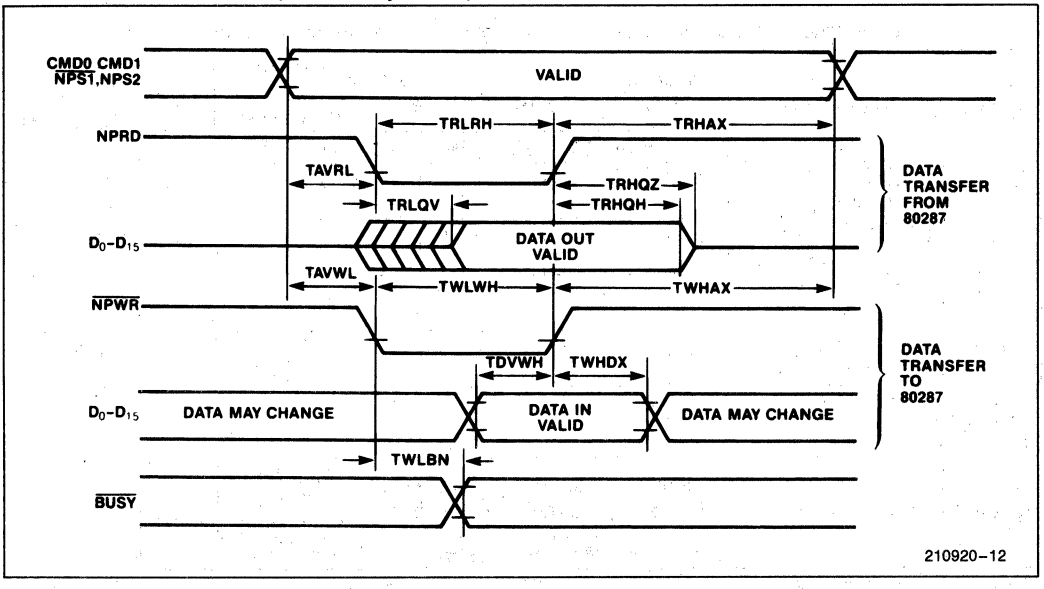
TIMING RESPONSES

Symbol	Parameter	80287-3 5 MHz		80287-6 6 MHz		80287-8 8 MHz		80287-10 10 MHz Preliminary		Units	Test Conditions
		Min	Max	Min	Max	Min	Max	Min	Max		
T_{RHQZ}	$\overline{NPRD}$ Inactive to Data Float		37.5		37.5		35		21	ns	(Note 2)
T_{RLOV}	$\overline{NPRD}$ Active to Data Valid		60		60		60		60	ns	(Note 3)
T_{ILBH}	$\overline{ERROR}$ Active to $\overline{BUSY}$ Inactive	100		100		100		100		ns	(Note 4)
T_{WLBV}	$\overline{NPWR}$ Active to $\overline{BUSY}$ Active		100		100		100		100	ns	(Note 5)
T_{KLML}	$\overline{PEAK}$ Active to $\overline{PEREQ}$ Inactive		127		127		127		100	ns	(Note 6)
T_{CMDI}	Command Inactive Time										
	Write-to-Write	95		95		95		75		ns	At 2.0V
	Read-to-Read	250		95		95		75		ns	At 2.0V
	Write-to-Read	105		95		95		75		ns	At 2.0V
	Read-to-Write	95		95		95		75		ns	At 2.0V
T_{RHQH}	Data Hold from $\overline{NPRD}$ Inactive	5		3		3		3		ns	(Note 7)

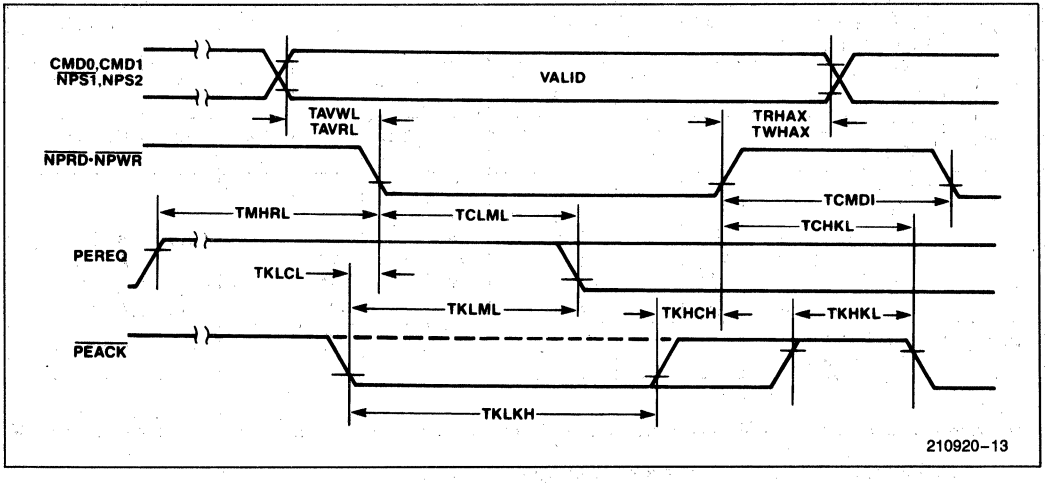
NOTES:
1. This is an asynchronous input. This specification is given for testing purposes only, to assure recognition at a specific CLK edge.
2. Float condition occurs when output current is less than I_{LO} on D0–D15.
3. D0–D15 loSINFϕ: XL = 100 pF.
4. $\overline{BUSY}$ loading: CL = 100 pF.
5. $\overline{BUSY}$ loading: CL = 100 pF.
6. On last data transfer on numeric instruction.
7. D0–D15 loading: CL = 100 pF.

WAVEFORMS

DATA TRANSFER TIMING (Initiated by 80286)

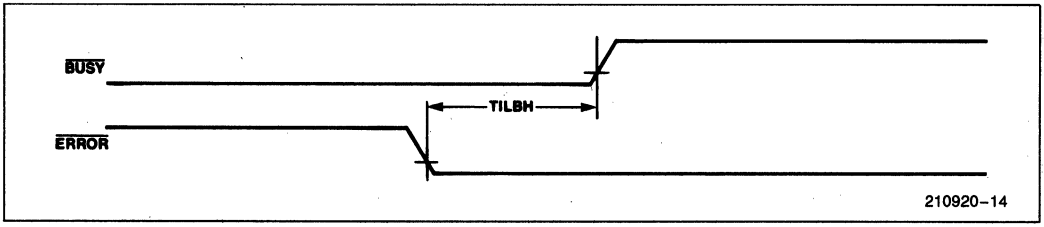


DATA CHANNEL TIMING (Initiated by 80287)

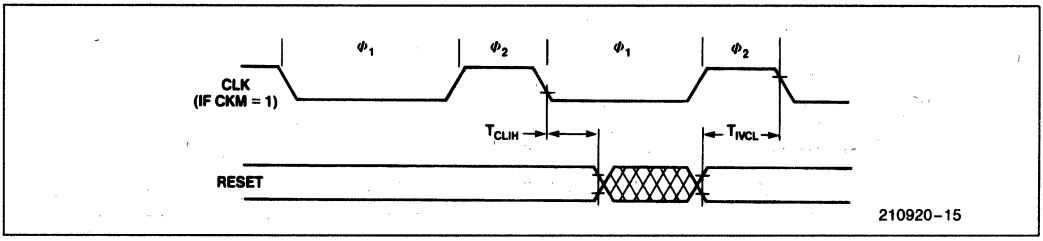


WAVEFORMS (Continued)

ERROR OUTPUT TIMING



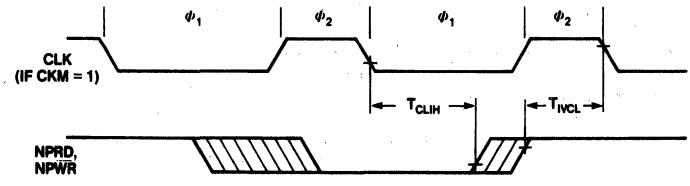
CLK, RESET TIMING (CKM = 1)



NOTE:
Reset, $\overline{\text{NPWR}}$, $\overline{\text{NPRD}}$ are inputs asynchronous to CLK. Timing requirements on this page are given for testing purposes only, to assure recognition at a specific CLK edge.

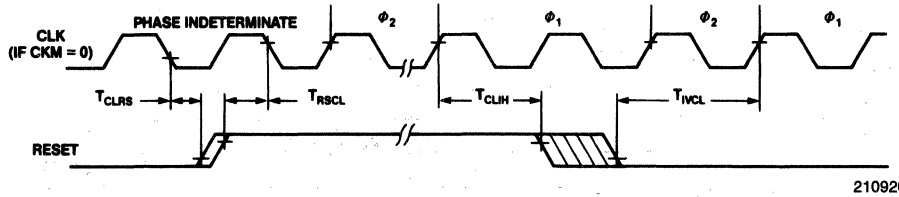
WAVEFORMS (Continued)

CLK, $\overline{NPRD}$, $\overline{NPWR}$ TIMING (CKM = 1)



210920-16

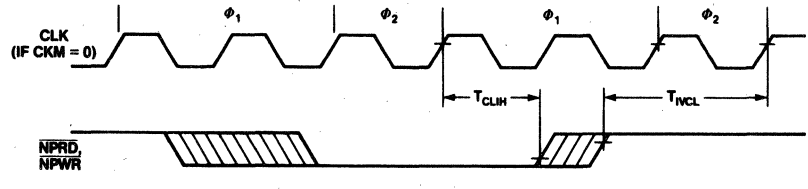
CLK, RESET TIMING (CKM = 0)



210920-17

NOTE:
Reset must meet timing shown to guarantee known phase of internal ÷3 circuit.

CLK, $\overline{NPRD}$, $\overline{NPWR}$ TIMING (CKM = 0)

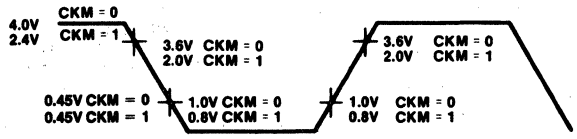


210920-18

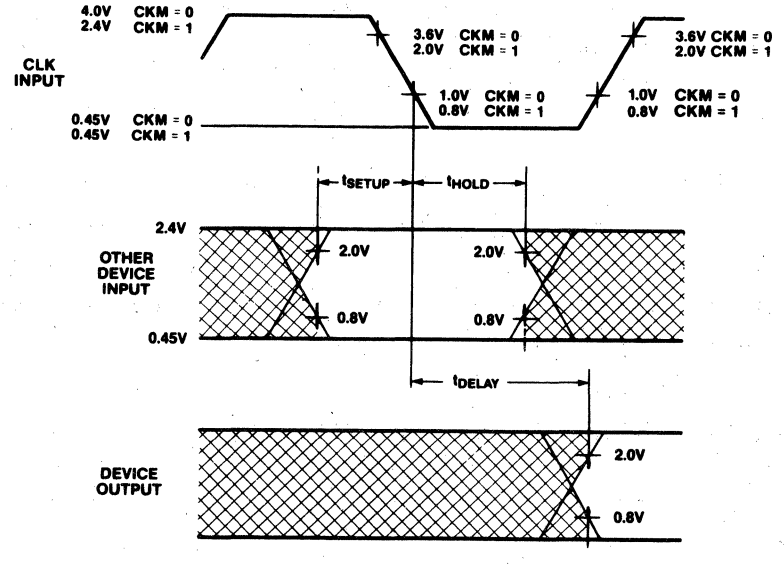
Table 6. 80287 Extensions to the 80286 Instruction Set

Data Transfer		Optional 8,16 Bit Displacement	Clock Count Range 32 Bit Real	32 Bit Integer	64 Bit Real	16 Bit Integer
FLD = LOAD	MF =		00	01	10	11
Integer/Real Memory to ST(0)	ESCAPE MF 1 | MOD 0 0 0 R/M	DISP	38–56	52–60	40–60	46–54
Long Integer Memory to ST(0)	ESCAPE 1 1 1 | MOD 1 0 1 R/M	DISP	60–68			
Temporary Real Memory to ST(0)	ESCAPE 0 1 1 | MOD 1 0 1 R/M	DISP	53–65			
BCD Memory to ST(0)	ESCAPE 1 1 1 | MOD 1 0 0 R/M	DISP	290–310			
ST(i) to ST(0)	ESCAPE 0 0 1 | 1 1 0 0 0 ST(i)		17–22			
FST = STORE						
ST(0) to Integer/Real Memory	ESCAPE MF 1 | MOD 0 1 0 R/M	DISP	84–90	82–92	96–104	80–90
ST(0) to ST(i)	ESCAPE 1 0 1 | 1 1 0 1 0 ST(i)		15–22			
FSTP = STORE AND POP						
ST(0) to Integer/Real Memory	ESCAPE MF 1 | MOD 0 1 1 R/M	DISP	86–92	84–94	98–106	82–92
ST(0) to Long Integer Memory	ESCAPE 1 1 1 | MOD 1 1 1 R/M	DISP	94–105			
ST(0) to Temporary Real Memory	ESCAPE 0 1 1 | MOD 1 1 1 R/M	DISP	52–58			
ST(0) to BCD Memory	ESCAPE 1 1 1 | MOD 1 1 0 R/M	DISP	520–540			
ST(0) to ST(i)	ESCAPE 1 0 1 | 1 1 0 1 1 ST(i)		17–24			
FXCH = Exchange ST(i) and ST(0)	ESCAPE 0 0 1 | 1 1 0 0 1 ST(i)		10–15			
Comparison						
FCOM = Compare						
Integer/Real Memory to ST(0)	ESCAPE MF 0 | MOD 0 1 0 R/M	DISP	60–70	78–91	65–75	72–86
ST(i) to ST (0)	ESCAPE 0 0 0 | 1 1 0 1 0 ST(i)		40–50			
FCOMP = Compare and Pop						
Integer/Real Memory to ST(0)	ESCAPE MF 0 | MOD 0 1 1 R/M	DISP	63–73	80–93	67–77	74–88
ST(i) to ST(0)	ESCAPE 0 0 0 | 1 1 0 1 1 ST(i)		45–52			
FCOMPP = Compare ST(1) to ST(0) and Pop Twice	ESCAPE 1 1 0 | 1 1 0 1 1 0 0 1		45–55			
FTST = Test ST(0)	ESCAPE 0 0 1 | 1 1 1 0 0 1 0 0		38–48			
FXAM = Examine ST(0)	ESCAPE 0 0 1 | 1 1 1 0 0 1 0 1		12–23			

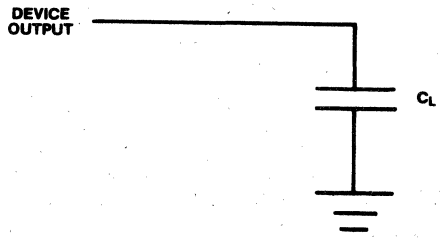
210920-19



AC Drive and Measurement Points—CLK Input



AC Setup, Hold and Delay Time Measurement—General



AC Test Loading on Outputs

Table 6. 80287 Extensions to the 80286 Instruction Set (Continued)

Constants		Optional 8,16 Bit Displacement	Clock Count Range 32 Bit Real	32 Bit Integer	64 Bit Real	16 Bit Integer
	MF =		00	01	10	11
FLDZ = LOAD + 0.0 into ST(0)	ESCAPE 0 0 1 | 1 1 1 0 1 1 1 0		11–17			
FLD1 = LOAD + 1.0 into ST(0)	ESCAPE 0 0 1 | 1 1 1 0 1 0 0 0		15–21			
FLDPI = LOAD π into ST(0)	ESCAPE 0 0 1 | 1 1 1 0 1 0 1 1		16–22			
FLDL2T = LOAD $\log_2 10$ into ST(0)	ESCAPE 0 0 1 | 1 1 1 0 1 0 0 1		16–22			
FLDL2E = LOAD $\log_2 e$ into ST(0)	ESCAPE 0 0 1 | 1 1 1 0 1 0 1 0		15–21			
FLDLG2 = LOAD $\log_{10} 2$ into ST(0)	ESCAPE 0 0 1 | 1 1 1 0 1 1 0 0		18–24			
FLDLN2 = LOAD $\log_e 2$ into ST(0)	ESCAPE 0 0 1 | 1 1 1 0 1 1 0 1		17–23			
Arithmetic						
FADD = Addition						
Integer/Real Memory with ST(0)	ESCAPE MF 0 | MOD 0 0 0 R/M	DISP	90–120	108–143	95–125	102–137
ST(i) and ST(0)	ESCAPE d P 0 | 1 1 0 0 0 ST(i)		70–100 (Note 1)			
FSUB = Subtraction						
Integer/Real Memory with ST(0)	ESCAPE MF 0 | MOD 1 0 R R/M	DISP	90–120	108–143	95–125	102–137
ST(i) and ST(0)	ESCAPE d P 0 | 1 1 1 0 R R/M		70–100 (Note 1)			
FMUL = Multiplication						
Integer/Real Memory with ST(0)	ESCAPE MF 0 | MOD 0 0 1 R/M	DISP	110–125	130–144	112–168	124–138
ST(i) and ST(0)	ESCAPE d P 0 | 1 1 0 0 1 R/M		90–145 (Note 1)			
FDIV = Division						
Integer/Real Memory with ST(0)	ESCAPE MF 0 | MOD 1 1 R R/M	DISP	215–225	230–243	220–230	224–238
ST(i) and ST(0)	ESCAPE d P 0 | 1 1 1 1 R R/M		193–203 (Note 1)			
FSQRT = Square Root of ST(0)	ESCAPE 0 0 1 | 1 1 1 1 1 0 1 0		180–186			
FSCALE = Scale ST(0) by ST(1)	ESCAPE 0 0 1 | 1 1 1 1 1 1 0 1		32–38			
FPREM = Partial Remainder of ST(0) ÷ ST(1)	ESCAPE 0 0 1 | 1 1 1 1 1 0 0 0		15–190			
FRNDINT = Round ST(0) to Integer	ESCAPE 0 0 1 | 1 1 1 1 1 1 0 0		16–50			

210920-20

NOTE:
1. If P = 1 then add 5 clocks.

Table 6. 80287 Extensions to the 80286 Instruction Set (Continued)

		Optional 8,16 Bit Displacement	Clock Count Range
FXTRACT = Extract Components of St(0)	ESCAPE 0 0 1 | 1 1 1 1 0 1 0 0		27–55
FABS = Absolute Value of ST(0)	ESCAPE 0 0 1 | 1 1 1 0 0 0 0 1		10–17
FCHS = Change Sign of ST(0)	ESCAPE 0 0 1 | 1 1 1 0 0 0 0 0		10–17
Transcendental			
FPTAN = Partial Tangent of ST(0)	ESCAPE 0 0 1 | 1 1 1 1 0 0 1 0		30–540
FPATAN = Partial Arctangent of ST(0) ÷ ST(1)	ESCAPE 0 0 1 | 1 1 1 1 0 0 1 1		250–800
F2XM1 = $2^{ST(0)} - 1$	ESCAPE 0 0 1 | 1 1 1 1 0 0 0 0		310–630
FYL2X = ST(1) • Log_2 |ST(0)|	ESCAPE 0 0 1 | 1 1 1 1 0 0 0 1		900–1100
FYL2XP1 = ST(1) • Log_2 |ST(0) +1|	ESCAPE 0 0 1 | 1 1 1 1 1 0 0 1		700–1000
Processor Control			
FINIT = Initialize NPX	ESCAPE 0 1 1 | 1 1 1 0 0 0 1 1		2–8
FSETPM = Enter Protected Mode	ESCAPE 0 1 1 | 1 1 1 0 0 1 0 0		2–8
FSTSW AX = Store Control Word	ESCAPE 1 1 1 | 1 1 1 0 0 0 0 0		10–16
FLDCW = Load Control Word	ESCAPE 0 0 1 | MOD 1 0 1 R/M	DISP	7–14
FSTCW = Store Control Word	ESCAPE 0 0 1 | MOD 1 1 1 R/M	DISP	12–18
FSTSW = Store Status Word	ESCAPE 1 0 1 | MOD 1 1 1 R/M	DISP	12–18
FCLEX = Clear Exceptions	ESCAPE 0 1 1 | 1 1 1 0 0 0 1 0		2–8
FSTENV = Store Environment	ESCAPE 0 0 1 | MOD 1 1 0 R/M	DISP	40–50
FLDENV = Load Environment	ESCAPE 0 0 1 | MOD 1 0 0 R/M	DISP	35–45
FSAVE = Save State	ESCAPE 1 0 1 | MOD 1 1 0 R/M	DISP	205–215
FRSTOR = Restore State	ESCAPE 1 0 1 | MOD 1 0 0 R/M	DISP	205–215
FINCSTP = Increment Stack Pointer	ESCAPE 0 0 1 | 1 1 1 1 0 1 1 1		6–12
FDECSTP = Decrement Stack Pointer	ESCAPE 0 0 1 | 1 1 1 1 0 1 1 0		6–12

210920-21

Table 6. 80287 Extensions to the 80286 Instruction Set (Continued)

		Clock Count Range
FFREE = Free ST(i)	ESCAPE 1 0 1 | 1 1 0 0 0 ST(i)	9-16
FNOP = No Operation	ESCAPE 0 0 1 | 1 1 0 1 0 0 0 0	10-16

210920-22

NOTES:
1. if mod = 00 then DISP = 0*, disp-low and disp-high are absent
   if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
   if mod = 10 then DISP = disp-high; disp-low
   if mod = 11 then r/m is treated as an ST(i) field
2. if r/m = 000 then EA = (BX) + (SI) + DISP
   if r/m = 001 then EA = (BX) + (DI) + DISP
   if r/m = 010 then EA = (BP) + (SI) + DISP
   if r/m = 011 then EA = (BP) + (DI) + DISP
   if r/m = 100 then EA = (SI) + DISP
   if r/m = 101 then EA = (DI) + DISP
   if r/m = 110 then EA = (BP) + DISP
   if r/m = 111 then EA = (BX) + DISP
   *except if mod = 000 and r/m = 110 then EA = disp-high; disp-low.
3. MF = Memory Format
   00—32-bit Real
   01—32-bit Integer
   10—64-bit Real
   11—16-bit Integer
4. ST(0) = Current stack top
   ST(i) = i^{th} register below stack top
5. d = Destination
   0—Destination is ST(0)
   1—Destination is ST(i)
6. P = Pop
   0—No pop
   1—Pop ST(0)
7. R = Reverse: When d = 1 reverse the sense of R
   0—Destination (op) Source
   1—Source (op) Destination
8. For **FSQRT:** $-0 \leq ST(0) \leq +\infty$
   For **FSCALE:** $-2^{15} \leq ST(1) < +2^{15}$ and ST(1) integer
   For **F2XM1:** $0 \leq ST(0) \leq 2^{-1}$
   For **FYL2X:** $0 < ST(0) < \infty$
   $-\infty < ST(1) < +\infty$
   For **FYL2XP1:** $0 \leq |ST(0)| < (2 - \sqrt{2})/2$
   $-\infty < ST(1) < \infty$
   For **FPTAN:** $0 \leq ST(0) \leq \pi/4$
   For **FPATAN:** $0 \leq ST(0) < ST(1) < +\infty$
9. ESCAPE bit pattern is 11011.

DATA SHEET REVISION REVIEW

The following list represents the key differences between this and the -006 80287 Data Sheet. Please review the summary carefully.

1. The CLK speed table in the section entitled "SYSTEM CONFIGURATION WITH 80286" was modified to show the required CLK frequencies in the divide-by-3 mode (CKM = 0) for the 287 speeds tabulated.
2. Obsolete components were replaced with readily available components in Figure 4A.
3. In the AC TIMING REQUIREMENTS table, the timing symbols, T_{AVRL} and T_{AVWL} were reversed in order to match the parameter description.

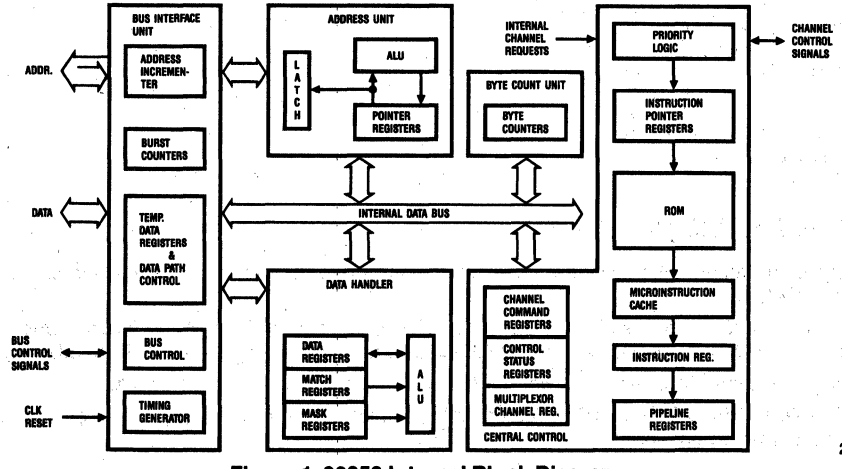
82258
ADVANCED DIRECT MEMORY ACCESS COPROCESSOR (ADMA)

- **High Performance 16 Bit DMA Coprocessor for the 80386, 80286 and 80186 Families**
  - **— 8 MByte/sec Maximum Transfer Rate in 8 MHz 80286 Systems**
- **Four Independently Programmable Channels**
- **Multiplexor Channel Capability to Support Up to 32 Subchannels**
- **On Chip Bus Interface for the Whole 8086 Architecture**
  - **— 80286**
  - **— 80186/188**
  - **— 8086/88**
- **Command Chaining for CPU Independent Processing**
- **Automatic Data Chaining for Gathering and Scattering of Data Blocks**
- **16 MByte Addressing Range**
- **16 MByte Block Transfer Capability**
- **"On the Fly" Compare, Translate and Verify Operations**
- **Automatic Assembly/Disassembly of Data**
- **Programmable Bus Loading**
- **6 and 8 MHz Speed Selections**
- **Available in 68-Pin LCC and PGA Packages**

(See Packaging Spec. Order #231369)

INTRODUCTION

Intel's 82258, Advanced Direct Memory Access Coprocessor is a high performance, 16 bit DMA processor optimized for the 80286, 80186 and the 8086 families of CPUs and compatible with 80386 CPU. It has on-chip bus interface for the whole 8086 family architecture. Four high speed, independently programmable DMA channels can achieve a maximum cumulative transfer rate of 8 MByte/sec in an 8 MHz 80286 system and 4 MByte/sec in 8 MHz 8086/80186 systems. Channel 3 can be used as a Multiplexor channel, whereby, it supports 32 subchannels. This flexibility allows one to use a single DMA channel to handle a large number of slow and medium speed I/O devices. Advanced capabilities like Command and Data chaining and "On the fly" operations allow the 82258 to remove the I/O management load from the processor. The 82258 addresses the full 80286 CPU memory (16 MB for 80286), thus simplifying the system design. Automatic assembly/disassembly of data allows 16 bit processors to interface with common 8 bit peripherals and vice-versa. Remote mode of operation, where the 82258 has its own resident bus, allows modular system design. The 82258 complements the high performance, multitasking capabilities of the 80286.



231263-1

Figure 1. 82258 Internal Block Diagram

November 1987
Order Number: 231263-004

FABRICATION

The 82258 is a 68 pin device, fabricated in Intel HMOS II technology. It is packaged in JEDEC type A hermetic leadless chip carrier and pin grid array.

PIN DEFINITIONS AND FUNCTIONS

The 82258 has four operational modes

— 286
— 186—for the 80186/88 and the 8086/88 (Min. mode) CPUs
— 8086—for the 8086/88 (Max. mode) CPUs
— Remote

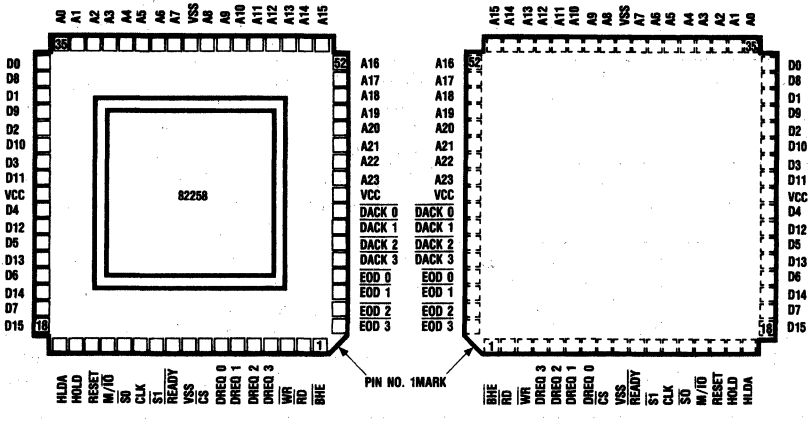
Pins of the 82258 have different definitions for different modes. 286 and remote modes have the same non-multiplexed bus structure and similar pin descriptions. Similarly, the 186 and the 8086 modes have multiplexed bus and similar pin description.

PINNING IN THE 286 MODE

In the 286 mode, the bus signals and the bus timings of the 82258 are the same as those of the 80286 processor. The processor can access the internal registers of the 82258 and these accesses must be supported by the bus signals. Therfore, some of the bus control signals are bidirectional and some additional bus control signals are necessary.

Component Pad View - As viewed from underside of component when mounted on the board

P.C. Board View - As viewed from the component side of the P.C. board



PGA

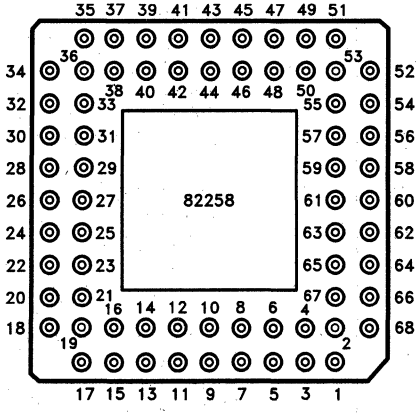


Figure 2. Pin Configuration in 286 Mode

Table 1. Pin Description for the 286 Mode (Also Contains Pins Identical in Other Modes)

Symbol	Pin: Type Input (I) Output (O)	Pin: Number	Identical In All Modes	Functions
$\overline{BHE}$	I/O	1	YES	**BUS HIGH ENABLE** indicates transfer of data on the upper byte of the data bus, D15–D8. Eight bit devices assigned to the upper byte of the data bus would normally use $\overline{BHE}$ to condition chip select function. $\overline{BHE}$ is active LOW and floats to Tri-State OFF when the 82258 does not own the bus.

$\overline{BHE}$ and A0 Encoding

$\overline{BHE}$ Value	A0 Value	Function
0	0	Word Transfer (D15–D0)
0	1	Byte Transfer on upper half of data bus (D15–D8)
1	0	Byte Transfer on lower half of data bus (D7–D0)
1	1	Odd addressed byte on 8 bit bus (D7–D0)

$\overline{RD}$	I	2	NO	**READ** command in conjunction with chip select ($\overline{CS}$) enables reading out of the 82258 register, addressed by the address lines A7-A0. $\overline{RD}$ is an active LOW signal and is asynchronous to the 82258 clock.
$\overline{WR}$	I	3	NO	**WRITE** command along with $\overline{CS}$ is used for writing into the 82258 registers. $\overline{WR}$ is an active LOW signal and is asynchronous to the 82258 clock.
DREQ3-DREQ0	I	4—7	YES	**DMA REQUEST** input signals are used for externally synchronized DMA transfers. If channel 3 is used as a Multiplexor channel, DREQ3 is defined as I/O Request (IOREQ) signal. These signals are active HIGH signals and are asynchronous to the 82258 clock. Unused DREQn lines should not be left floating, but should be tied inactive to V_{SS}.
$\overline{CS}$	I	8	NO	**CHIP SELECT** is used to enable a processor to access the 82258 registers. This access is additionally controlled either by bus status signals or by the Read or Write command signals. $\overline{CS}$ is an active LOW signal, asynchronous to the 82258 clock.
$\overline{READY}$	I	10	NO	**BUS READY** terminates a bus cycle. Bus cycles are extended without limit until terminated by an active $\overline{READY}$. $\overline{READY}$ is an active LOW, synchronous input, requiring set up and hold times relative to system clock to be met for correct operation.
$\overline{S1}$, $\overline{S0}$	I/O	11,13	YES	**BUS CYCLE STATUS** signals control the support circuitry. The beginning of a bus cycle is indicated by $\overline{S1}$, or $\overline{S0}$, or both going active. The termination of a bus cycle is indicated by all the status signals going inactive in the 186 mode or the bus ready ($\overline{READY}$) going active in the 286 mode. Both $\overline{S0}$ & $\overline{S1}$ are active LOW signals. $\overline{S0}$, $\overline{S1}$ along with $\overline{S2}$ (in the 186 mode) or M/$\overline{IO}$ (in the 286 mode) define the type of bus cycle. $\overline{S2}$ and M/$\overline{IO}$ have the same meaning but, in the 186 mode $\overline{S2}$ signal can be active only when at least one of $\overline{S1}$ and $\overline{S0}$ is active, whereas in the 286 mode the M/$\overline{IO}$ signal is valid with the address on address lines.

Table 1. Pin Description for the 286 Mode (Also Contains Pins Identical in Other Modes) (Continued)

<table>
<tr><th rowspan="2">Symbol</th><th colspan="2">Pin</th><th rowspan="2">Identical In All Modes</th><th rowspan="2">Functions</th></tr>
<tr><th>Type Input (I) Output (O)</th><th>Number</th></tr>
<tr><td></td><td></td><td></td><td></td><td>
The 82258 Bus Cycle Status Definitions (82258 Local Bus Master, All Signals (O))
<table>
<tr><th>M/$\overline{IO}$ or $\overline{S2}$</th><th>$\overline{S1}$</th><th>$\overline{S0}$</th><th>Bus Cycle Initiated</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Read I/O-Vector (For Multiplexor channel)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Read from I/O space</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Write into I/O space</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>None. (Does not occur in the 186 mode).</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>None. (Does not occur)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Read from memory space</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Write into memory space</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>None; not a bus cycle</td></tr>
</table>
When the 82258 is not a bus master of the local bus, the status signals are used as inputs for detection of synchronous accesses to the 82258.
Interpretation of the Status and $\overline{CS}$ Signals by the 82258 (82258 Slave, All Signals (I))
<table>
<tr><th>$\overline{CS}$</th><th>$\overline{S1}$</th><th>$\overline{S0}$</th><th>Interpretation</th></tr>
<tr><td>1</td><td>X</td><td>X</td><td>82258 not selected (No action)</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>No 82258 access (No action)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Read from an 82258 register</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Write into an 82258 register</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Not a bus cycle*</td></tr>
</table>
*: The 82258 is selected but no synchronous access is activated. The 82258 monitors $\overline{RD}$ and $\overline{WR}$ signals for detection of an asynchronous access.
</td></tr>
<tr><td>CLK</td><td>I</td><td>12</td><td>NO</td><td>SYSTEM CLOCK provides the fundamental system timing. It is divided by two to generate the 82258 internal clock. CLK is an active HIGH signal which can be connected directly to the 82284 CLK output. The internal divide-by-two circuitry is synchronized to the external clock generator by a LOW to HIGH transition on the RESET input, or by first HIGH to LOW transition on the Status Input $\overline{S0}$ or $\overline{S1}$ after RESET.</td></tr>
<tr><td>M/$\overline{IO}$</td><td>O</td><td>14</td><td>NO</td><td>MEMORY/$\overline{IO}$ SELECT distinguishes between memory and I/O space addresses.</td></tr>
<tr><td>RESET</td><td>I</td><td>15</td><td>YES</td><td>SYSTEM RESET forces the 82258 to the initial state. RESET is an active HIGH signal and must be synchronous to the system clock. Reset must be activated for at least 16 CLK cycles.</td></tr>
</table>

Table 1. Pin Description for the 286 Mode (Also Contains Pins Identical in Other Modes) (Continued)

Symbol	Pin		Identical In All Modes	Functions
	Type Input (I) Output (O)	Number		
HOLD HLDA	O I	16 17	NO	**BUS HOLD REQUEST AND HOLD ACKNOWLEDGE** control ownership of the local 82258 bus. When active, HOLD indicates a request for the control of the local bus. HOLD goes inactive when the 82258 relinquishes the bus. HLDA, when active, indicates that the 82258 can acquire the control of the bus. When HLDA goes inactive, the 82258 must relinquish the bus at the end of its current cycle. HLDA may be asynchronous to the system clock. Both HOLD and HLDA are active HIGH signals.
D15-D0	I/O	18-25, 27-34	NO	**DATA BUS** is the bidirectional 16 bit bus. For use with an 8 bit bus, only the lower 8 data lines D0–D7 are relevant. The data bus is active HIGH.
A0–A7	I/O	35-42	NO	**ADDRESS LINES** A0–A7 are the lower 8 address lines for DMA transfers. They are also used to input the register address when the processor accesses an 82258 register. All lines are active HIGH.
A8–A23	O	44-59	NO	**ADDRESS LINES** A8–A23 form the remainder of the 82258 address bus. Address bus is active HIGH. ***Pin A21 must have a pull-up resistor (n 10k Ω) connected to it to ensure that it is high during reset.***
$\overline{DACK0}$–$\overline{DACK3}$	O	61-64	YES	**DMA ACKNOWLEDGE** signal acknowledges the requests of the corresponding DREQ signal. $\overline{DACK}$i goes active when the requested transfers are performed on the channel i in response to a DREQi. If channel 3 is in the multiplexor mode, $\overline{DACK3}$ is defined as I/O acknowledge (IOACK). These signals are active LOW.
$\overline{EOD0}$–$\overline{EOD3}$	I/O	65-68	YES	**END OF DMA** signals are open drain drivers with internal high impedance pull up resistors (an external pull-up resistor is required) and can be used as quasi-bi-directional lines. These signals are active LOW. As OUTPUTs the signals are activated (if enabled) for two T-STATE cycles at the end of the DMA transfer of the corresponding channel or they are activated under program control (End of DMA output or interrupt output). $\overline{EOD}$s acts as "End of DMA" level triggered INPUTs if the signals are held high internally but forced low by the external circuitry for at least 250 ns. The current transfer is aborted and the 82258 continues with the next command. $\overline{EOD2}$ can also be used as a common active high interrupt signal (INTOUT) for all four channels. In this mode, this signal is a push-pull output and not an open drain output. Other EODi pins may still be used in their regular I/O mode.
V_{SS}	I	9, 43	YES	**SYSTEM GROUND:** 0 Volt.
V_{CC}	I	26, 60	YES	**SYSTEM POWER:** +5V Power Supply Pin.

PINNING IN THE 186 MODE

The 80186 has a multiplexed bus structure. Therefore, many 82258 pins have different meaning in the 186 mode than in the 286 mode. Since the 80186 has 20 address lines compared to 24 for the 80286, the 4 extra lines are used to generate additional bus control signals. The following table gives the details of pins having different meaning in the 186 mode compared to the 286 mode:

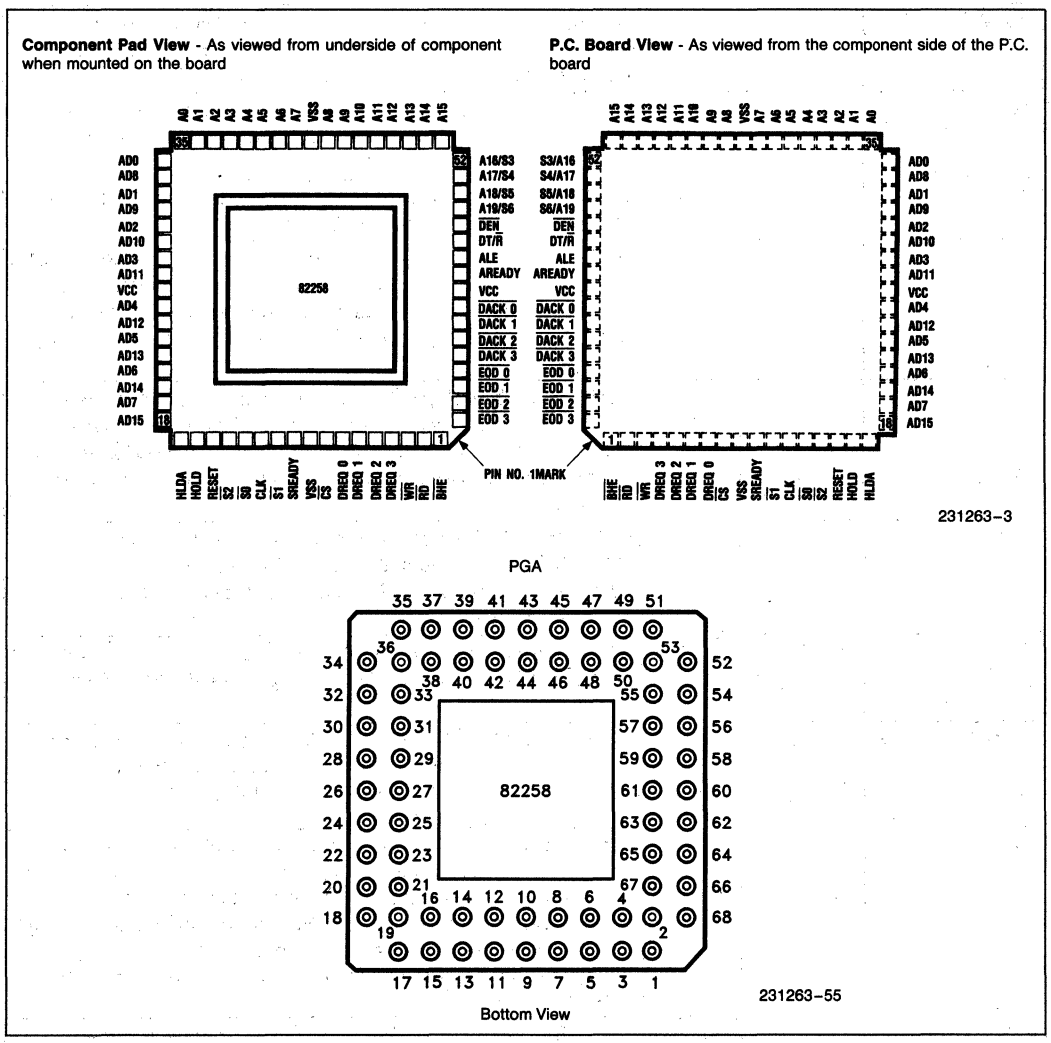


Figure 3. Pin Configuration in the 186 Mode

Table 2. Changes in Pin Description in the 186 Mode: (Compared to the 286 Mode)

<table>
<tr><th rowspan="2">Symbol</th><th colspan="2">Pin</th><th rowspan="2">Functions</th></tr>
<tr><th>Type Input (I) Output (O)</th><th>Number</th></tr>
<tr><td>$\overline{RD}$, $\overline{WR}$</td><td>I/O</td><td>2, 3</td><td>READ, WRITE In the 186 mode, the $\overline{RD}$ & $\overline{WR}$ pins are used additionally as output pins to support the 80186 or the 8086 minimum systems. These signals are active LOW.</td></tr>
<tr><td>ALE</td><td>O</td><td>58</td><td>ADDRESS LATCH ENABLE signal provides a strobe to separate the address information on the multiplexed address-data lines. ALE is an active HIGH signal.</td></tr>
<tr><td>$\overline{DEN}$</td><td>O</td><td>56</td><td>DATA ENABLE signal is used for enabling the data transceiver, 8286/8287. $\overline{DEN}$ is an active LOW signal.</td></tr>
<tr><td>DT/$\overline{R}$</td><td>O</td><td>57</td><td>DATA TRANSMIT/RECEIVE signal controls the direction of data flow through the external data bus transceiver, depending on whether a read, or a write bus cycle is performed. This pin must have a pullup resistor connected to it to ensure that it is high during reset.</td></tr>
<tr><td>SREADY</td><td>I</td><td>10</td><td>SYNCHRONOUS READY input signal must be synchronized externally. Use of this pin permits a relaxed system and timing specification by eliminating the clock phase, required for resolving the signal level, when using AREADY input. SREADY is an active HIGH signal.</td></tr>
<tr><td>CLK</td><td>I</td><td>12</td><td>SYSTEM CLOCK input gets a prescaled signal from the 186 clock (CLKOUT) or the 8086 clock (50% duty cycle for 186 and 33% duty cycle for 8086). No internal prescaling is done. CLK is an active HIGH signal.</td></tr>
<tr><td>$\overline{S2}$</td><td>O</td><td>14</td><td>STATUS SIGNAL along with $\overline{S0}$ and $\overline{S1}$ provides the bus cycle description (for details see 286 mode pin description of $\overline{S0}$ and $\overline{S1}$).</td></tr>
<tr><td>AD0–AD15

A0–A7
A8–A15</td><td>I/O

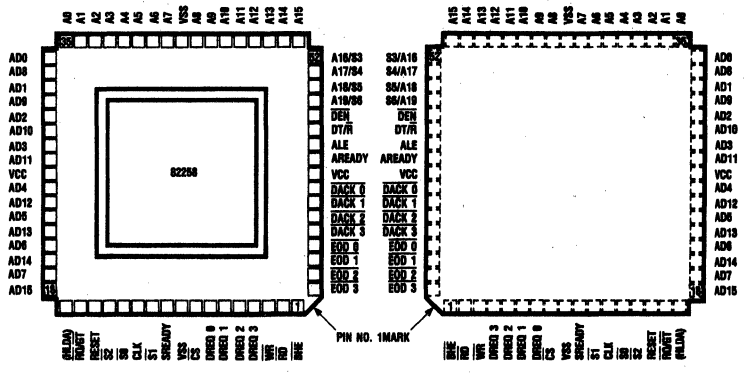
I/O
O</td><td>18–25
27–34
35–42
44–51</td><td>ADDRESS/DATA BUS signals AD0–AD15 contain multiplexed lower address and data information. Also, the demultiplexed address information is available on address pins A0–A15.</td></tr>
<tr><td>A16/S3–A19/S6</td><td>O</td><td>52–55</td><td>ADDRESS PINS A16–A19 are multiplexed with additional status information on the bus cycle. These pins are active HIGH.
Signals S5 and S6 provide information on the status of the bus cycle. During an active bus cycle, S6 is always high and S5 always low. Low S6 implies a processor bus cycle.
Signals S4 and S3 give the channel number for the running bus cycle as follows:
<table>
<tr><th>S4</th><th>S3</th><th>Channel Number</th></tr>
<tr><td>0
0
1
1</td><td>0
1
0
1</td><td>0
1
2
3</td></tr>
</table></td></tr>
<tr><td>AREADY</td><td>I</td><td>59</td><td>ASYNCHRONOUS READY is an asynchronous bus ready signal. The rising edge is internally synchronised. During reset, AREADY must be low to enter the 82258 into the 186 mode. AREADY is an active HIGH signal.</td></tr>
</table>

PINNING FOR THE 8086 MODE

For the 8086 MIN configuration the pinning is identical to the 186 mode. For the 8086 MAX configuration, the bus arbitration is done via the $\overline{RQ}/\overline{GT}$ protocol. Otherwise, the function of pins is identical to the 186 mode.

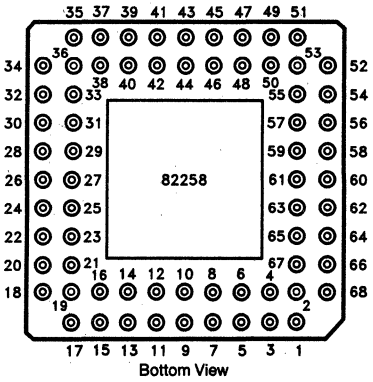
Component Pad View - As viewed from underside of component when mounted on the board

P.C. Board View - As viewed from the component side of the P.C. board



231263-4

PGA



Bottom View

231263-56

Figure 4. Pin Configuration in the 8086 (Max) Mode

Table 3. Changes in Pin Description in the 8086 (Max) Mode
(Compared to the 186 Mode)

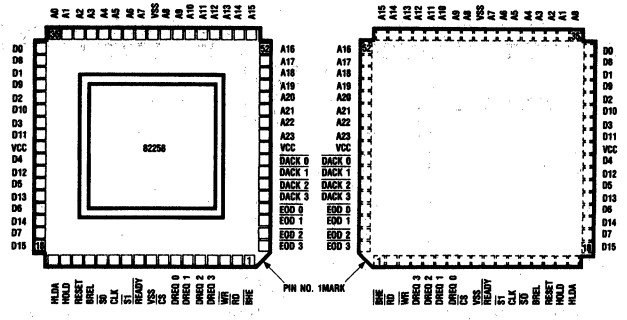
Symbol	Pin		Functions
	Type Input (I) Output (O)	Number	
$\overline{RQ}/\overline{GT}$	I/O	16	**REQUEST/GRANT** implements a one line communication protocol to arbitrate the use of the system bus; normally done via HOLD/HLDA. $\overline{RQ}/\overline{GT}$ is an active LOW signal having an internal pull-up resistor.
HLDA	I	17	**HOLD ACKNOWLEDGE** has no meaning in the 8086 (Max) mode. It should be tied high for mode recognition during reset.

PINNING IN THE REMOTE MODE

In the remote mode, most of the signals have the same function as in the 286 mode. Exceptions are noted in the following table:

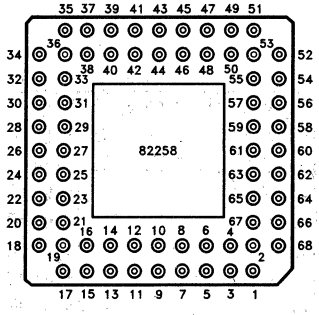
Component Pad View - As viewed from underside of component when mounted on the board

P.C. Board View - As viewed from the component side of the P. C. board



231263-5

PGA



231263-57

Bottom View

Figure 5. Pin Configuration in Remote Mode

Table 4. Changes in Pin Description in the Remote Mode (Compared to the 286 Mode)

Symbol	Pin		Functions
	Type Input (I) Output (O)	Number	
$\overline{CS}$	I	8	**CHIP SELECT** has two functions in the remote mode. As in the 286 mode, $\overline{CS}$ enables access to the 82258 internal registers. In addition $\overline{CS}$ works as an Access Request Input. When forced LOW, it signals to the 82258 that another bus master needs access to the local bus of the 82258. The 82258 releases the bus as soon as possible and signals it to the CPU by activating BREL (Bus Release) output. $\overline{CS}$ is an active LOW signal.
BREL	O	14	**BUS RELEASE** signal is used to indicate when the 82258 releases control of the resident bus.
HOLD HLDA	O I	16 17	**HOLD & HOLD ACKNOWLEDGE** signals are used only for access to the system bus. They are connected to the bus arbiter (i.e., 82289). Resident bus accesses are directly executed without the HOLD/HLDA sequence.

FUNCTIONAL DESCRIPTION

The 82258 is an advanced DMA coprocessor for the 8086 family architecture. In addition to providing high speed DMA transfers (8 MByte/sec in an 8 MHz 80286 and 4 MByte/sec in 8 MHz 80186/86 systems), the 82258 takes I/O processing load off the CPU, thus improving overall system performance. The 82258 has advanced features not found in the previous generation DMA controllers: multiple-xor channel, command & data chaining and 'on the fly' data manipulation operations.

MODES OF OPERATION

The 82258 has a number of different modes of operation based upon its coupling with the CPU (tight or loose) and its adaptive on-chip bus interface (the 286 bus or the 186 bus).

Figure 6 shows the different operating modes of the 82258 and the CPUs it can interface with in those modes. Figure 7 shows how to configure the 82258 into these different modes.

LOCAL MODE

In this mode the 82258 shares the local bus and all the support/control devices with the CPU. Because of its on-chip bus interface, the 82258 can be directly coupled to the whole 8086 family of microprocessors.

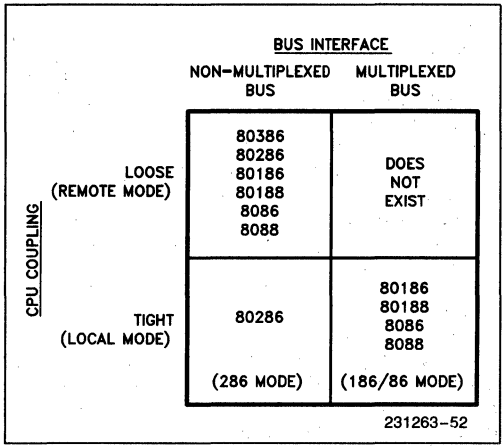


Figure 6. Operating Modes for the 82258

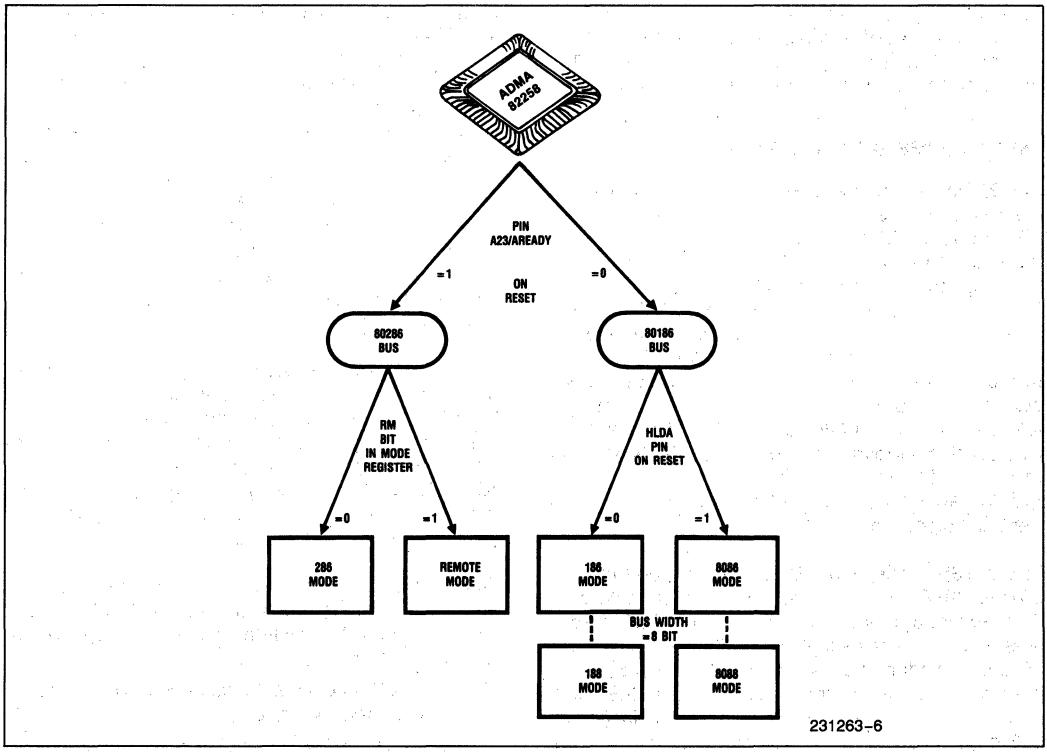


Figure 7. Selecting Modes of Operation

286 System

The configuration in Figure 8 shows the 82258 in the local mode (286 mode) in an 80286 system which includes the Numeric Processor Extension, 80287. The 286 mode is selected during reset (Figure 7). In this mode the 82258 supports the non-multiplexed, pipelined 286 bus. The DMA coprocessor resides on the processor's local bus (physical pins of the 80286) and shares all the support circuits: latches, tranceivers, bus controller and arbiter, clock generator etc. By residing on the 286 bus, the 82258 achieves maximum data transfer rate; up to 8 MByte/sec at 8 MHz for single cycle transfer. HOLD/ HLDA protocol is used for bus exchange between the 80286 and the 82258. The 82258 can be programmed to handle both internal and external terminate conditions. Internal termination is programmed in the command block (in type 2 command as explained later). External termination is handled by the $\overline{EOD}$ (end of DMA) pins if they are enabled. Interrupts for the CPU are handled by an interrupt controller (e.g. 8259A) which receives the end of DMA pins ($\overline{EOD}$ 0-3) as interrupts. The multiplexor channel uses external 8259As to prioritize and arbitrate service requests between peripherals (Figure 13).

To link this system to the MULTIBUS® bus architecture another set of latches, tranceivers, bus controllers and a bus arbiter (i.e., 82289) as shown in Figure 11 (for remote mode configuration) are needed.

186/188 (8086/8088 Min) Systems

The 82258 can be configured into the 186 mode during reset (Figure 7). In this mode it supports the 80186 and the 8086 (Min) processors. It can be programmed to support the 80188 and the 8088 (Min) by programming the bus width in General Mode Register (GMR). Figure 9 shows the 82258 used in an 80186 system containing the 8087 numeric coprocessor. This system uses the 8086 bipolar support components: latches, transceivers and the bus controller (8288). The Integrated Bus Controller (82188) links the 80186 to the 8087. The 82188 is also used to support the 82258, since the 80186 has only one set of bus exchange signals (HOLD/ HLDA). An interrupt controller (8259A) processes the $\overline{EOD}$ signals for the CPU.

In the 186 mode, the 82258 directly supports the 80186/ 8086 bus with 16 address bits internally multiplexed into the data lines (AD15-AD0). The address pins A19-A16 are multiplexed with the status lines S6-S3. The address pins A22-A20 (in the 286 mode) are used to generate the control signals ALE, $\overline{DEN}$ and DT/$\overline{R}$ (in the 186 mode). The A23 pin (in the 286 mode) serves as an asynchronous ready input AREADY (in the 186 mode). As a master in the 186 mode, the 82258 offers address lines A15-A0 as latched outputs and shares all the 186/8086 support components with the processor.

8086/88 Systems

The 82258 is configured into the 8086 mode during reset (Figure 7). In this mode the 82258 supports 8086/88 in the maximum mode and uses the $\overline{RQ}/\overline{GT}$ protocol for the processor - DMA coprocessor bus exchange. The 8087 can be supported in the system without requiring the integrated bus controller, 82188. To support the 8088 system in the maximum mode, the General Mode Register is programmed for 8 bit bus width. Figure 10 shows the 82258 in an 8086 system containing the 8087. The system configuration is very similar to the 80186 system in Figure 9.

REMOTE MODE

The 82258 is configured to be in the Remote Mode (Figure 7) by programming the General Mode Register (RM bit), after putting the 82258 in the 286 mode during the reset. The 82258 has the bus timings and signals compatible to the 286 bus.

In the remote mode, the ADMA can access two 16 MByte address spaces normally called the resident space and the system space. The ADMA does not distinguish between accessing an I/O device and accessing a memory in the remote mode, so either peripheral or memory can belong to either of the two spaces.

In the remote mode, the 82258 is the sole local bus (resident bus) master and interfaces to the processor through the system bus (using a bus arbiter). Therefore, the 82258 can work in parallel with the processor. The remote mode is useful for a modular I/O subsystem.

Figure 11 shows the 82258 configured in the remote mode of operation. The peripherals interface to the 82258 on the resident bus. The resident bus components are similar to the ones used for the 286 system. Additional support components are used to interface the 82258 to a system bus e.g. the MULTIBUS. The 82258 communicates with the CPU (80286) over the system bus.

Since the 82258 is the only master of the local/resident bus, it can start the local bus cycles without any bus arbitration. For system bus accesses, a deadlock can arise if:

— The 82258 occupies the local bus to gain access to the system bus and

— The CPU (80286) occupies the system bus to gain access to the 82258 (through its local bus)

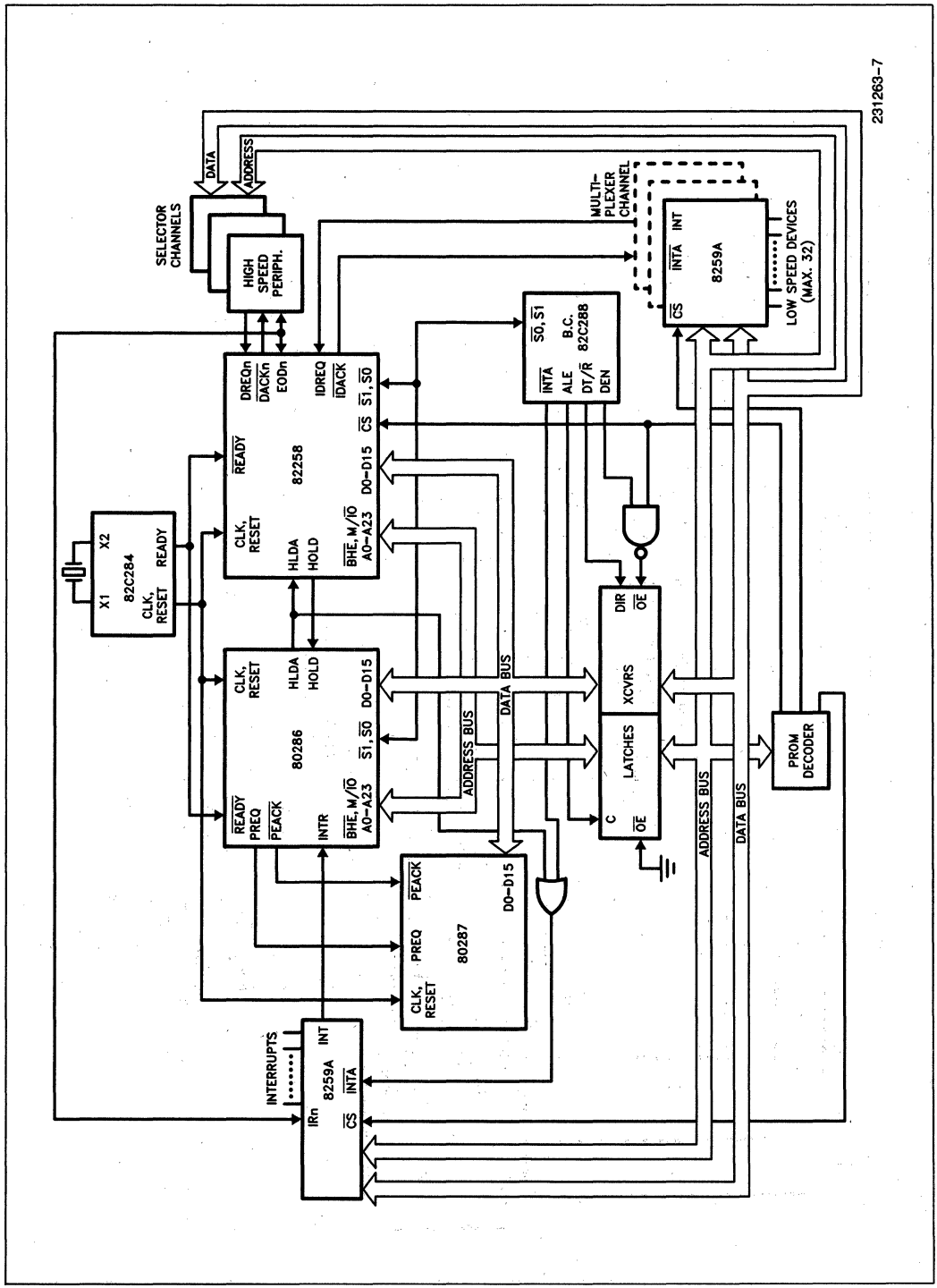


Figure 8. 80286 in an 80286 System

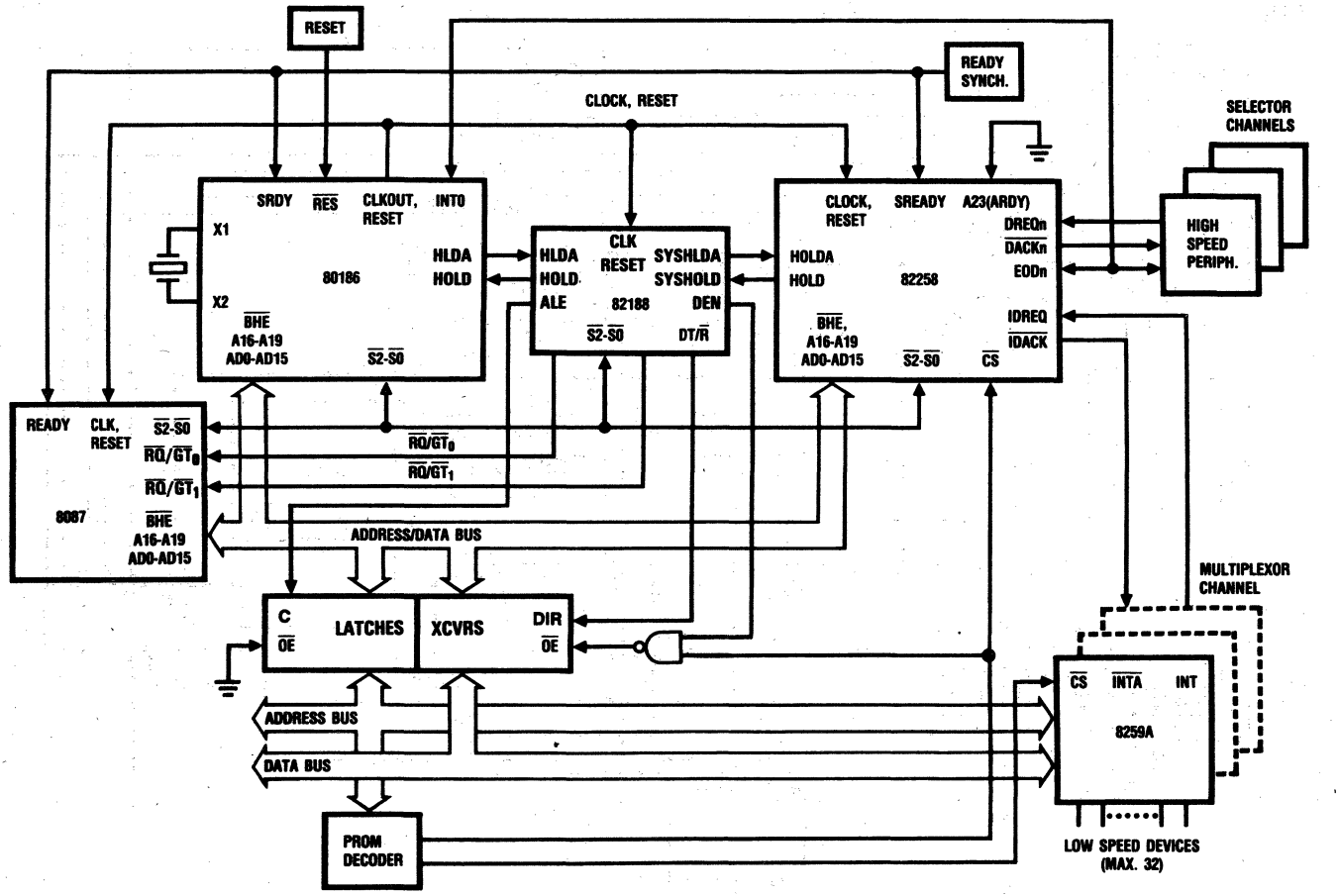


Figure 9. 82258 in an 80186 System

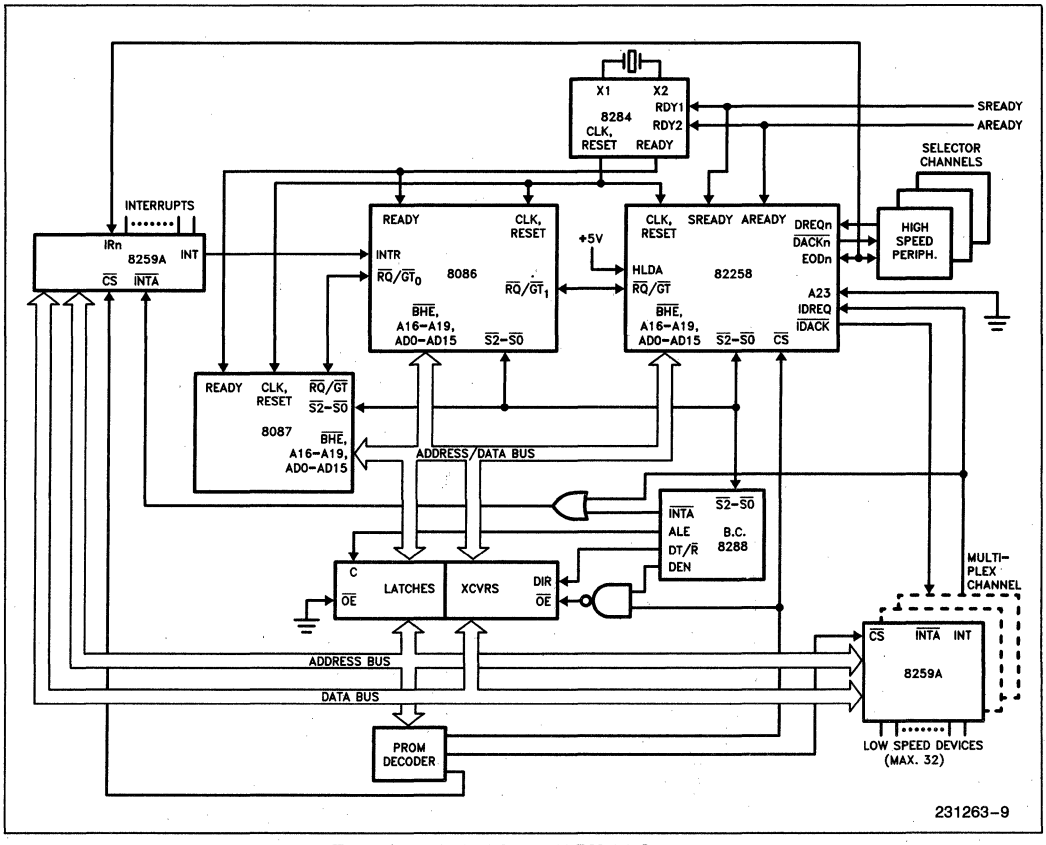


Figure 10. 82258 in an iAPX 86 System

To prevent this deadlock, for the system bus accesses the 82258 does not occupy the local bus until it has the system bus. Therefore, in the remote mode, the 82258 initiates all system bus accesses (and only these) through the HOLD/HLDA protocol. The local bus arbitration (for the CPU) is done through the $\overline{CS}$ and the BREL lines.

COMMUNICATION MECHANISMS

CPU → 82258 COMMUNICATION

Communication from the CPU to the 82258 is twofold:

— Some 82258 registers receive the main commands from the CPU, through the slave interface of the 82258. Access to the 82258 is either synchronous (using $\overline{CS}$, $\overline{S1}$, $\overline{S0}$) or asynchronous (using $\overline{CS}$, $\overline{RD}$, $\overline{WR}$; $\overline{S1}$ = $\overline{S2}$ = 1).

— Most of the data is transferred via the control space in the memory in terms of organization blocks e.g. command blocks and multiplexor table. Control space can lie in the memory space or the memory mapped I/O space (system or resident space for the remote mode) and can be dynamically changed with every start channel command.

The CPU communicates with the 82258 by depositing data in the memory and into the on-chip registers of the 82258. The CPU can access the 82258 general registers and status registers, and can start a channel by writing the proper command to the general command register (GCR). The 82258 will then read the data from the memory command block and set itself up.

Slave Interface

The slave interface of the 82258 is used by the CPU to access the 82258 internal registers. Although most of the CPU to 82258 communication is done through memory based data blocks, some direct accesses to the 82258 registers are necessary. For example, during the initialization phase the general

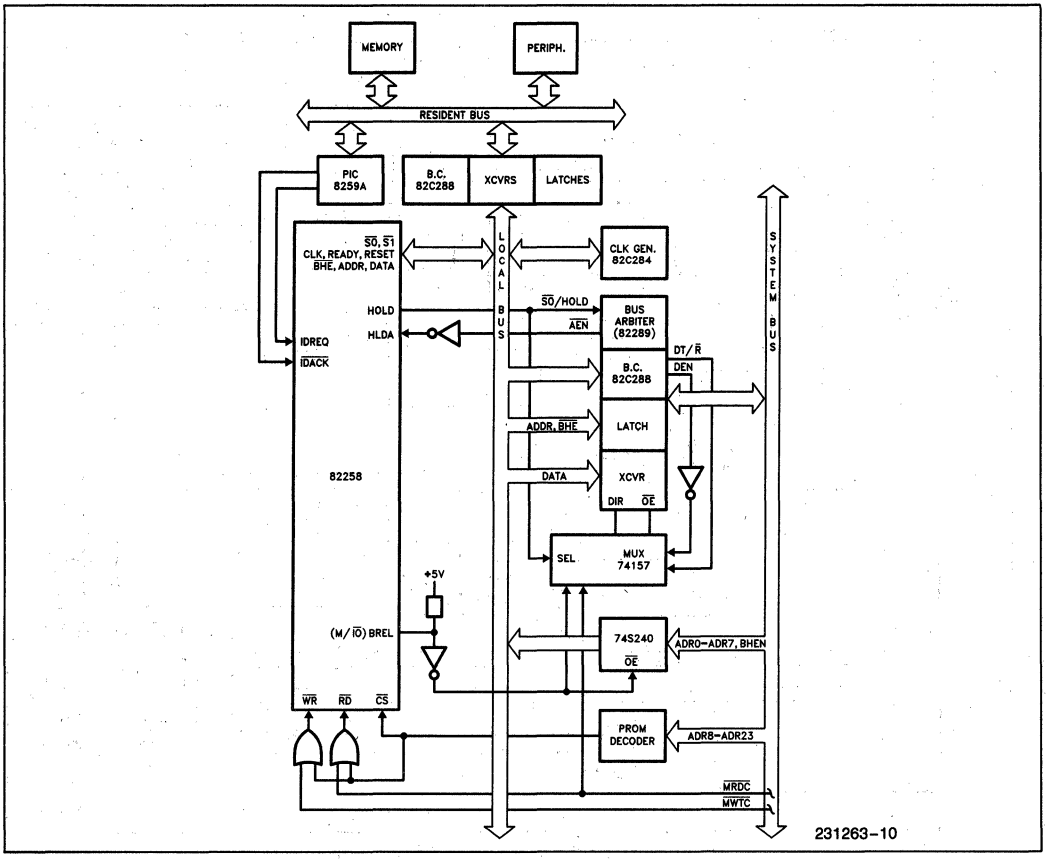


Figure 11. 82258 In Remote Mode

mode register (GMR) must be written to set up the 82258 or, to start a channel the command pointer registor (CPR) and the general command register (GCR) must be loaded. During the system debugging phase, access to the 82258 internal registers is very important.

The slave interface is enabled by the $\overline{CS}$ input and consists of the following lines:

- $\overline{S1}$, $\overline{SO}$ —Status Lines (inputs)
- $\overline{RD}$, $\overline{WR}$ —Control Lines (inputs)
- A7–A0 —Register Address (inputs)
- D15–D0 —Data Lines (inputs/outputs)-(for the 286 and the remote modes)
- AD15–AD0 —Address/Data Lines (inputs/outputs)-(for the 186 and 8086 modes)

In the 286 mode and the 186/86 mode, two types of accesses are possible:

— synchronous access through the status lines $\overline{SI}$ and $\overline{S0}$

— Asynchronous access using $\overline{RD}$ and $\overline{WR}$

The register address must be supplied on the address pins A7–A0, except for the synchronous access in the 186/86 mode. Address data lines AD7–AD0 are used for the register address information in case of a synchronous access in the 186/86 mode.

In the remote mode, a synchronous access is not possible as the 82258 has to release its local bus to enable the CPU to access its registers. On receiving an access request ($\overline{CS}$ input asserted), the 82258 releases the local bus as soon as possible and signals it by asserting the BREL line. Only then, can the CPU access the 82258 registers.

82258 → CPU COMMUNICATION

The 82258 to the CPU communication is also twofold:

— Hardware based communication, using one or more $\overline{EOD}$ lines as interrupt request lines to the CPU. The CPU can then read the status registers

(and the interrupt vector register for the multiplexor channel) and service the interrupt.

— Control space based communication: At the end of a DMA transfer, the 82258 writes the contents of the appropriate channel status register into the channel command block. Additionally, it may transfer some other information (e.g. the updated source pointer) into the command status block.

The 82258 updates its internal registers (e.g. the channel command pointer, the general status register etc.) for any CPU access.

82258 — PERIPHERAL COMMUNICATION

The DMA interface of the 82258 is used for its communication with the peripherals. It consists of three signal lines:

DREQ —DMA Request
$\overline{DACK}$ —DMA Acknowledge
$\overline{EOD}$ —End of DMA

DREQ and $\overline{DACK}$ control the externally synchronized DMA transfers. A burst of data is transferred for a continuous DMA request, as long as the request signal is active.

$\overline{EOD}$ lines, which are quasi-bidirectional, enhance the 82258—Peripheral communication link. First these can be used as inputs to the 82258 to receive an asynchronous external terminate signal to terminate a running DMA. As outputs, they can be used to interrupt the CPU and/or to signal a specific status to the peripheral (e.g. transfer aborted or, end of a block or, send/receive next block..). In addition, the $\overline{EOD}$ output of channel 2 can be used as a collective interrupt output (INTOUT) for all the DMA channels while the other three $\overline{EOD}$ lines retain their normal function.

An $\overline{EOD}$ output signal can be generated synchronous to a synchronising device at the last data transfer or, synchronous to the internal clock at the last destination cycle. An $\overline{EOD}$ can also be generated asynchronously through a Type 2 command.

BUS ARBITRATION

HOLD/HLDA Sequence

These signals are used for the bus arbitration in the 286 mode and the 186/88 (8086/88 Min.) mode. Whenever the 82258 needs the bus, it activates the HOLD signal and the processor surrenders the local bus as soon as possible by asserting HLDA. The 82258 performs the transfer and switches the HOLD to low. The processor takes the bus and switches the HLDA to low. To force the 82258 to surrender the bus, the HLDA must be set to low. The 82258 will release the bus after the currently running bus cycle or the unseparable bus cycles. Unseparable bus cycles are:

— The two IO acknowledge bus cycles for the 8259A PIC.

— Word transfers on odd boundary addresses, realised by two bus cycles where each transfer is a byte.

— Fetch of 24 bit address pointers out of the memory or restore of the pointers.

— Read- modify- write the 8259A mask registers.

The 82258 signals the surrendering of the bus by floating the bus and removing the HOLD signal. If requests for bus cycles are present, the HOLD will go active after a delay of two T-states.

$\overline{RQ}/\overline{GT}$ Sequence

$\overline{RQ}/\overline{GT}$ protocol is used for the 8086/88 (Max.) Mode. The 82258 requests the bus by sending a request pulse of one CLK period length, via the $\overline{RQ}/\overline{GT}$ signal, to the processor. The processor acknowledges it with a pulse on the same line. Then the 82258 controls the bus. When surrendering the bus, it sends a release pulse on the $\overline{RQ}/\overline{GT}$ line.

$\overline{CS}$/BREL Sequence

This is used in the remote mode along with the HOLD/HLDA signals. HOLD/HLDA are used for system bus arbitration and $\overline{CS}$/BREL for local bus arbitration (to allow the CPU to access the 82258 registers or the resident bus). The CPU asserts the $\overline{CS}$ signal to ask for the local bus and the 82258 releases the bus as soon as possible by activating BREL. After the CPU has completed its access, it should set $\overline{CS}$ high. The 82258 deactivates BREL and proceeds with its own bus cycles on the local bus.

NOTE:
When the 82258 is not in possession of the bus, all output signals are tristated except the following:
HOLD (except in the $\overline{RQ}/\overline{GT}$ protocol), $\overline{DACK}0$–$\overline{DACK}3$, $\overline{EOD}0$–$\overline{EOD}3$,
BREL (remote mode) and ALE (186 mode)

CHANNEL CONFIGURATION

The 82258 has four independently programmable DMA channels with their own register sets. All channels can be used as high speed selector channels for achieving maximum transfer rate or channel 3 can be used as a multiplexor channel to allow the 82258 to interface to a large number of I/O devices.

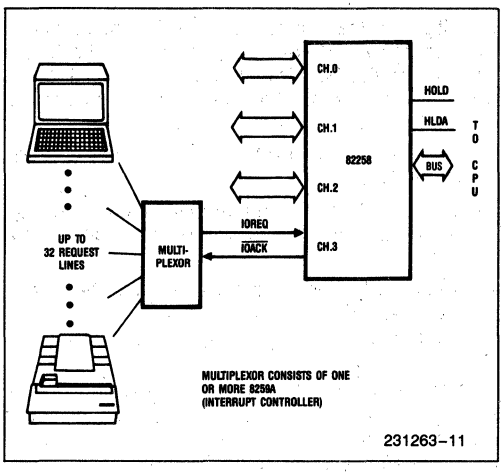


Figure 12. 82258 Channel Configuration

The selector channels support synchronised and non synchronised transfers as well as advanced features like single cycle transfer, command and data chaining. Channel switching imposes no performance penalty on the 82258. Programmable priority schemes allow flexible multiple channel processing.

MULTIPLEXOR CHANNEL

Channel 3 of the 82258 can also be operated as a multiplexor channel supporting up to 32 subchannels. External 8259As are used to arbitrate and prioritize channel requests (Figure 13). Multiplexor channel allows command chaining but data chaining is not supported.

As a multiplexor channel, channel 3 uses an external multiplexor table (MT) in the memory to store separate command pointers and, the PIC (8259A) mask register locations for each device in that channel. Each entry in the MT consists of 8 bytes; the first 4 give the command pointer for the subchannel and the second 4 the address of the mask register of the 8259A for that subchannel (Figure 14).

After an I/O request from the 8259A, the 82258 fetches an 8 bit vector (device number) from the interrupt controller (by the INT/INTA mechanism), left shifts it by three and, uses that as an offset into the multiplexor table with that entry pointing to the current subchannel command block. The 8259A should be programmed for AEOI mode.

Each subchannel can have a subchannel program or a command chain. The command chain must be terminated by a stop and mask command (as opposed to a stop command for a selector channel). Three kinds of data transfers are possible:

Byte/Word Multiplex: One byte/word is transferred per request. The source/destination pointer and the byte count fields of the command block are updated. The command pointer is not advanced until the block transfer is terminated. Maximum cumulative data transfer rate of 275K Bytes/sec can be achieved for the channel.

Single Transfer: Similar to the byte/word multplex. But, the command pointer is advanced after each transfer, thus, executing command chaining.

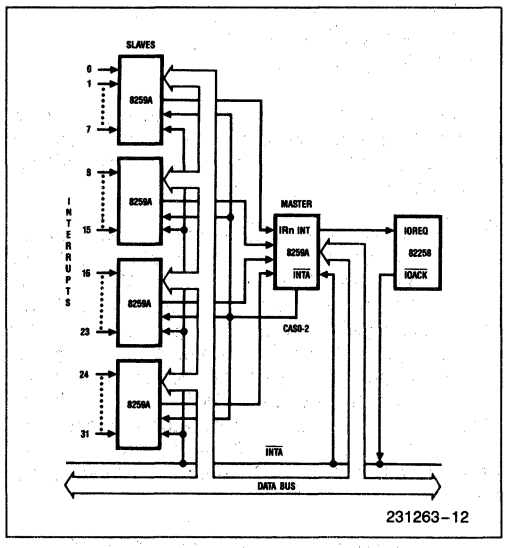


Figure 13. Multiplexor Configuration

Block Multiplex Transfer: The whole command block is executed and a block transfer made upon receiving a request. Such transfer is necessarily free running or non-synchronized and is carried out at a maximum speed of 4 MByte/sec in an 8 MHz 80286 system. After termination, the command pointer is advanced (command chaining).

The type 2 commands have the same function as for the selector channels (Table 6). A subchannel is stopped with a stop and mask command which must occur at the end of a command block chain. The 82258 generates the interrupt (INTOUT) or $\overline{EOD}$, if programmed. The 82258 automatically masks the request line on the 8259A by setting its mask bit. Thus no further requests can come from this subchannel until it is enabled by the CPU. The 82258 indicates the interrupted subchannel (vector) in the Multiplexor Channel Interrupt Vector Register (MIVR). The MIVR can be accessed by the CPU and, after reading the MIVR, the stop bit of the indicated subchannel is reset. If no channel 3 interrupt (EOD or programmed INTOUT) is enabled, the internal interrupt flag is set by the stop and mask command. Then the CPU checks the MIVR by polling, i.e., with each reference of this register, the CPU can read off the stopped subchannel vector that has the highest priority in queue until the NV (vector is not valid) bit in MIVR is set.

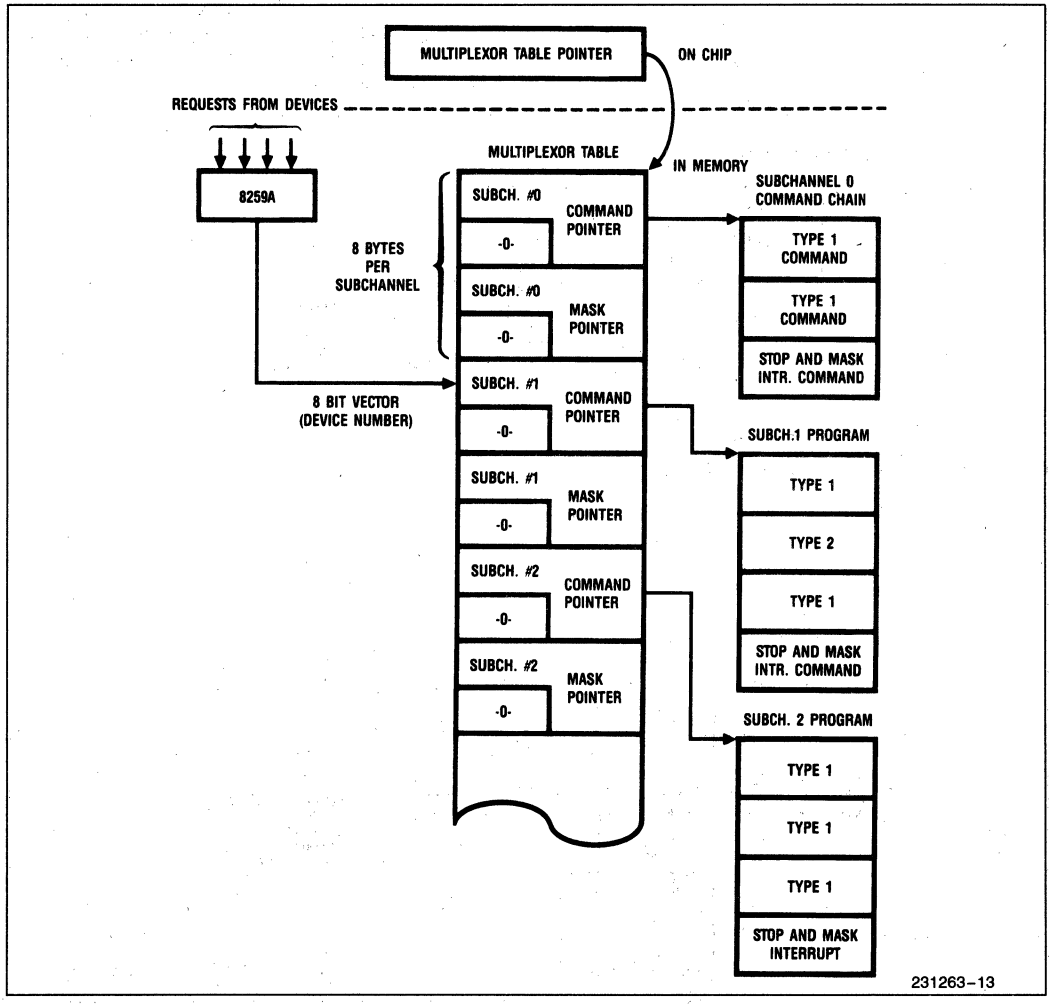


Figure 14. Multiplexor Table

DATA TRANSFER AND MANIPULATION CONTROL

SINGLE CYCLE AND TWO CYCLE TRANSFERS

The 82258 provides the flexibility to optimize the system design by allowing:

— Highest speed DMA transfers in the single cycle transfer mode. In this mode bytes or words (16 bits) are transferred directly from the source to the destination without storing the data in the 82258 registers (Figure 15). The single cycle transfer mode does not, necessarily, mean one bus cycle for transfer (though most of the transfers require either a source or a destination data cycle only). Maximum single channel or multiple channel transfer rate of 8 MByte/sec. in an 8 MHz 80286 system (4 MByte/sec in 8 MHz 80186 systems) is achieved in this mode.

In the single cycle transfer mode, while the requesting device is serviced (and addressed) using $\overline{\text{DACK}}$ signal, the pointer to the other location (memory or I/O) is issued and its bus cycle executed by the 82258. It is the duty of the I/O device to know whether the cycle is a read cycle or a write cycle and, to generate its command signal out of the bus command signals.

Single cycle transfers mode is not allowed for the multiplexor channel. All single cycle transfer are externally synchronised and "On the fly" operations are restricted (see Table 5).

— Maximum data manipulation operations in the two cycle transfer mode. The two cycle transfer mode does not, necessarily, imply two bus cycles, though most of the transfers consist of a fetch cycle from the source and a store cycle to the destination location. In this mode the source data is always stored in the 82258 registers before being sent out to the destination. Although half as

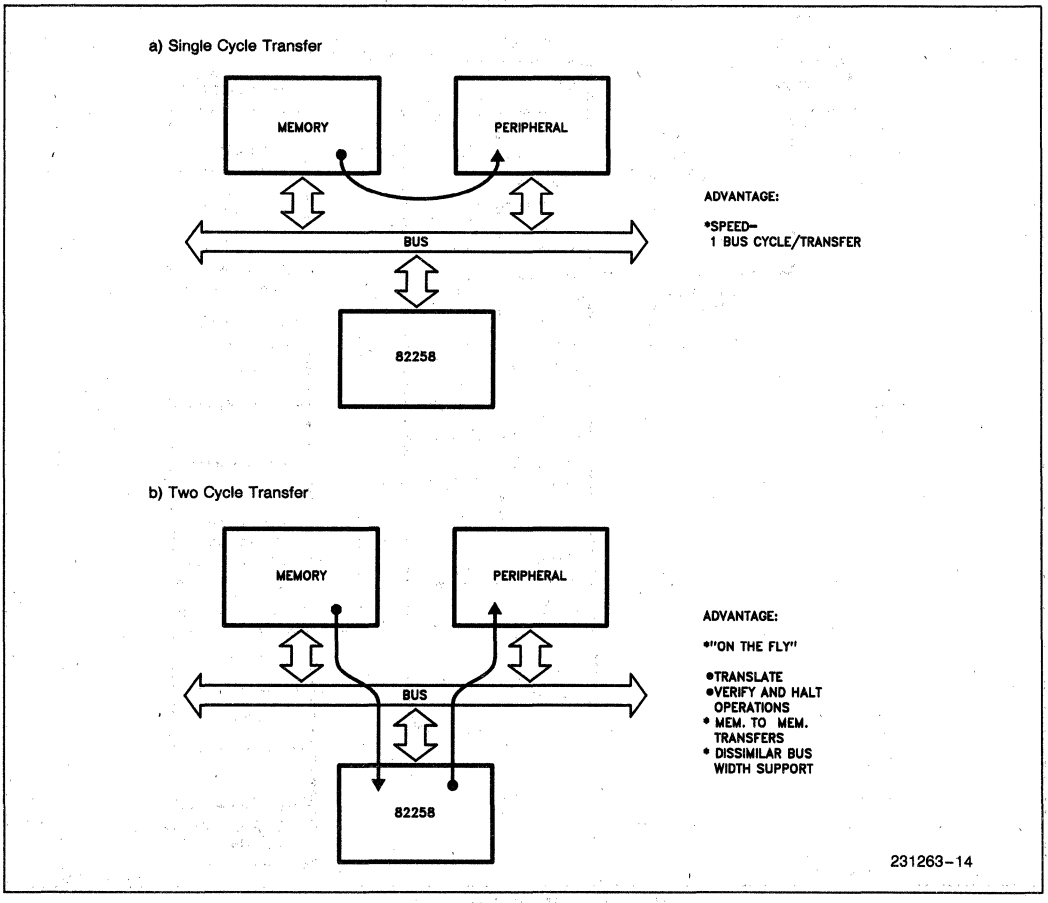


Figure 15. Single/Two Cycle Transfer

fast as the single cycle mode, a number of "On the fly" operations e.g., translation, make this mode extremely versatile. The two cycle transfer mode also allows automatic assembly and disassembly of the data, i.e., the data can be read as one 16 bit word and written as 2 bytes or vice-versa. It is useful for linking the 8 bit peripherals to a 16 bit system and vice-versa.

The two cycle transfer mode allows multiplexor channel operation and memory to memory transfers. Two special cases of two cycle data transfer are:

Read Operation or, data transfer without a destination address (the data assembly register of the 82258 itself is the destination of the source data). Compare operations on the source data are possible (e.g. to test the status of a disk controller).

Write Operation or, data transfer with no source address i.e., the source data is a byte or word constant (literal) in the data assembly register of the 82258 (loaded during the setup routine with a low word out of the source pointer field). The write operation can be used to erase a memory/peripheral data block (or peripheral register) or to load it with a certain constant.

CHANNEL COMMANDS AND COMMAND BLOCKS

The 82258 controls the data transfer, with all its modifications, through the channel command blocks. These contain the channel command word and all the initial parameters for the data transfer execution. The channel start command from the CPU causes the 82258 to read the channel command block, with all its parameters from the memory and, to load them into the internal channel registers. The channel registers that are loaded via the command blocks are: CCR, SPR, DPR, BCR, TTPR,

Table 5. Data Manipulation Operations

Operation	Single Cycle	Two Cycle	Byte/Word Multiplex*	Block Multiplex*
	Bus Cycles Required**			
Masked Compare (Byte/Word)	2	2	2	2
Verify	N/A	2	N/A	2
Verify and Halt	N/A	2	N/A	2
Verify and Save	2	F	F	F
Translate	F	3	3	3
Transfer w/o Source or Destination	F	1	1	1
	Operation Allowed			
Command Chaining	Yes	Yes	Yes	Yes
List Data Chaining	Yes	Yes	No	No
Linked List Data Chaining	Yes	Yes	No	No
Assembly/Disassembly	No	Yes	Yes	Yes
Source Synchronization	Yes	Yes	Yes	Yes
Destination Synchronization	Yes	Yes	Yes	Yes
Free Running	No	Yes	Yes	Yes

* : The multiplexor channel can only run in the two cycle transfer mode.

** : Actual number of bus cycles may vary depending upon address boundary, hardware wait state number, pointer modification direction etc.

F : Fatal error is generated.

N/A : Not Allowed

LPR/MTPR, MASKR and COMPR (see the register description for details on these registers). After examining the channel command for programming errors, the data block transfer is executed if no errors are detected. After the transfer termination, the reason for the termination is displayed in a word in the channel command block (channel status). Optionally, the last values of the source and the destination pointers and the byte count register may also be written out to the command block (constituting a status block if enabled). The CPU should not access the channel's control space while the channel is active (not stopped).

There are two basic types of channel commands:

Type 1 Channel Command—Data transfer Operation (Transfer Channel Command).

Type 2 Channel Command—Control Operation (Organizational Channel Command).

A complete channel program consists of at least one channel command block with a type one command and one type 2 command (stop).

Type 1 Channel Commands And Command Blocks

A command block always specifies a data transfer operation. The type 1 channel command defines the task to be performed by the channel (see the channel command register for details). Simple block transfer is specified by the short channel command block (Figure 16), which also allows data chaining. For more complex operations, the standard block is expanded by a command and a block extension, forming a long channel command block (Figure 16). The command block is always pointed at by the command pointer. Each channel has its own command pointer. Enabling of the status block (a bit in the channel command extension) extends the long channel command block by a status field of 12 byte length. This status field is loaded by the 82258 after the termination of the block transfer (Figure 16).

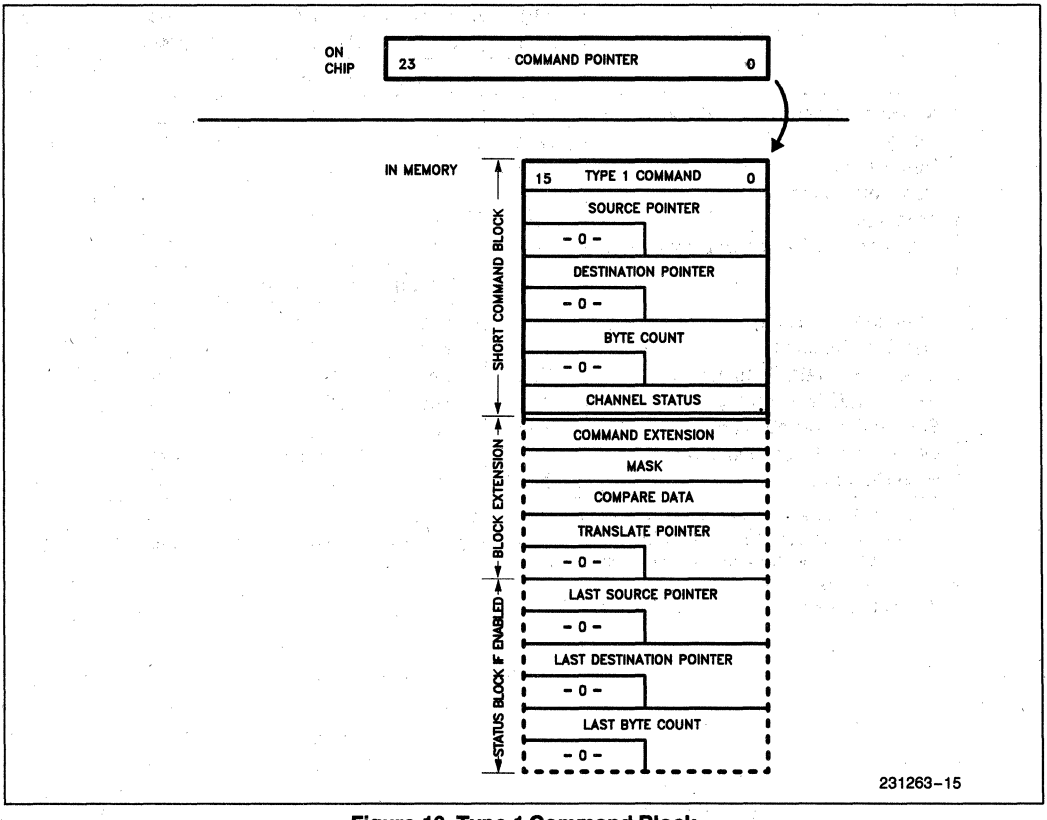


Figure 16. Type 1 Command Block

Type 2 Channel Commands and Command Blocks

The type 2 channel commands support the construction of channel programs by allowing operations such as auto-initialization, conditional chaining or program controlled interrupts. Figure 17 shows the structure of the type 2 channel command blocks.

The first word of the type 2 command block is the command and the second and the third may be an address.

Most of the type 2 commands can be executed conditionally; only exception being the unconditional stop which on the multiplexor channel functions as the Stop and Mask command. The 4 termination conditions are given in the CSR. If more than one condition is specified, the conditions are ORed. A special flag in the command word (I flag) allows to invert the channel status register bits before they are compared with the termination conditions. Table 6 gives the list of the different type 2 channel commands.

The type 2 commands can also activate a program controlled interrupt (INTOUT) and/or an $\overline{EOD}$ signal during the execution of a command (controlled by the ED and the IT flags). In the type 2 command the $\overline{EOD}$ is an asynchronous $\overline{EOD}$ (compared to the type 1 $\overline{EOD}$ which is synchronous to the last data transfer). If the ED or the IT flag is set, the signal generation is unconditional, independent of the condition code.

Table 6. Type 2 Channel Commands

Command
Relative Jump*
Absolute Jump*
Unconditional Stop
(Stop and Mask Subchannel for multiplexor channel)
Conditional Stop**

* : Both conditional or unconditional

** : The 82258 does not check if a selector channel only type 2 command is used on the multiplexor channel, but its execution will lead to erroneous channel processing.

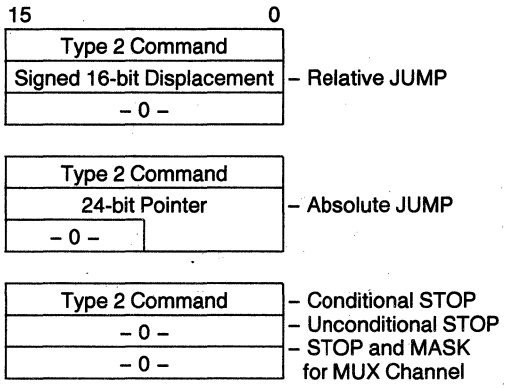


Figure 17. Type 2 Command Block

COMMAND AND DATA CHAINING

Command Chaining

The 82258 allows chaining of the command blocks in the memory, for any channel, for sequential execution. Figures 16 and 17 show channel command blocks and Figure 18 shows the examples of command chaining. The 82258 gets the address of the command block from its on-chip command pointer (initialized by the CPU) and starts executing. When it comes to the end of one command, it automatically starts to fetch and execute the next command block until a stop command is found. Conditional and unconditional STOP and JUMP commands allow complex sequences of DMAs to be performed.

Command chaining allows the 82258 to do CPU indepedent I/O processing, thus, saving valuable CPU time.

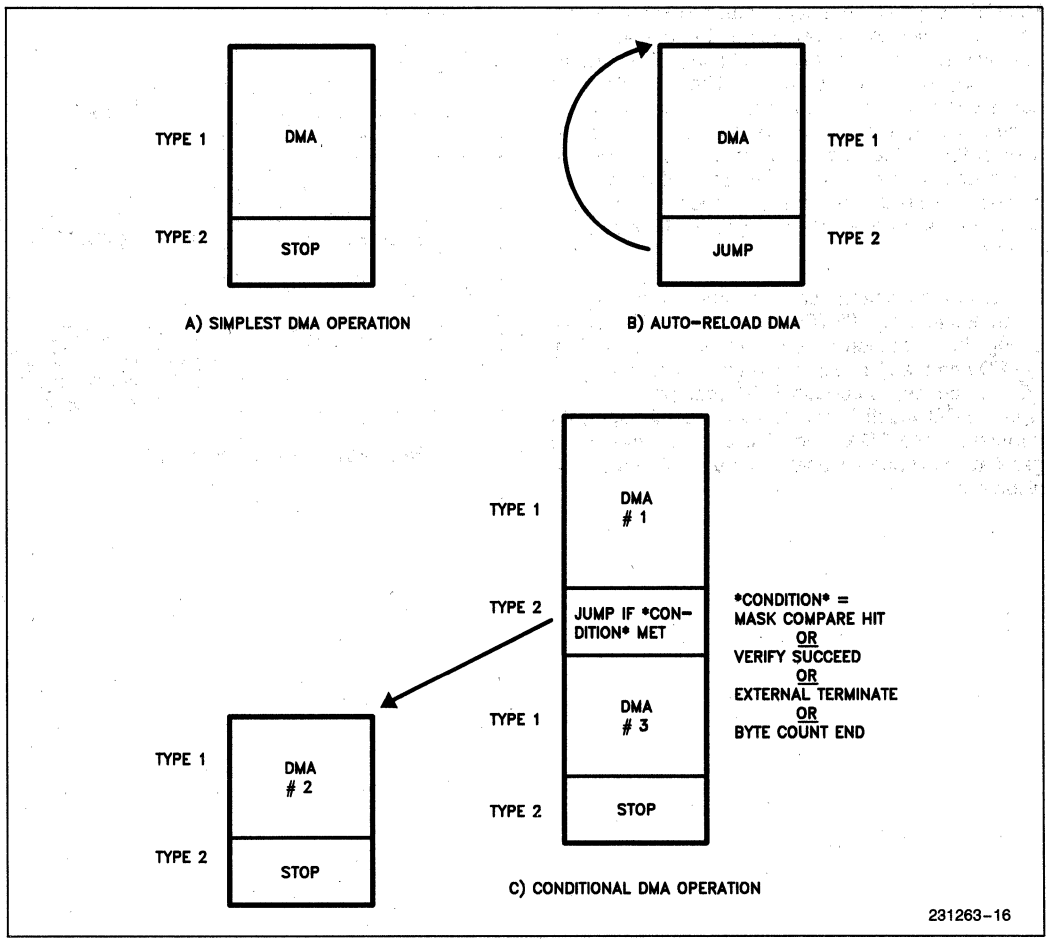


Figure 18. Command Chaining

Data Chaining

Data chaining allows gathering and scattering of data blocks. The 82258 permits automatic, dynamic linking of the data blocks scattered in the memory. Each data block in a chain can be up to 64K bytes. Two types of data chaining are allowed:

List Chaining: The chained data block descriptors are contiguous in a block which forms the data chain list (Figure 19). End of the chain is indicated by making the byte count field zero in the data chain list. List chaining is fast (1 microsecond between completion of one block transfer and going to the next element in the list, in an 8 MHz 80286 system) but not very flexible.

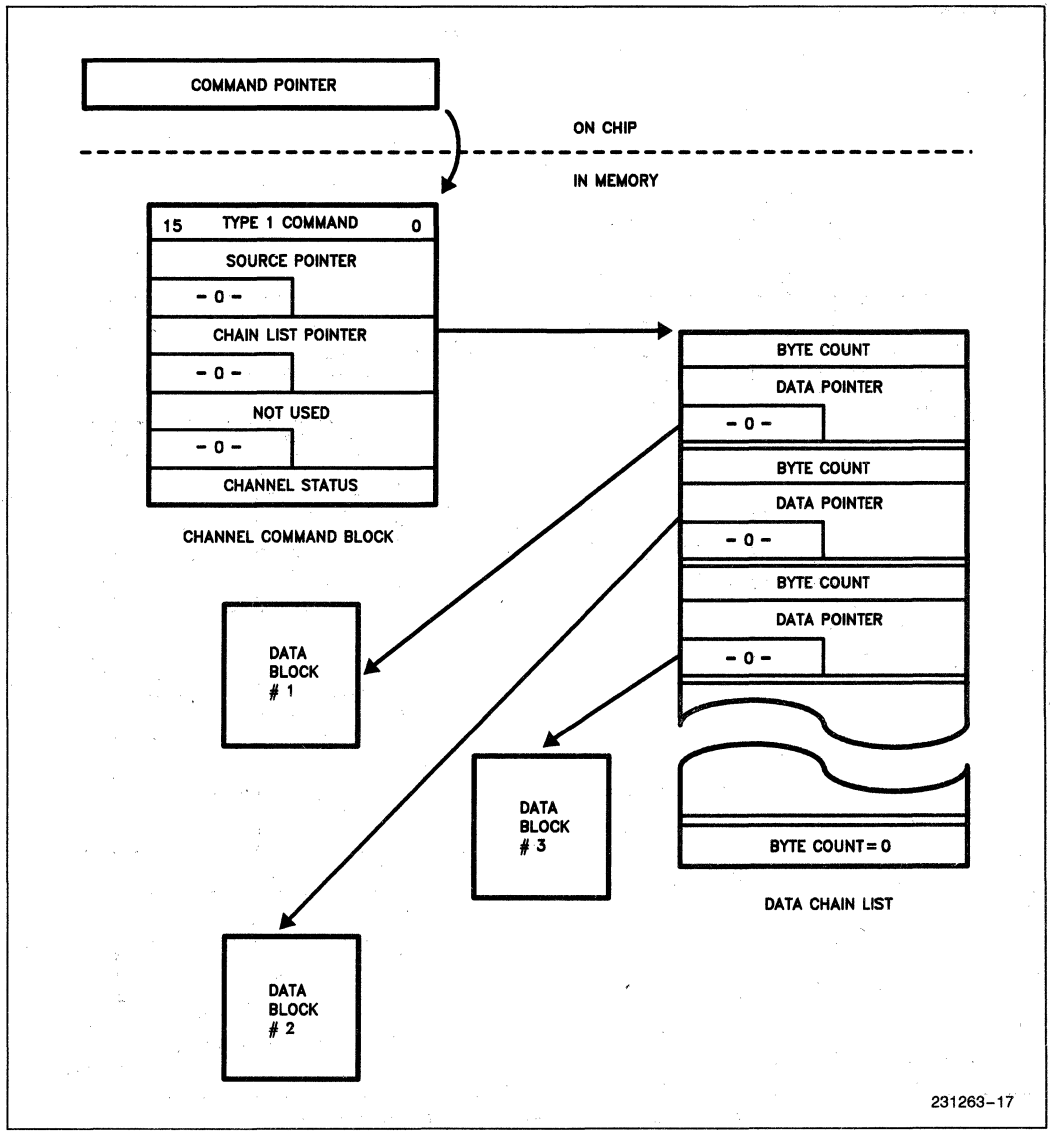


Figure 19. Destination List Chaining of Data

Linked List Chaining: Each list element which describes a particular data block (location and length) also holds a pointer to the next list element to be processed (Figure 20). End of the chain is indicated by making the byte count field zero in the linked list.

Linked list chaining is slower than the list chaining but the data blocks can be included, removed or, their sequence altered dynamically, through the link pointer manipulation by the CPU.

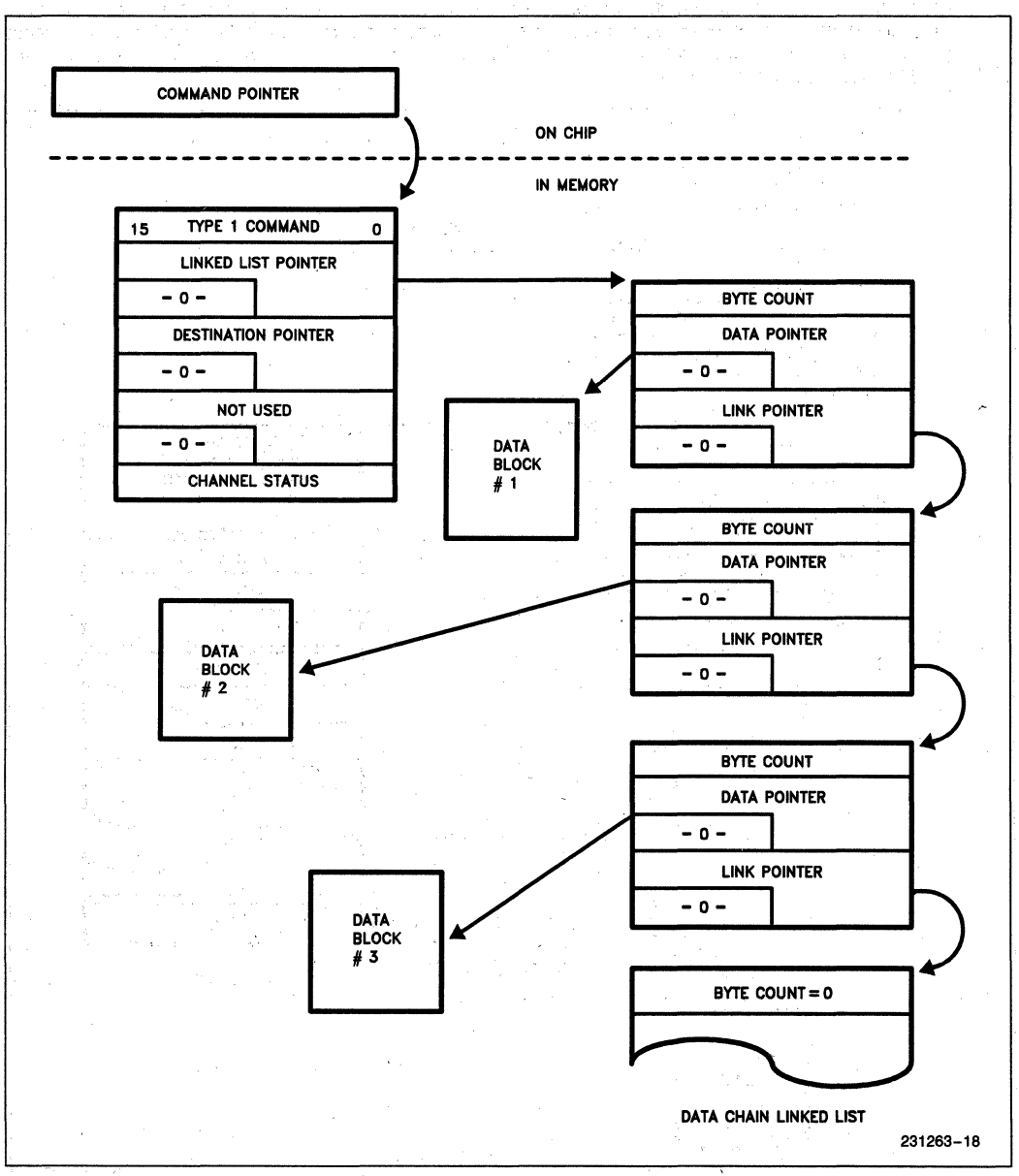


Figure 20. Source Linked List Chaining of Data

"ON THE FLY" OPERATIONS

The 82258 allows various data manipulation operations during the transfer:

Mask and Compare

Allows comparison of each byte, word or bit field (masking) in source data with some given pattern. Data transfer can be terminated on a match or a mismatch depending upon the program. This is possible both for the single and the two cycle transfer modes but, the transfer rate is halved in the single cycle mode.

Verify

No data transfer is performed, but the complete source data block is compared with a given data block. The data conversion can be terminated on mismatch (Verify and Halt). Supported only for the two cycle transfer mode.

Verify and Save

The data block is transferred from source to destination and in parallel compared with a given data block. The data transfer is not stopped on a mismatch. This operation is supported only for the single cycle transfer.

Translate

The source data (bytes) is translated with the aid of a translation table (Figure 21) before being sent to the destination. Translation is supported for the two cycle transfer mode only. If the destination is 16 bits, the two translated source bytes are assembled in the DAR before the destination cycle is executed.

Various 'on the fly' operations can be combined to allow the 82258 to perform versatile DMA operations.

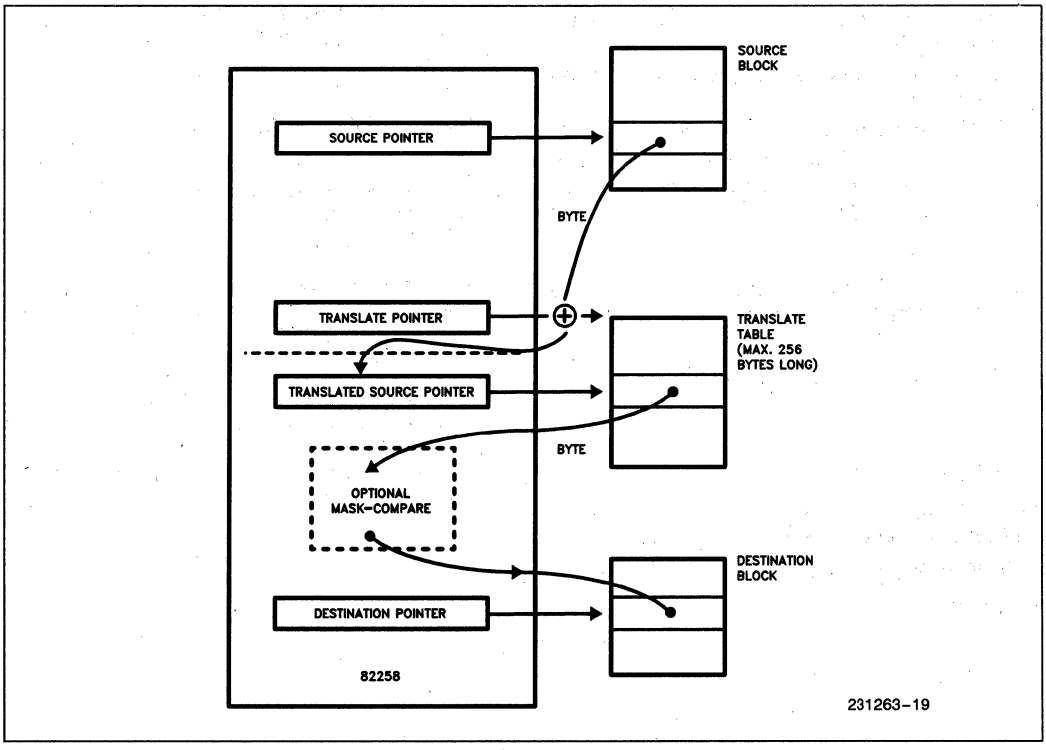


Figure 21. Translate Operation

PRIORITY CONTROL

The 82258 controls concurrent processing of its different channels (and subchannels) and, the internal and the external requests through a flexible priority scheme.

The PRI bits in the GMR are used to select the priority scheme which can be fixed or variable or a combination of the two (see the GMR description for the details). The unseparable bus cycles (e.g., 24 bit pointers) are not affected by the priority rotation. External 8259As determine the priorities for the multiplexed subchannels.

The processing of the internal or the external requests is controlled by a fully nested priority system including all four channels. Since more than one request can compete for the same channel, the requests are also prioritised in relation to their types as follows (in descending order of priority).

— Channel Stop (Command from the CPU out of the GCR)

— External asynchronous termination request (through $\overline{EOD}$)

— Internal continue request on previously interrupted sequence

— Start or stop subchannel or multiplexor channel

— Internal (without synchronization) or external (with synchronization) data service request or IO request for the multiplexor channel

— Channel wait (idle)

Data chaining and internal termination belong to the data service request processing, command chaining belongs to the termination processing.

Slave operations, where the 82258 is addressed by the CPU, have the highest priority of all the activities.

ADDRESSABILITY

The 82258 has two address spaces like the 80286, the 80186/188 and the 8086/88 processors:

— Memory space

— I/O space

Both the spaces are 16 MByte large for the 286/remote mode and 1 Mbyte for the 186/8086 mode. All types of transfers are possible:

— Memory/Memory

— I/O / I/O

— Memory/I/O

— I/O / Memory

Either of the memory or the peripheral can lie in either of the two spaces. Each space can be independently 8 bit or 16 bit wide. All possible Even-Odd boundary address combinations are supported for the data transfer from source (8 bit or 16 bit) to destination (8 bit or 16 bit) in the two cycle transfer mode. The source and the destination pointers can be incremented, decremented or not modified at all (INC/DEC bits of type 1 channel command in the CCR) after the corresponding data bus cycle. The 82258 does not indicate or check an 'address out of range' condition. Address overflow and underflow during a block transfer results in an address wrap around. Maximum length of the data block can be 16 MBytes in an 80286 system. In the 186/86 mode the maximum byte count is (1M-1). This is not checked by the 82258.

SYNCHRONIZATION OF DATA TRANSFER

The 82258 allows both the external synchronization of a DMA transfer (from a source or a destination device) or a free running DMA (internally synchronized).

The external synchronization allows control of input/output operations in the cycle of the peripheral device, hence occupying the bus only when the peripheral is able to receive or transmit data.

Free running DMA (no external synchronization) is used for the memory to memory transfers, during a continuous DMA request or, in the block multiplex subchannel after the channel start. It is not supported for the single cycle transfer mode.

286 PROTECTION

The 82258 needs special consideration to operate in an 80286 system in the protected mode. The 82258 works only with the real addresses but it supports a protected mode 80286 system if the following conditions are fulfilled:

— The 286 kernel software must check all the protection rules during the set up routine for the 82258 and perform the limit checks for the block transfers. This is supported by the 80286 instructions e.g. VERR (verify Read Access), VERW (verify Write Access), LSL (load Segment Limit).

— The 286 kernel has to translate the logical addresses into the physical addresses.

— All the 82258 registers should be memory mapped and access to them should be allowed only for a 286 kernel routine (task isolation).

Normally an I/O utility routine is provided by the operating system to service the 82258. No direct user access should be allowed to the 82258 from the lower privilege levels. The real addresses can be generated only by using the 286 protection mechanism and are so checked against any protection violation.

82258 REGISTER MODEL

The 82258 has three sets of registers (Figure 22):

General Registers
Channel Registers
Multiplexor Channel Registers

All registers can be read or written into by the CPU but, most are accessed only for the test purposes. The CPU loads some registers (e.g. General Mode Register) during the initialization after the reset, and others during the invocation of a channel (General Command Register). Some of the channel registers are programmed or read by the CPU but most of them are loaded by the 82258 itself during the setup routine after a channel start. All accessible registers can be accessed bytewise or wordwise by the CPU.

Figure 23 gives a layout of the registers. Note that all registers lie on even addresses.

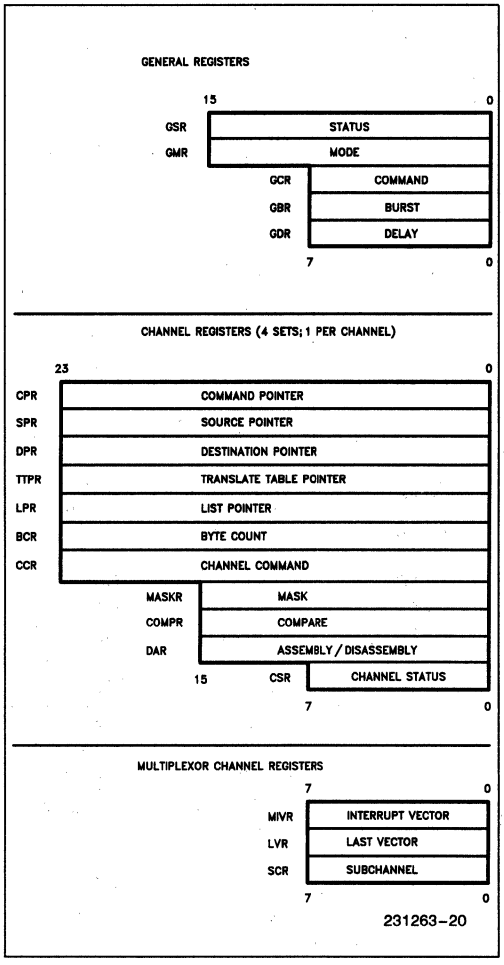


Figure 22. 82258 Register Set

Address Bits 5-0 (hexadecimal)	Address Bits 7,6				Address Bits 5-0 (binary)
	00	01	10	11	
0	GCR	RESERVED	RESERVED	RESERVED	000000
2	SCR				000010
4	GSR				000100
6	RESERVED				000110
8	GMR				001000
A	GBR				001010
C	GDR				001100
E	RESERVED				001110
10	CSR0	CSR1	CSR2	CSR3	010000
12	DAR0	DAR1	DAR2	DAR3	010010
14	MASKR0	MASKR1	MASKR2	MASKR3	010100
16	COMPR0	COMPR1	COMPR2	COMPR3	010110
18				MIVR	011000
1A	RESERVED	RESERVED	RESERVED	LVR	011010
1C				RESERVED	011100
1E					011110
20	CPRL0	CPRL1	CPRL2	CPRL3	100000
22	CPRH0	CPRH1	CPRH2	CPRH3	100010
24	SPRL0	SPRL1	SPRL2	SPRL3	100100
26	SPRH0	SPRH1	SPRH2	SPRH3	100110
28	DPRL0	DPRL1	DPRL2	DPRL3	101000
2A	DPRH0	DPRH1	DPRH2	DPRH3	101010
2C	TTPRL0	TTPRL1	TTPRL2	TTPRL3	101100
2E	TTPRH0	TTPRH1	TTPRH2	TTPRH3	101110
30	LPRL0	LPRL1	LPRL2	LPRL3/MTPRL	110000
32	LPRH0	LPRH1	LPRH2	LPRH3/MTPRH	110010
34	RESERVED	RESERVED	RESERVED	RESERVED	110100
36	RESERVED	RESERVED	RESERVED	RESERVED	110110
38	BCRL0	BCRL1	BCRL2	BCRL3	111000
3A	BCRH0	BCRH1	BCRH2	BCRH3	111010
3C	CCRL0	CCRL1	CCRL2	CCRL3	111100
3E	CCRH0	CCRH1	CCRH2	CCRH3	111110

GCR = General Command Register
SCR = Subchannel Register
GSR = General Status Register
GMR = General Mode Register
GBR = General Burst Register
GDR = General Delay Register
CSR = Channel Status Register
DAR = Data Assembly Register
MASKR = Mask Register
COMPR = Compare Register
L = Low Word
H = High Byte

MIVR = Multiplexor Interrupt Vector Register
LVR = Last Vector Register
CPR = Command Pointer Register
SPR = Source Pointer Register
DPR = Destination Pointer Register
TTPR = Translate Table Pointer Register
LPR = List Pointer Register
MTPR = Multiplexor Table Pointer Register
BCR = Byte Count Register
CCR = Channel Command Register
0, 1, 2, 3 = Channel Number

Figure 23. Layout of Register Addresses

GENERAL REGISTERS

These registers are common to all the channels.

General Mode Register (GMR)

This is the first register to be programmed after the reset since it describes the 82258 environment. Here the system wide parameters are specified. The 16 bit register is loaded bytewise with the low byte being programmed first.

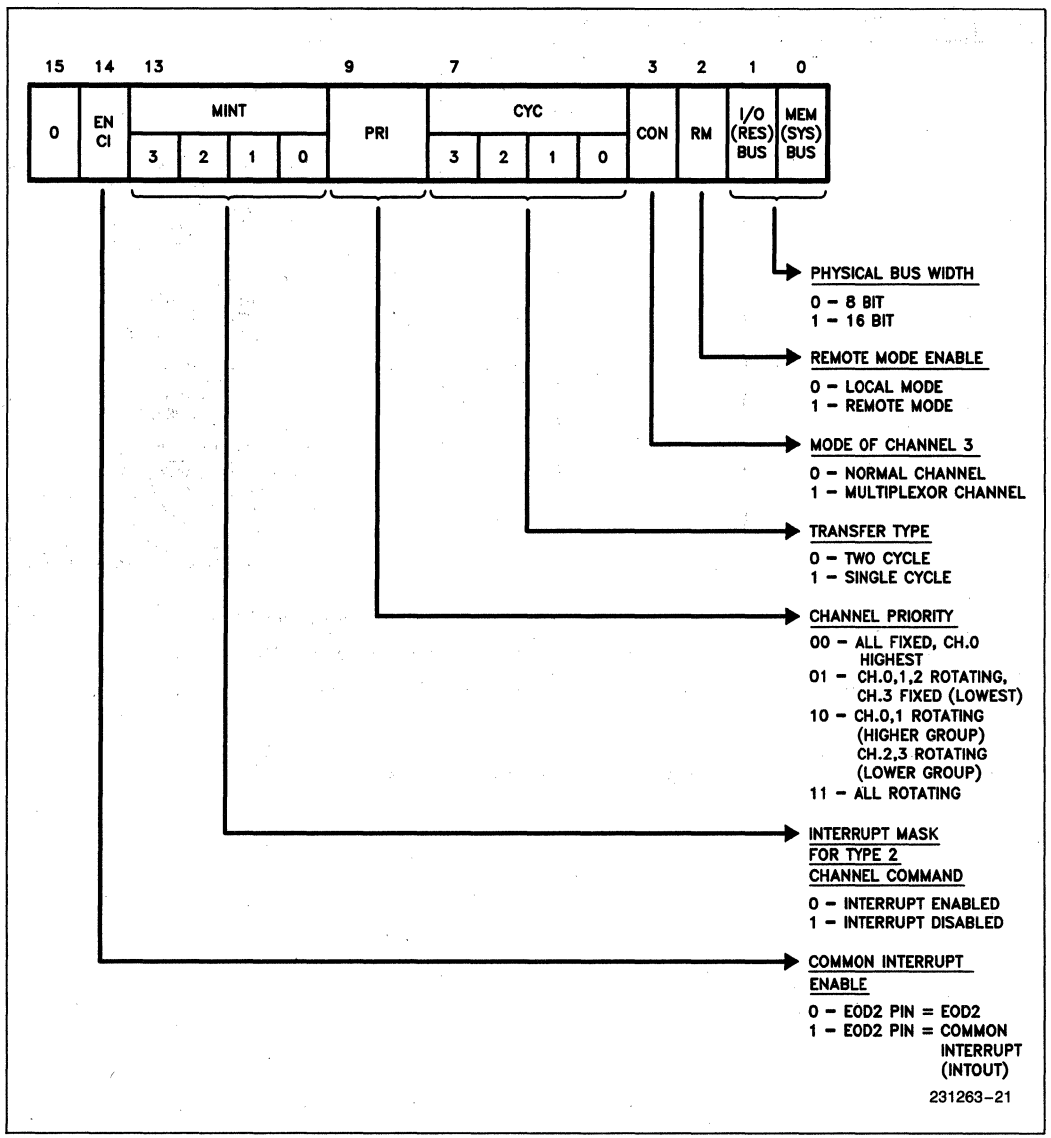


Figure 24. General Mode Register

General Status Register (GSR)

This register provides the status information for all the channels. It also shows which channels have interrupts pending and, where the channel control space lies. It is a 16 bit register.

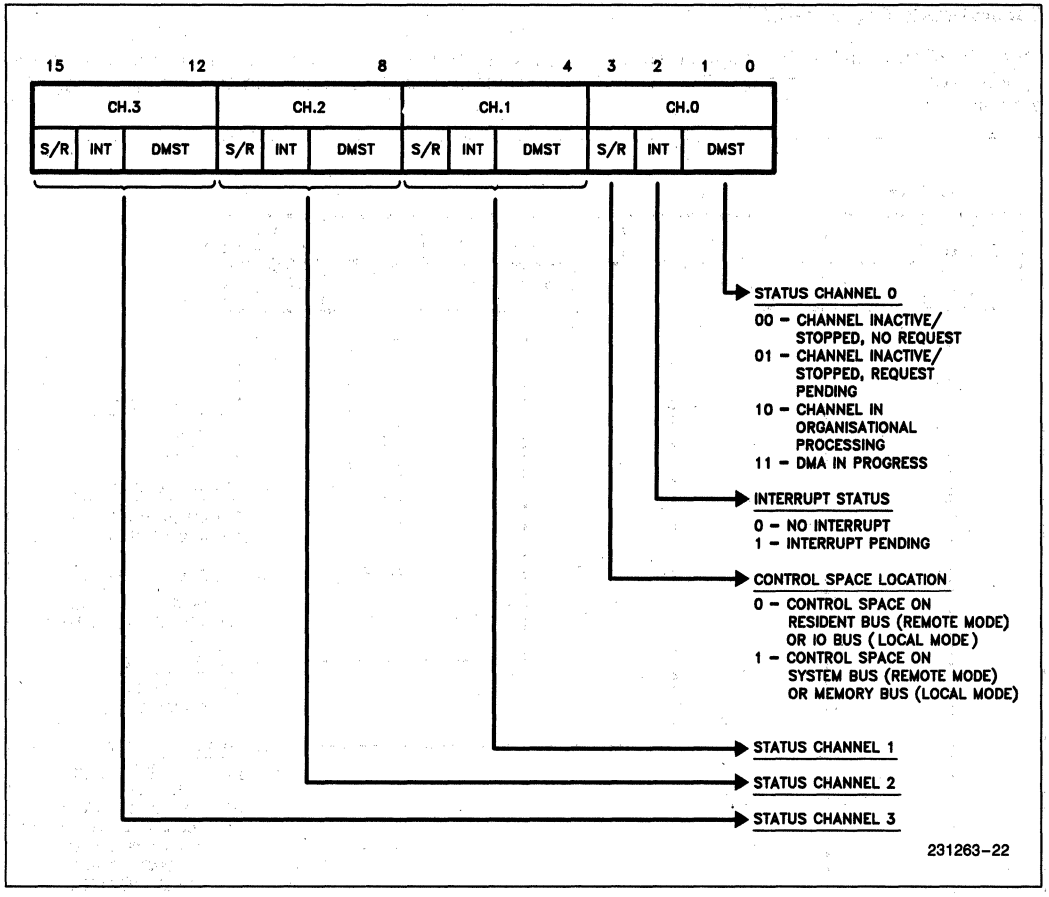


Figure 25. General Status Register

General Command Register (GCR)

GCR is an 8 bit register directly loaded by the CPU to start or stop a channel. The START command also defines the control space assignment. The pending interrupt from any channel is also cleared through the GCR. Any combination of channels can be addressed simultaneously. To start/stop a multiplexor subchannel, the subchannel number must be first loaded in the Subchannel Register (SCR). The Halt/single step command is useful for the system debugging.

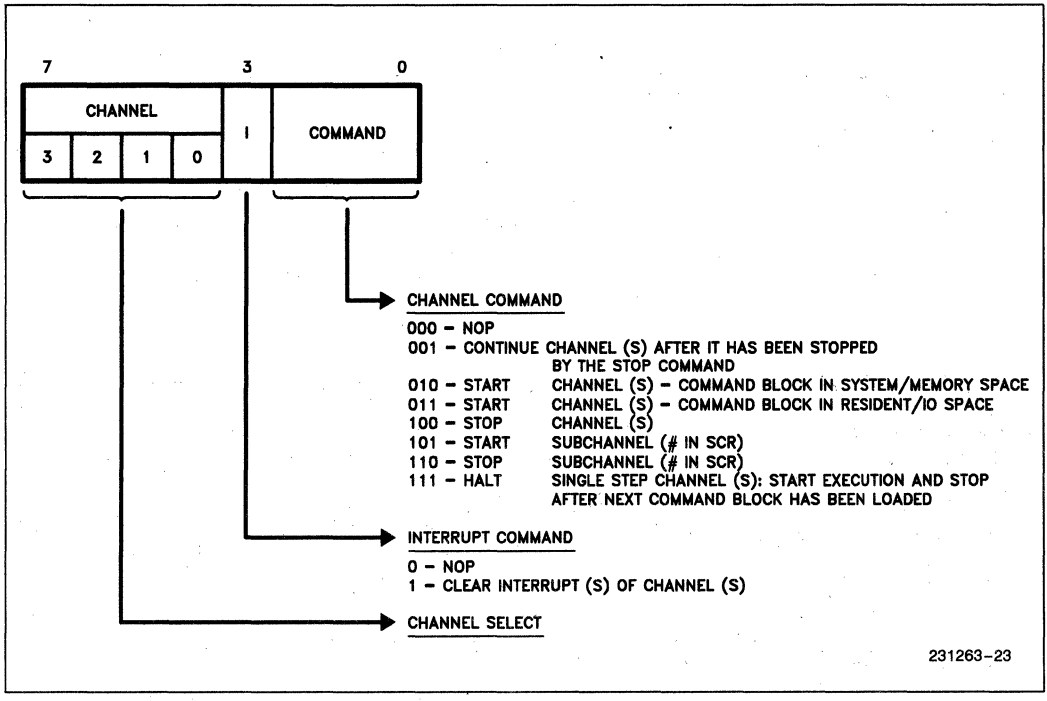


Figure 26. General Command Register

General Burst Register (GBR)

This 8 bit register determines the maximum number of contiguous bus cycles that can be requested by the 82258. GBR = 0 means unlimited contiguous bus cycles for the 82258. The GBR must be directly loaded by the CPU.

General Delay Register (GDR)

GDR is an 8 bit register which determines the minimum number of clocks between the 82258 burst accesses. GDR = 0 means no minimum delay between the HOLD request.

Burst/Delay Algorithm

Both the GBR and the GDR do their actual counting through their respective counters the GBC and the GDC. For the burst and delay counters, the following rules apply:

— Whenever the 82258 controls a bus cycle the burst counter is decremented by one but not beyond zero.

— Whenever the 82258, in the local mode, does not have the bus, the delay counter is decremented by one: every second T-state in the 286 mode or, every fourth T-state in the 186 mode.

— Whenever the delay counter is zero, the burst and the delay counters are loaded from the burst and the delay registers.

— If the burst counter is zero (and no exception occurs), the 82258 releases the bus and the delay counter counts until it is zero. Then both counters are loaded from their corresponding registers and the 82258 can again request the bus by activating HOLD signal. Unseparable bus cycles are the exception to this rule. Counting of the burst is not prevented but surrendering of the bus is.

— In the remote mode the burst and the delay are relevant only for the system bus cycles. The GBC is only decremented while the 82258 performs the system bus cycles and the GDC decrements when the 82258 does not control the system bus (idling or the resident bus cycles).

CHANNEL REGISTERS

Each of the four 82258 channels has these registers. All the channel registers are loaded by the 82258 from the memory except the Command Pointer (CPR) [Multiplexor Table Pointer (MTPR) & Subchannel Register (SCR) for the channel 3 in the multiplexor mode]. The initial contents of the registers are specified, by the CPU in the command blocks in the memory.

Command Pointer Register (CPR)

This 24 bit register contains the physical address of the command block. It must be loaded by the CPU before starting the channel. For the channel 3 in the multiplexor mode, the CPR is loaded by the 82258 from the multiplexor table (MT) in the memory.

Source Pointer Register (SPR)

SPR is 24 bits and contains the physical address of the source (memory or I/O, system or resident space) in a DMA transfer. In the single cycle transfer mode, it contains the only address pointer (source or destination).

Destination Pointer Register (DPR)

DPR contains the physical address of the destination (memory or I/O, system or resident space) in a DMA transfer. During Verify operations it contains the verify pointer (pointer to compare the data block). For the single cycle transfer mode, it is only used for the verify and save operation. It is a 24 bit register.

Translate Table Pointer Register (TTPR)

This 24 bit register is used to reference the translate table in the memory when the translate function is enabled in the channel command register extension (CCRX).

List Pointer Register (LPR)

LPR is used for data chaining (list and linked list) operation. It is a 24 bit register and points to the list element. In the multiplexor mode for the channel 3, it is used as the Multiplexor Table Pointer Register (MTPR). (Multiplexor mode does not support data chaining).

Byte Count Register (BCR)

BCR is a 24 bit register and contains the byte count for the DMA transfer.

Channel Command Register (CCR)

CCR specifies the type of DMA transfer or the type of internal operation. The channel commands are contained in a channel command block. The 82258 has two types of channel commands:

— Type 1 for data movement

— Type 2 for command chaining control

The channel command register has three configurations:

— Short Type 1 command: SYN field NE. 00 and ECX = 0. Upper 8 bits, i.e., Channel Command Register Extension (CCRX field), are not valid.

— Long Type 1 command: SYN field. NE. 00 and ECX = 1. All 24 bits are valid.

— Type 2 command: SYN field = 00, Upper 8 bits (CCRX field) are not valid.

Figure 27 shows CCR for Type 1 command and Figure 28 has the CCRX (Channel Command Register Extension). Figure 29 shows CCR for type 2 command.

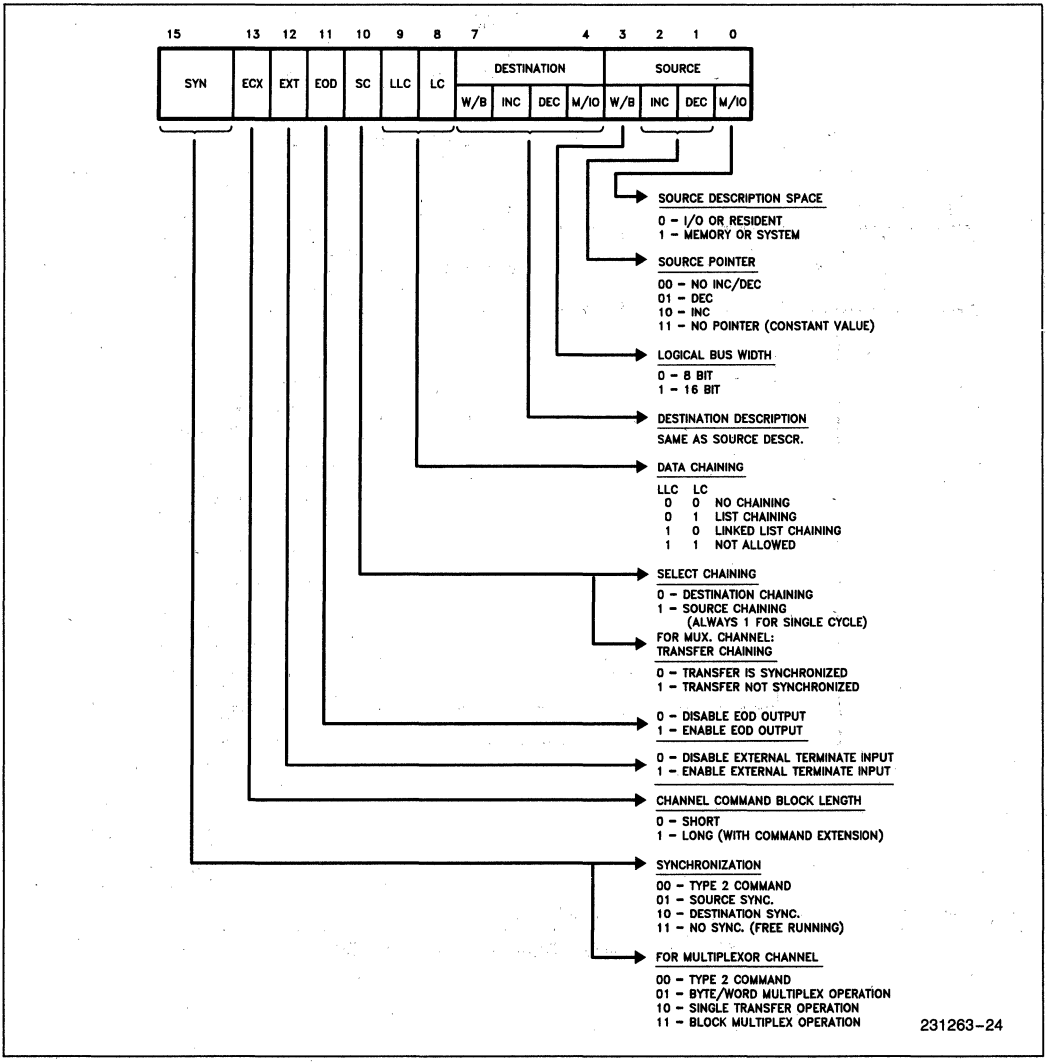


Figure 27. Type 1 Channel Command CCR

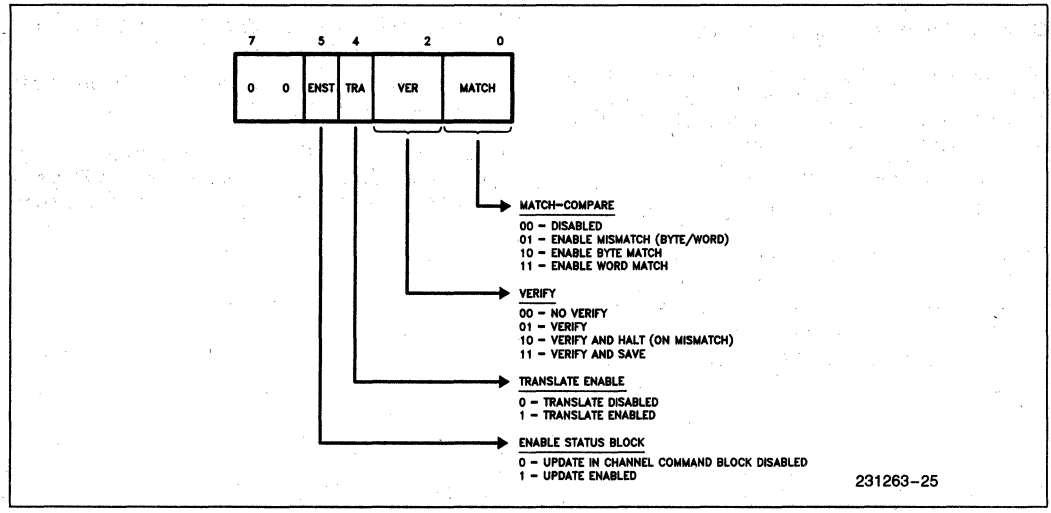
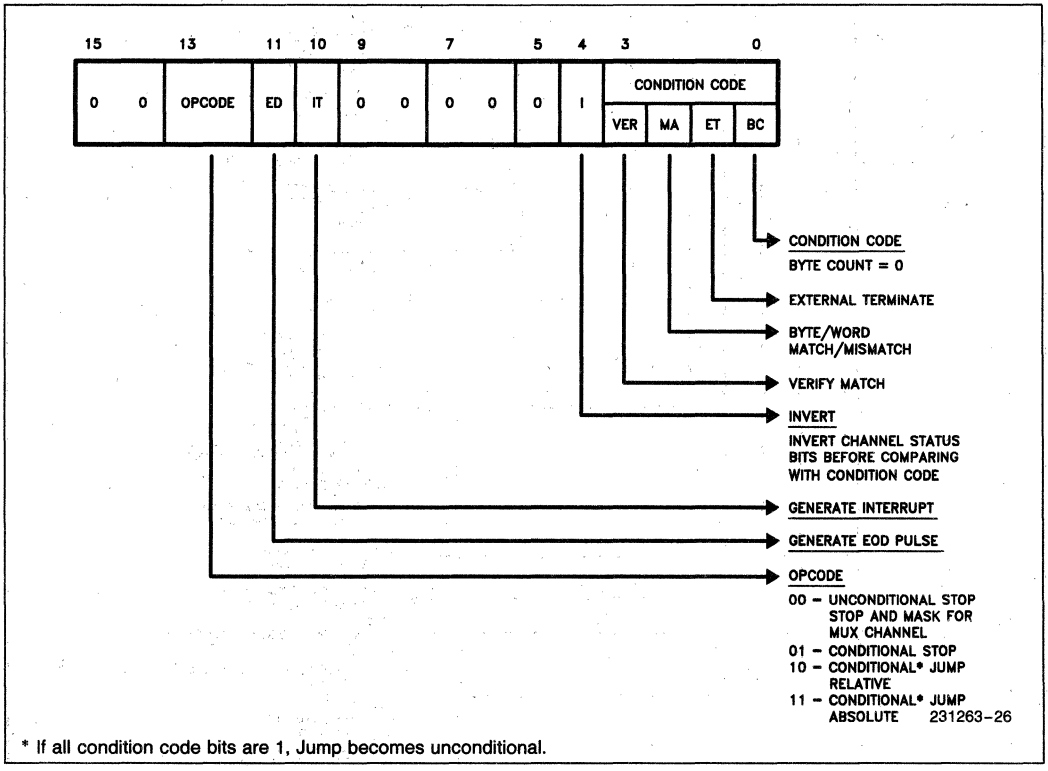


Figure 28. Channel Command Register Extension CCRX



* If all condition code bits are 1, Jump becomes unconditional.

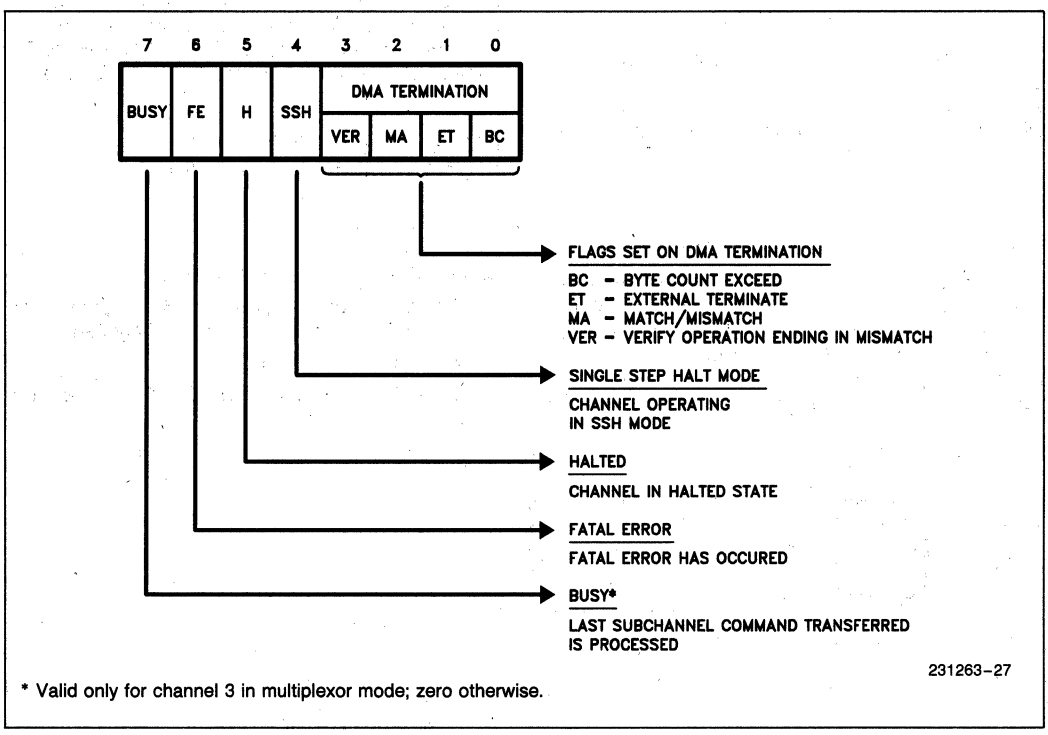
Figure 29. Type 2 Channel Command CCR

Mask Register (MASKR) and Compare Register (COMPR)

Both of these registers are 16 bit and are used during the match/mismatch operation. For comparison with the transferred data, only those bit positions in the Compare Register which are not masked with 1's in the Mask register are considered. These two registers together allow byte, word or bit level comparisons. MASKR is also used during the verify operations. MASKR and COMPR each should contain two identical bytes for Byte Match/Mismatch operations.

Channel Status Register (CSR)

CSR, an 8 bit register, reflects the status of the channel. The least significant half byte is the termination condition and the most significant half byte indicates fatal error, busy state and halted state.



* Valid only for channel 3 in multiplexor mode; zero otherwise.

Figure 30. Channel Status Register

Data Assembly Register (DAR)

This 16 bit register is used for automatic assembly/disassembly of data.

Mutiplexor Channel Registers

These registers are valid only for channel 3, when used as a multiplexor channel.

Multiplexor Table Pointer (MTPR)

This register is used to reference the multiplexor table in the memory when channel 3 is programmed as a multiplexor channel. Since data chaining is not allowed for the multiplexor channel, the List Pointer Register (LPR) is used as the MTPR. MTPR is 24 bit and must be loaded by the CPU.

Multiplexor Interrupt Vector Register (MIVR)

This 8 bit register is used by the CPU to determine which channels are stopped. The vectors of the stopped subchannels are output in the priority order (0 has the highest priority) upon each reference of this register, until the NV bit is set. A maximum of 32 vectors can be distinguished.

Last Vector Register (LVR)

LVR gives the last vector read by the 82258 (from the 8259A). In case of a fatal error stop of channel 3, LVR determines the guilty subchannel. LVR is an 8 bit register.

Subchannel Register (SCR)

This register gives the 8 bit subchannel number for the general commands START/STOP Subchannel. It must be loaded by the CPU before a subchannel command is written into the GCR. MIVR limits the number of subchannels supported to 32 (5 bits).

82258 OPERATION AND PROGRAMMING OVERVIEW

INITIAL STATE

Upon activation of the RESET signal:

— all channels are disabled (by clearing the DMA status bits in the General Status Register)

— all bus activities are stopped

— all tristate signals are tristated and the others enter the inactive state

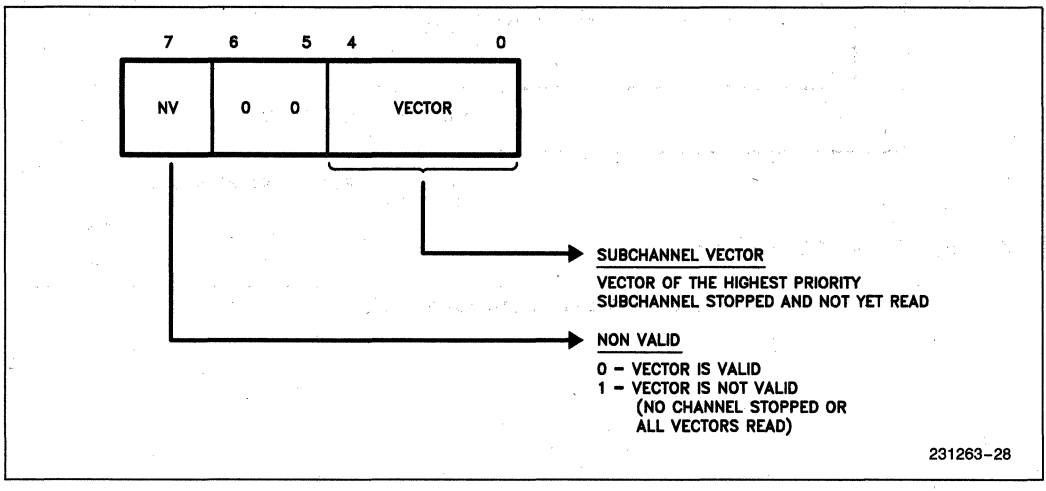


Figure 31. Multiplexor Interrupt Vector Register

After the RESET signal becomes inactive, the 82258 state gets defined:

— it is in the 186 mode if A23 pin was low at the falling edge of RESET; otherwise it is in the 286 mode

— it is in the 8086 max (Request/Grant) mode if the 186 mode is detected and HLDA pin was high at the falling edge of RESET; otherwise it is in the 186/8086 Min. (HOLD/HLDA) mode.

— The contents of the 82258 registers are as follows:

- GMR: All bits are zero
- GBR: Zero value
- GDR: Zero value
- GSR: — DMST bits for channels: 0X (Stopped)
  — INT for all channels: 0 (no interrupt pending)
  — S/R = 0 (I/O or resident space)
- All Channel Status Registers (CSR): Zero Values
- MIVR: NV = 1 (Vector not valid)
  — Vector is all 1, rest zero
  — All stop bits in matrix are reset
- All other registers (GCR, LVR, SCR, CPRn, SPRn, DPRn, TTPRn, LPRn, BCRn, CCRn, COMPRn, MASKRn, MTPR) are undefined

INITIALIZATION AND CHANNEL INVOCATION

After RESET, the 82258 has to be initialized by the CPU. The General Mode Register (GMR) should be loaded first in the 16 bit systems; the lower byte of the GMR (which gives main configuration information) in the 8 bit systems.

SYSBUS (MEMBUS) bit of the GMR determines the physical bus width of the CPU-82258 communication. All register write and read operations are executed:

— Bytewise on the lower half of the data bus (D7–D0), if SYSBUS (MEMBUS) = 0

— wordwise on D15–D0 if SYSBUS (MEMBUS) = 1. Byte transfers are also possible here with the bytes being transferred on that half of the data bus which is addressed by the least significant bit of the register address.

Internally the 82258 uses $\overline{BHE}$ and A0 to detect the effective transfer width of the 82258—CPU communications. After the GMR, the General Burst Register (GBR) and the General Delay Register (GDR) should be programmed, if needed (Initial state = 0 for both), by the CPU.

Before a channel is invoked, the control space in the memory and the channel registers in the 82258 have to be initialized:

Selector Channel Start

Following conditions should be met:

— channel program in the control space

— if data chaining enabled, the chaining list or the linked lists in the control space

— if translate enabled, the translate table in the control space

— load the CPR with the start address of the channel program

Multiplexor Channel Start

For the multiplexor channel operation, the following is essential:

— the multiplexor table MT in the control space with the subchannel command pointer and the mask register pointer of the associated 8259A for each subchannel

— initialization of the 8259A's mask registers by masking off all the request inputs. In the remote mode, this can also be done by the data transfer operation on the selector channel (or by stop subchannel commands)

— load MTPR with the base address of the multiplexor table (MT)

For the subchannel start

— the subchannel program should be in the control space

— if translate enabled, the translate table should be in the control space

— the subchannel command pointer should be in the multiplexor table

— read the multiplexor channel status register CSR3. Write a new subchannel number into the SCR only if BUSY bit = 0.

In case of a normal channel start, the last CPU operation is to write the general command into the GCR. Then the start will be processed by the 82258 according to the requested channel's priority, with the highest priority being processed first. If the addressed channel is already active, the start command is ignored. If I = 1 in GCR, the INT bit(s) of the indicated channel(s) will be erased in the GSR.

COMMAND EXECUTION

Selector Channel: The command bits in the GCR give the commands available to a selector channel. Execution of the continue and the start commands is prioritized; the stop commands are executed immediately. The stop command forces the DMA status bit (DMST) in the GSR to channel inactive (stopped) without any additional routine. The continue command works directly with internal stored register parameters and continues a previously stopped channel operation. The start commands define the location of the control space and initiate the set up routine. The halt command has multiple functions:

— It forces the channel into the single step and halt mode, indicated by the SSH bit in the CSR

— If the channel is running, it will be halted after the completion of the current command block execution; the halted data is shown by the H bit of the CSR; the DMST bits of the GSR are not changed

— If the channel is halted (or stopped) the halt/single step command starts the channel, and the channel will again be halted after the completion of the next command block execution (type 1 or 2)

The single step and halt mode is finished by a start or a continue command. After a channel start, first the general status reflected in the GSR is changed into 'DMA in organizational processing'. GSR also indicates the location of the control space (S/R bit). After the prioritization of the start command, the channel's set up routine is executed.

After the set up routine execution, all the transfer parameters are accessible in the 82258 internal registers. The SYN bits in the CCR decide:

— if the channel activity is continued by an immediate start of the data transfer (i.e., free running mode or an internal data transfer service request)

— or the channel is waiting for a DMA request i.e., external synchronization mode.

Multiplexor Channel: On the multiplexor channel, there are two cases:

a. The whole channel has to be treated by a general command

b. Only the addressed subchannel has to be treated by a general command

a. In case of the whole channel, the commands are the same as the selector channel commands. Execution of the continue and the stop (stops whole channel) is the same. The channel 3 start command has only two functions:

— specify whether the system/memory or the resident/IO control space has to be used on the multiplexor channel (S/R bit in GSR)

— change of the general status of the channel 3 (DMST bits in GSR) into "Channel started but idling" thus, enabling the IOREQs and the Subchannel commands.

The general channel command "Halt/Single Step" has a slightly different interpretation for the multiplexor channel. While the selector channel can only be halted during the chaining of the command blocks, the multiplexor channel in the single step/halt mode will also be halted when it takes the idle state. In that case, a new halt/single step command will only be executed if an IOREQ or a subchannel start/stop command is pending.

b. With the start subchannel command, the 82258 unmasks the corresponding bit in the 8259A mask register for the addressed subchannel, thus enabling the subchannel. The BUSY bit in the CSR is set indicating the state: "subchannel command pending". After prioritization, the subchannel routine is executed. When an I/O request is received on the subchannel, the command pointer is fetched from the MT and the channel's set up routine is executed. After the reset of the BUSY bit, a new start/stop subchannel command can be accepted by the multiplexor channel.

Only distinction between the stop subchannel command and the start subchannel command is the handling of the mask bit in the 8259A. For the STOP command, the vector specific mask bit is set by the 82258. As the start command, the stop command has also to be prioritized before execution.

For the multiplexor channel the following rules are observed:

— Before any IOREQ can be processed, the whole channel 3 has to be started and the channel 3 must be in the idle state

— In any state a subchannel command can be accepted and transferred into the state "subchannel command pending"

— A pending subchannel command can be processed only in the idle state

— In the idle state, a subchannel command has a higher priority than an IOREQ

— In case of a fatal error stop of a subchannel, the whole channel 3 is stopped. LVR identifies the guilty subchannel. To stop (mask) this subchannel, the CPU at first has to issue a START CH3 command and then stop the affected subchannel.

TERMINATION CONDITIONS

The 82258 distinguishes the following conditions for termination of a block transfer:

— byte count is zero and the data chaining not enabled; a standard termination condition

— data chaining enabled and the new fetched byte count is zero

— external termination via the channel's $\overline{EOD}$ line if enabled by the EXT bit in the CCR

— match/mismatch during the masked byte or word compare, as specified and enabled in the command extension CCRX

— mismatch during a verify & halt operation, as specified and enabled in the command extension CCRX

— The CPU loading the GCR with a stop command, though the channel is not really terminated.

INTERRUPT CONTROL

The 82258 has four programmable $\overline{EOD}$ pins (one for each channel) for the CPU interruption and for communication with the system environment. As inputs, the $\overline{EOD}$ pins are used for external termination, enabled by the EXT bit of the type 1 channel command in the CCR. When used as output, the $\overline{EOD}$ pins provide two basic functions:

$\overline{EOD}$ (end of DMA), a channel specific active LOW pulse signal of 2 T-states length, always enabled by the software. With a type 1 channel command, $\overline{EOD}$s, if enabled, are synchronous and always controlled by the byte count. If data chaining is enabled, type 1 $\overline{EOD}$s should not be used for interrupts since multiple $\overline{EOD}$s (with every exceeding byte count) are issued. With a type 2 command, the $\overline{EOD}$, if enabled (ED = 1 in the CCR), is an asynchronous signal generated after a command execution.

INTOUT (interrupt output) is a hardware generated (error detection) or a software enabled static active HIGH signal on the $\overline{EOD}$2 pin, if programmed (ENCI = 1 in the GMR). The channel generating the INTOUT is indicated by the INT bit in the GSR. Hardware generated interrupt occurs in case of a fatal error (INTOUT issued if not masked by the MINT bit in the GMR). Type 2 channel command allows software generated INTOUT if programmed (IT = 1 in the CCR and not masked by the MINT bit in the GMR). A channel's INT bit in the GSR is activated independent of the MINT (in GMR). INTOUT remains active until all INT bits in the CSR are reset by the CPU with the general command CLEAR INTERRUPT.

Multiplexor Channel Interrupts

Interrupts from the multiplexor channel belong to a certain subchannel. For program controlled interrupts, the status and the context information cannot be fetched from the internal 82258 registers (since the multiplexor channel is not stopped). Hence, the CPU can only investigate the interrupt via the MIVR register. After the MIVR read from the CPU, the valid bit and matrix stop bit (the vector of which was indicated in the MIVR) are erased. For multiple stop conditions in the stop matrix, the stopped subchannels get their vectors in the MIVR in the priority order (highest for vector zero). The MIVR is activated independent of the programming of $\overline{EOD}$ or INTOUT. Therefore, the CPU can sample the MIVR in a polling mode when neither $\overline{EOD}$ nor INTOUT is used. With the interrupt vector out of the MIVR, the CPU finds the related command pointer (in MT) which points to the last executed channel command (stop and mask). For status information of last block transfer, the CPU has to find the last type 1 command block in the channel program. Programmable intermediate interrupt messages should not be used on the multiplexor subchannels (MIVR is activated only for the stopped subchannel).

For hardware generated INTOUT the whole channel 3 is stopped with the LVR indicating the last (guilty) vector. After the error investigation the CPU should start the channel 3 and then stop the affected subchannel.

FAULT DETECTION

On detecting a fatal error, the 82258 does the following:

immediately stops the affected channel
sets error bit in the channel's status register
sets channel specific INT bit in the GSR
sends interrupt if not masked (in GMR)

For error investigation, the CPU should:

read GSR (what channel?, channel stopped?)
read CSR (error?)
read CPR and investigate the channel command (type 1 command)
read LVR for multiplexor channel, if affected (what subchannel?)

The 82258 recognizes only type 1 command errors. Other error types are defaulted into non-fatal errors and not identified. The FE bit in the CSR indicates the fatal errors.

Fatal Errors: Fatal errors are detected during the decoding of a type 1 channel command with the GMR. Six conditions are used for detection and the allowed six combinations of them lead to six different transfer executions (Table 7). All other combinations of the six conditions generate a fatal error.

Table 7. Fatal Error Detection

Valid Combination	Conditions Decoded						Operation Performed
	Single Cycle	No Dst. Ptr.	No Src. Ptr.	Verify & Save	Trans-late	Sync. Error	
1	False	False	False	False	False	—	Two Cycle DMA
2	False	False	False	False	True	—	Translate
3	False	False	True	False	False	False	No Source Ptr. DMA
4	False	True	False	False	False	False	No Dest. Ptr. DMA
5	True	False	False	False	False	False	Single Cyc. DMA
6	True	False	False	True	False	False	Verify & Save

The synchronization error is predecoded and activated in the following cases:

Single cycle combined with free running

No source pointer mode combined with the source synchronization on a selector channel

No destination pointer combined with the destination synchronization on a selector channel

Non Fatal Errors and Undetected Fatal Errors

A non fatal error is not indicated in the channel status register. It is only defaulted. Channel processing is not interrupted. Following are some examples of non fatal errors and the undetected fatal errors:

Fault	Action
Remote mode + 186 mode	RM not inhibited but read/write pins are also used as outputs
Both list chaining and linked list chaining enabled	Linked list data chaining executed
Start/Stop subchannel and BUSY active	New command overwrites old command (Fatal Error)
Data chaining enabled on the multiplexor channel	MTPR is overwritten with the list pointer (Fatal Error)

TRANSFER RATES

Selector Channel

Table 8 illustrates the different transfer rates (in MBytes/sec) for the 286 mode of operation. These transfer rates are not affected by switching channels and are halved for both 186 and 86 modes of operation.

Table 8. Cummulative Selector Channel Transfer Rates (8 MHz 286 System)

Transfer	Single Cycle	Two Cycle
Word → Word	8	4
Word → Byte	not possible	2.66
Byte → Word	not possible	2.66
Byte → Byte	4	2
Byte → Byte w/ Translate	not possible	800 KBytes

Multiplexor Channel

The transfer rates on the multiplexor channel are different from the selector channel and depend on the mode of operation and the size of the command block.

Table 9. Cummulative Multiplexor Channel Transfer Rates

Mode	Command Block	Word Transfers	Byte Transfers
Byte/ Word	short	275 KBytes/sec	138 KBytes/sec
Multiplex	long	240 KBytes/sec	120 KBytes/sec
Block Multiplex	short	4 MBytes/sec	2 MBytes/sec
	long	4 MBytes/sec	2 MBytes/sec

Data Chaining

The transfer rate for data chaining depends on the block length of each chained data block, the number of blocks in the chain and also the type of chaining that is being done. See the section on data chaining latencies.

LATENCIES

The latency calculations do not take into account set up, hold and output delay times which are specified in the A.C. Characteristics section. These should be added to get the final latency figures. All timings are in units of T-states (125 ns in an 8 MHz system). If bus cycles are involved then the following abbreviations are used:

T = time for one bus transfer

W = wait time during bus cycles for a slow device

In case of various influences affecting the timing, the most typical case is mentioned in the table and explained in notes.

Assumptions:

1. The channel for which latencies are calculated currently has the highest priority and will not be blocked by other still higher priority requests.
2. In remote mode delays due to CPU accesses to the 82258 are not taken into account for latencies.
3. All control space accesses are on a 16 bit bus and command blocks and data chain lists are addressed on even boundaries.
4. Organizational and other unsynchronized transfers (e.g. prefetch) have been completed before the processing of DREQ starts.

DMA Request Processing:

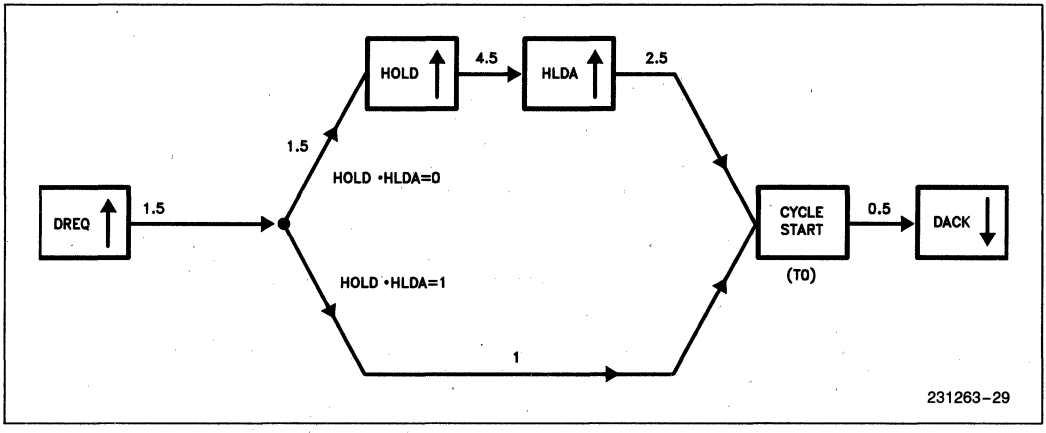


Figure 32. DREQ to DACK Latency in Local Mode*

Table 10. DREQ to DACK in Local Mode*

	Minimum	Typical	Maximum
DREQ to HOLD	2.5	3	3 + W (1) (2)
HOLD to HLDA	1	4.5	(3)
HLDA to CYCLE START	1.5	2.5	2.5
DREQ to CYCLE START (without bus arbitration)	2	2.5	4 + W (1)
CYCLE START to DACK	0.5	0.5	0.5

Notes are indicated in parenthesis

*All timings are in units of T-states (125 ns in an 8 MHz system). If bus cycles are involved then the following abbreviations are used:
T = Time for one bus transfer
W = Wait time during bus cycles for a slow device

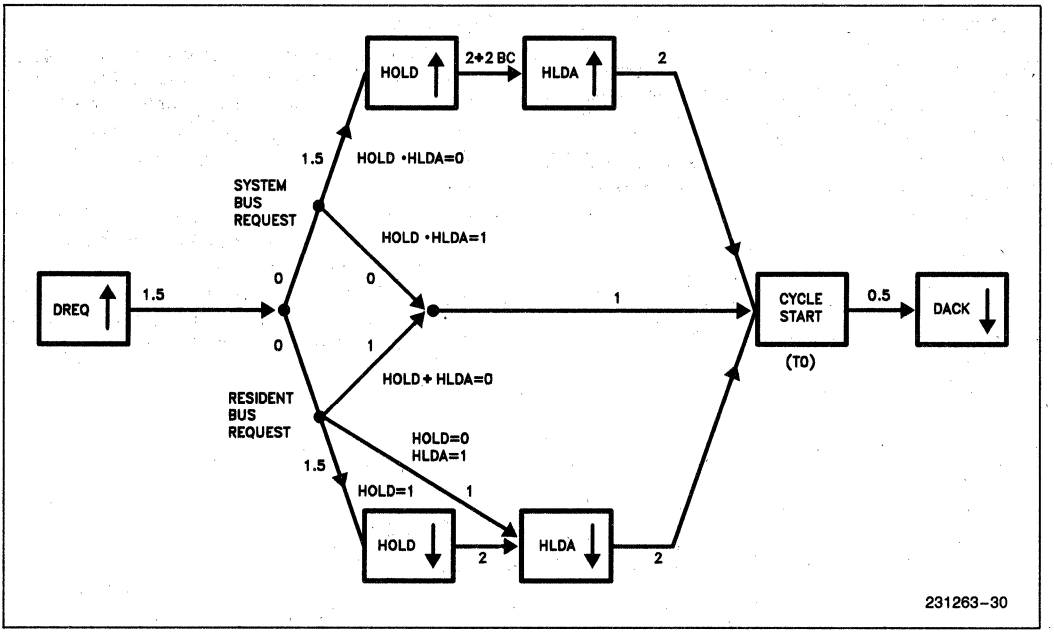


Figure 33. DREQ to DACK Latency in Remote Mode*

Table 11. DREQ to DACK in Remote Mode*

	Minimum	Typical	Maximum
DREQ to HOLDset	2.5	3	3 + W (1) (2)
HOLDset to HLDAset	2 BC	2 + 2 BC	(4)
HOLDAset TO CYCLE START	1.5	2	2.5
DREQ to HOLDreset	1.5	3	5.5 + W (1)
HOLDreset to HLDAreset	1	2	2
HLDAreset to CYCLE START	1.5	2	2.5
DREQ to CYCLE START (without bus change)	2	3.5	5 + W (1)
CYCLE START to DACK	0.5	0.5	0.5

Notes:
(1) Single bus cycle running: 1 + W
unseparable bus cycles running:
—word access at odd addresses (and pointer transfers): 3 + 2W
—IOACK cycle (only multiplexor channel): 7 + 2W
(2) General Burst Counter = 0: 2 × GDR
HLDA = 1, HOLD = 0: Wait for HLDA = 0
HLDA lost: 2
(3) 16 + 15W (from the 286 manual, assumed repeat and lock prefix not combined)
(4) Bus arbitration + currently running bus transfers.
BC = Multibus clock cycle.
* All timings are in units of T-states (125 ns in an 8 MHz system).
If bus cycles are involved then the following abbreviations are used:
T = Time for one bus transfer
W = Wait time during bus cycles for a slow device

General Command Processing:*

	Minimum	Typical	Maximum
WRITE to Set Up	6.5	8	9.5
	+ HOLD/HOLDA sequence		

At this point the start command is ready for the start of the channel set up routine

Set Up Processing:*

Standard command block	: 7T + 4
additional for long command block	: 5T
additional for list data chaining	: 1T + 2
additional for linked list data chaining	: 3T + 2

Type 1 Command Processing:*

Chaining : same as the set up processing

Termination :

store CSR and calculate next command pointer	: 1T + 6
store status block (if programmed)	: 6T

Type 2 Command Processing:*

Standard :

CCR load	: 1T
CCR decode and execution	: 2T + 2
additional for jump	: 4

START/STOP Subchannel:*

(see General Command Processing for set up)

Execution : 4T + 6

Multiplexor Channel:*

(see General Command Processing for set up)

IOREQ to IOACK : identical to DREQ to DACK timing

First IOACK to second IOACK	: 1T + 2
Second IOACK to vector in LVR	: 1T + 2
Calculate MT address and read command pointer into CPR	: 2T + 4
Data transfer	: 2T + 2
Restore pointers	: 4T + 4
Restore byte count	: 2T

Data Chaining:*

Latencies in data chaining occur when transfers are changed between data blocks.

List Chaining	: 3T + 6
Linked List Chaining	: 5T + 6

* All timings are in units of T-states (125 ns in an 8 MHz system).

If bus cycles are involved then the following abbreviations are used:

T = Time for one bus transfer
W = Wait time during bus cycles for a slow device

Absolute Maximum Ratings

Ambient Temperature Under Bias	0°C to 55°C
Case Temperature	0°C to 85°C
Storage Temperature	−65°C to +150°C
Voltage on Any Pin with Respect to Ground	−1.0V to +7V
Power Dissipation	3.6 Watt

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. Characteristics $V_{CC}=5V \pm 5\%$; $T_A=0°C$ to $+55°C$, or $T_{CASE}=0°C$ to $+85°C$

Symbol	Parameter	Limit Values		Units	Test Conditions
		Min	Max		
V_{IL}	Input Low Voltage (except CLK)	−0.5	+0.8	V	—
V_{IH}	Input High Voltage (except CLK)	2.0	V_{CC} + 0.5		
V_{OL}	Output Low Voltage	—	0.45		I_{OL} = 3.00 mA
V_{OH}	Output High Voltage	2.4	—		I_{OH} = −400 μA
I_{CC}	Power Supply Current	—	475 370	mA	T_A = 0°C, T_A = 55° all outputs open
I_{LI}	Input Leakage Current		±10	μA	$0V \le V_{IN} \le V_{CC}$
I_{LO}	Output Leakage Current		−200	μA	$0.45V \le V_{OUT} = V_{CC}$
	$\overline{S0}$, $\overline{S1}$, $\overline{S2}$, $\overline{BHE}$, $\overline{RD}$, $\overline{WR}$, M/$\overline{IO}$				
	HOLD (RQ/GT mode), $\overline{EOD}$		−1.5	mA	
	other pins		± 10	μA	
V_{CL}	Clock Input Low Voltage	−0.5	+0.6	V	—
V_{CH}	Clock Input High Voltage	3.8	V_{CC}+1.0		
C_{IN}	Capacitance of Inputs (except CLK)	—	10	pF	fc = 1 MHz
C_O	Capacitance of I/O or Outputs		20		
CCLK	Capacitance of CLK Input		12		

A.C. Characteristics $V_{CC} = 5V \pm 5\%$; $T_A = 0°C$ to $+55°C$, or $T_{CASE} = 0°C$ to $+85°C$

AC timings are referenced to 0.8V and 2.0V points of signals as illustrated in datasheet waveforms, unless otherwise noted.

Sym	Parameter	6 MHz		8 MHz		Unit	Test Conditions
		Min	Max	Min	Max		
1	CLK Cycle Period (286 Mode)	83	250	62	250	ns	
2	CLK Low Time (286 Mode)	20	225	15	230	ns	at 1.0V
3	CLK High Time (286 Mode)	25	230	20	235	ns	at 3.6V
4	Output Valid Delay	1 –	80	1 –	60	ns	CL = 125 pF
5	Output Valid Delay	1 –	55	1 –	40	ns	CL = 125 pF
6	Data Setup Time	15		10		ns	
6a	Address Input Setup (186 Mode)	20		15		ns	
7	Data Hold Time	8		5		ns	
8	READY Setup Time	50		38		ns	
9	READY Hold Time	35		25		ns	
10	Input Setup Time	25		20		ns	
10a	Status Setup Time (186 Mode)	30		30		ns	
11	Input Hold Time	25		20		ns	
11a	BHE Hold Time (186 Mode)	15		10		ns	
12	Address Setup Time	3		2		ns	
13	Data Valid Delay	0	60	0	50	ns	
14	Data Float Delay	8	80	5	60	ns	
15	Chip Select Setup	30		20		ns	
16	Command Length	320		290		ns	
17	Data Setup Time	185		165		ns	
18	Address Setup Time	30		20		ns	
19	Command Inactive	320		290		ns	
19a	Access Time		420		380	ns	
20	CLK Period (186 Mode)	166	500	125	500	ns	
21	CLK Low Time (186 Mode)	76		55		ns	
22	CLK High Time (186 Mode)	76		55		ns	
23	CLK Rise Time (186 Mode)		15		15	ns	
24	CLK Fall Time (186 Mode)		15		15	ns	
25	READY Active Setup Time	20		20		ns	
26	READY Hold Time	10		10		ns	
26a	SREADY Hold Time (186 Mode)	15		15		ns	
27	READY Inactive Setup Time	35		35		ns	
28	Control Reset Setup Time	25		20		ns	
29	Control Reset Hold Time	0		0		ns	
30	Address/Data Valid Delay	10	55	10	50	ns	
31	Status Delay	10	75	10	55	ns	
32	Address/Data Float Delay	10	50	10	50	ns	
33	DT/$\overline{R}$ Delay (186 Mode)	10	76	10	55	ns	
34	DEN Delay (186 Mode)	10	80	10	60	ns	

A.C. MEASUREMENT POINT DESCRIPTION

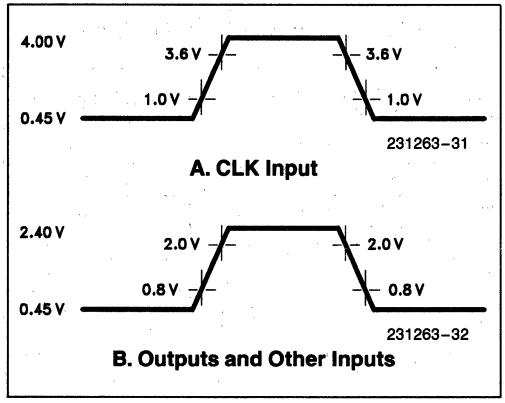


Figure 33a. AC Drive and Measurement Points

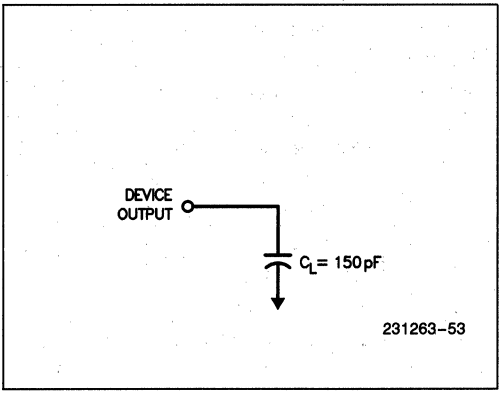


Figure 33b. AC Test Loading on Outputs

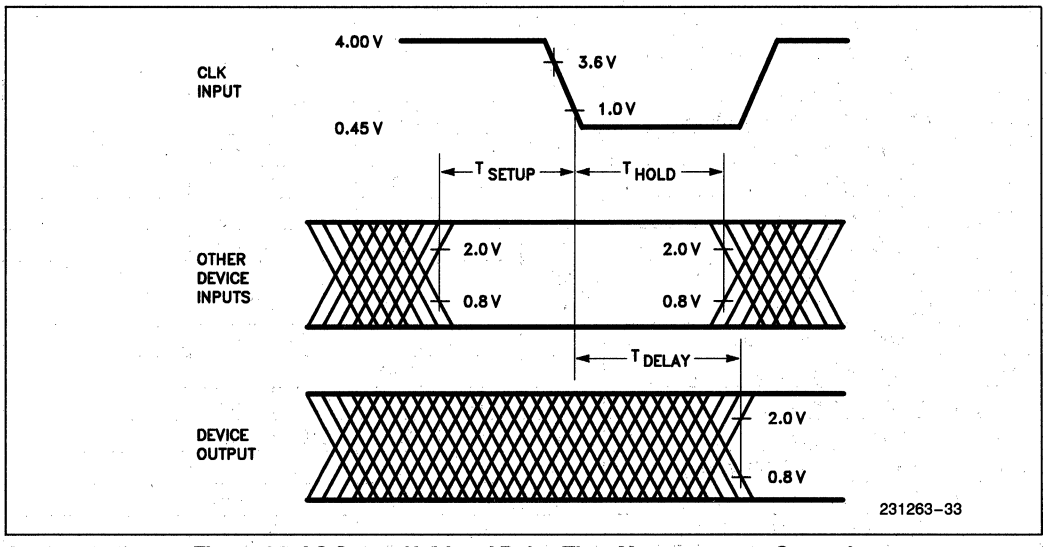


Figure 34. AC Setup, Hold and Delay Time Measurement - General

BUS CYCLE T-STATES:

The bus cycles are subdivided into T-states which are interpreted differently depending on whether the 82258 is in the 286 mode or the 186 mode.

286 Mode T-states: Each T-state is two clock cycles long and starts in the middle of a processor cycle and ends in the middle of the succeeding processor cycle.

TI: [The bus is idle] This state will occur if the 82258 cannot start the next bus cycle.

T0: [A new bus cycle is beginning] When the address and status of a new bus cycle is to be sent as output, this state is used.

T1: [A bus cycle is proceeding] This state is used to allow the bus controller commands to become active and, to output data during a write cycle.

T2I: [A bus cycle is prepared for termination with no new cycle ready to begin] If the $\overline{\text{READY}}$ signal is active and no new bus cycle is ready to begin, this will be the state used. Input data will be accepted during this state if the $\overline{\text{READY}}$ signal is active and if the bus cycle is an input cycle.

T20: [A bus cycle is prepared for termination with a new cycle ready to begin] This state terminates a bus cycle if the $\overline{\text{READY}}$ signal is active and if a new bus cycle is ready to

begin. As with the T2I state, input data will be accepted during this state if the cycle is an input cycle and if the READY signal is active. This state will also output the address of the new bus cycle, and if READY is active, the status also.

186 Mode T-states: The T-states are one CLK period long, beginning and ending with the falling edge of the CLK signal.

TI: [The bus is idle] This state occurs if the 82258 cannot start the following bus cycle.

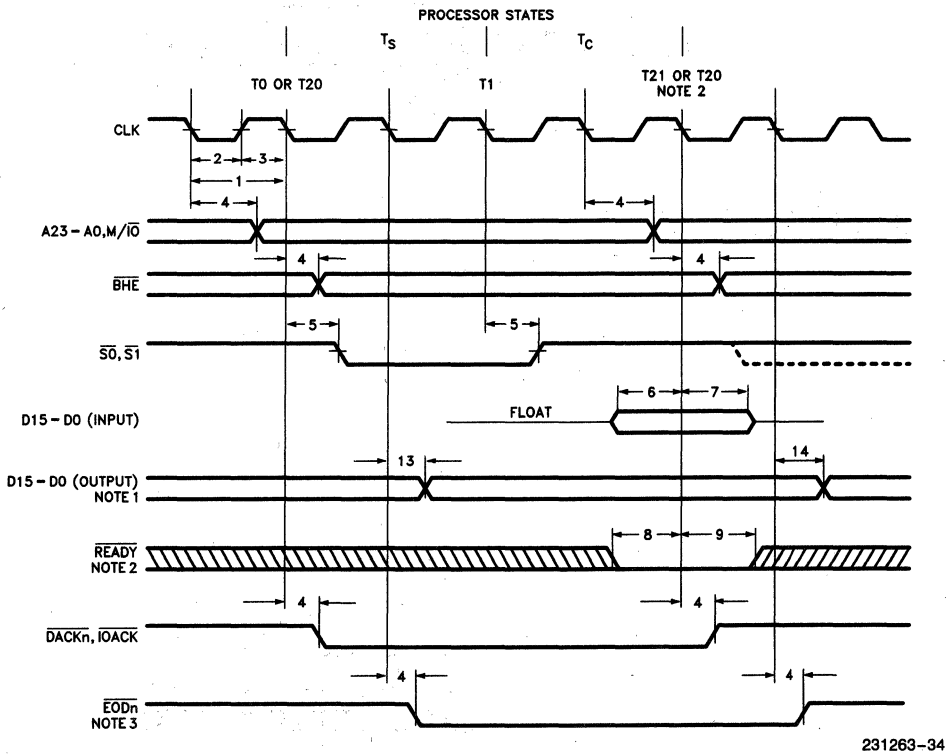
T1: [The first bus cycle T-state] During this state, address information is output to the A19/S6–A16/S3 and AD15–AD0 pins. The status is activated with the rising edge of the CLK previous to this state.

T2: [The second bus cycle T-state] This state allows the bus controller and the 82258 commands to become active and outputs data if the cycle is a write cycle.

T3: [The third bus cycle T-state] This state is used to synchronize the ready signals. If the bus is not ready, then the bus cycle is extended by repeating this state, with the status lines going inactive during the last T3-state.

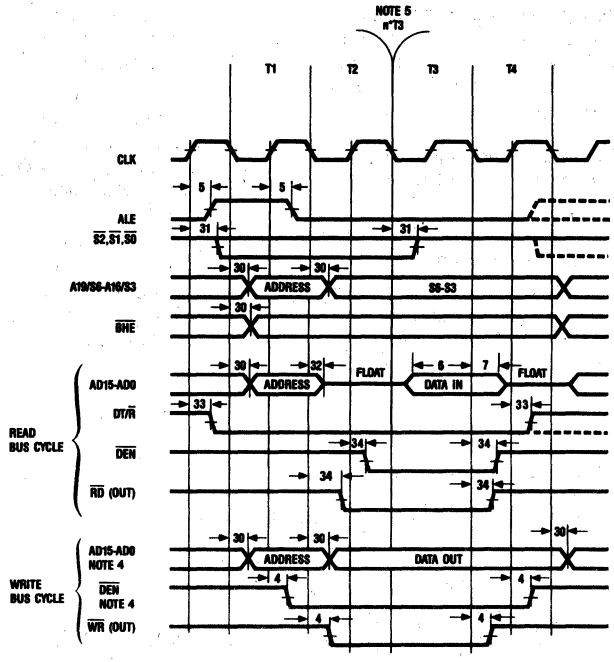
T4: [The last bus cycle T-state] During this cycle, data is input for input cycles and the bus controller and the 82258 commands are disactivated. If the following state is T1, then the status is activated during this state.

Waveforms



NOTES:
1. D15–D0 floats during Single Cycle Transfer like a Read Cycle.
2. T2 will be repeated, if READY is inactive.
3. Initiated by terminal count.

Figure 35. Timing of an Active Bus Cycle (286 and Remote Modes)



NOTES:
4. For a Single Cycle Transfer the timings of AD15–AD0, $\overline{DEN}$ and DT/$\overline{R}$ are the same as in a Read Bus Cycle.
5. Additional T3 cycles will be inserted if bus is not ready (see Figure 40).

Figure 36. Timing of an Active Bus Cycle (186 and 8086 Modes)

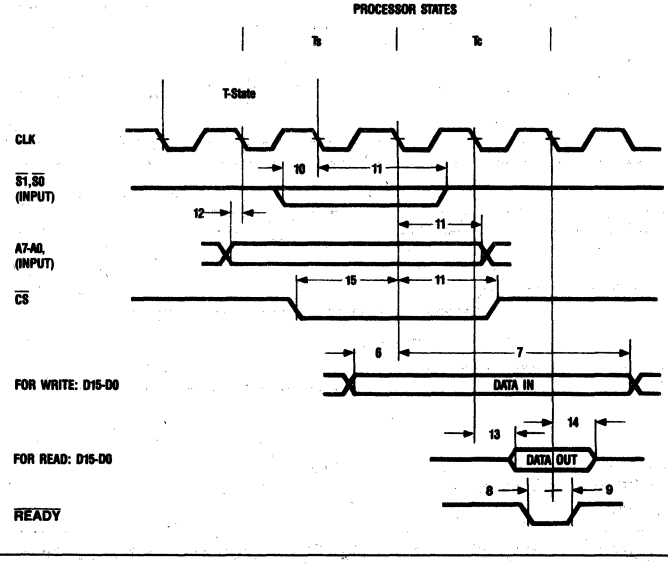


Figure 37. Timing of a Synchronous Access to the 82258 (286 Mode)

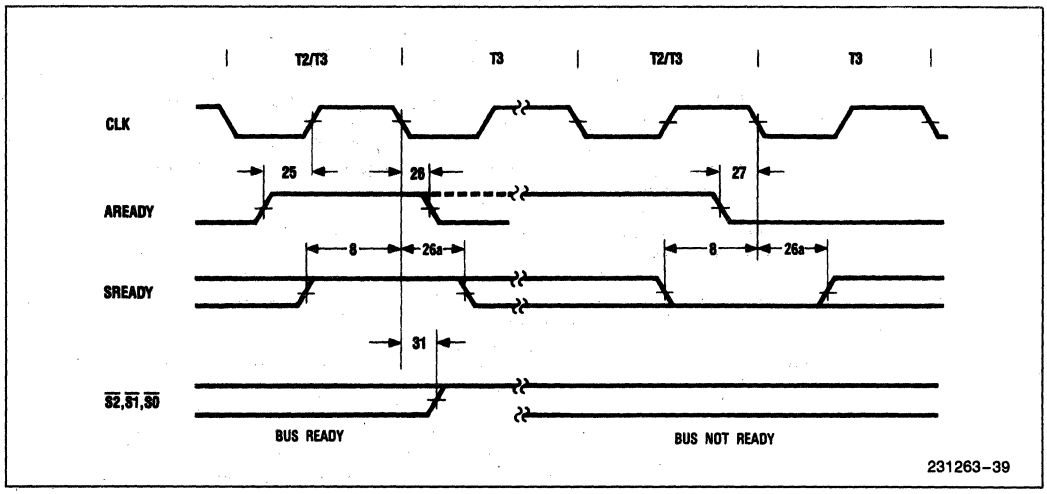
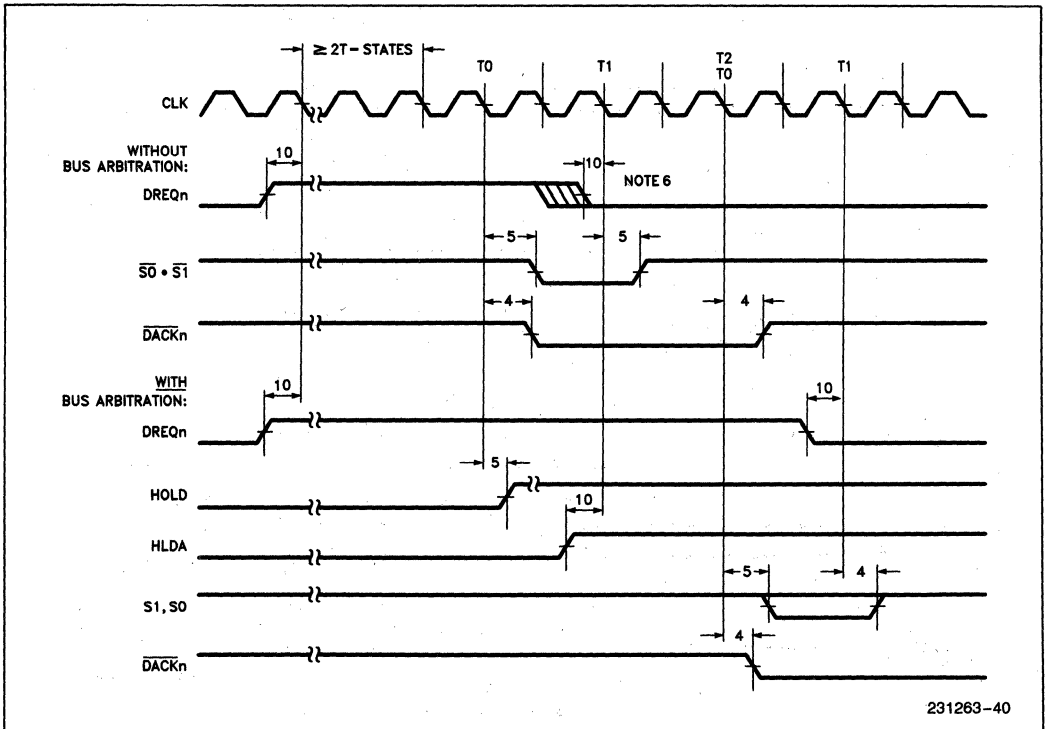


Figure 40. READY Timing (186 Mode)



NOTE:
6. The trailing edge of DREQn, as specified in this diagram, is necessary if only one bus cycle should be executed. A later trailing edge may cause an additional bus cycle (continuous DREQ), if no READY-wait-states are inserted.

Figure 41. DREQ, $\overline{\text{DACK}}$ Timing (286 and Remote Modes)

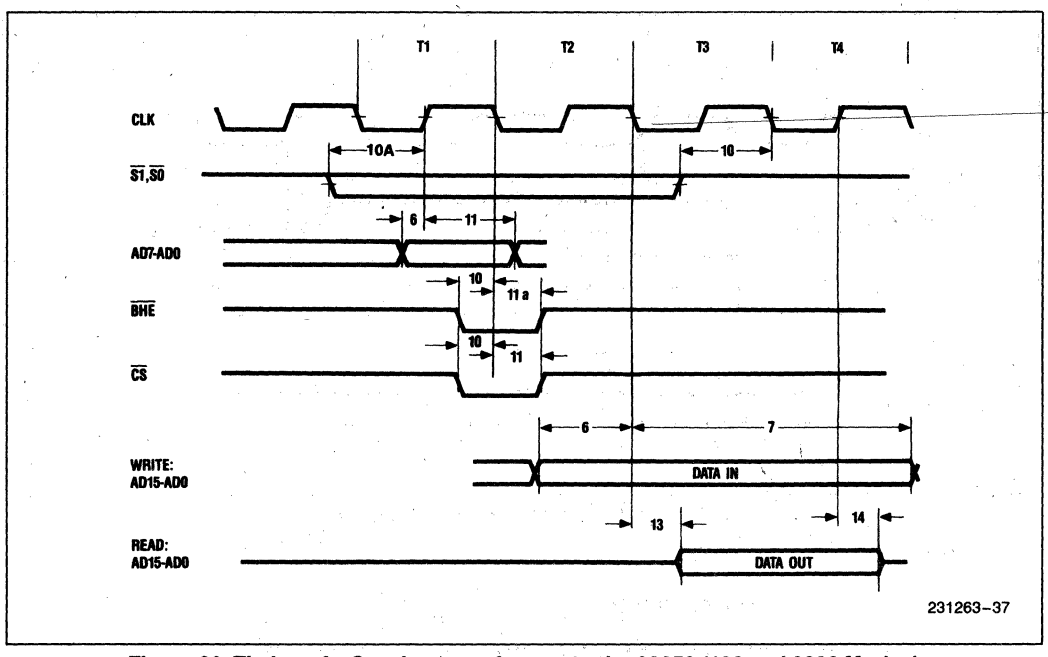


Figure 38. Timing of a Synchronous Access to the 82258 (186 and 8086 Modes)

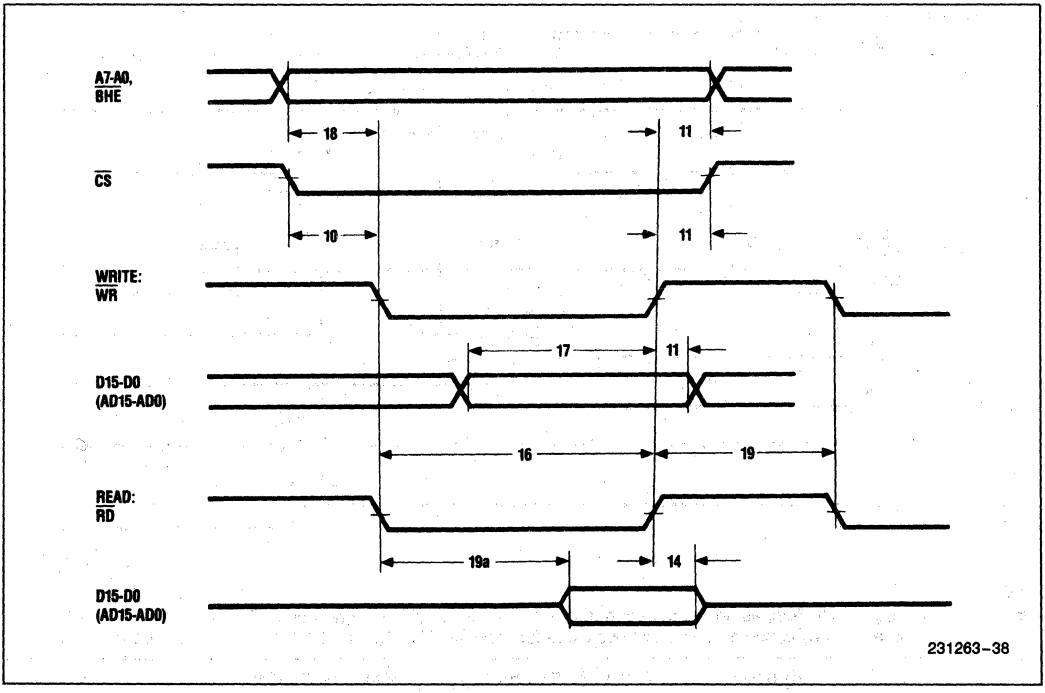
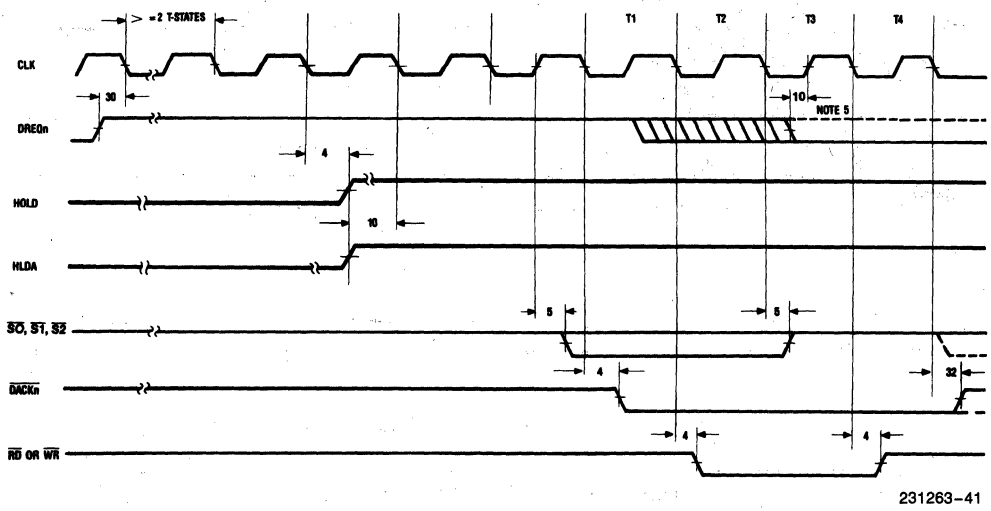


Figure 39. Timing of an Asynchronous Access to the 82258 (All Modes)



NOTE:
The DREQ and $\overline{\text{DACK}}$ signal timings are the same for the 8086 mode.

Figure 42. DREQ, $\overline{\text{DACK}}$ Timing (186 Mode)

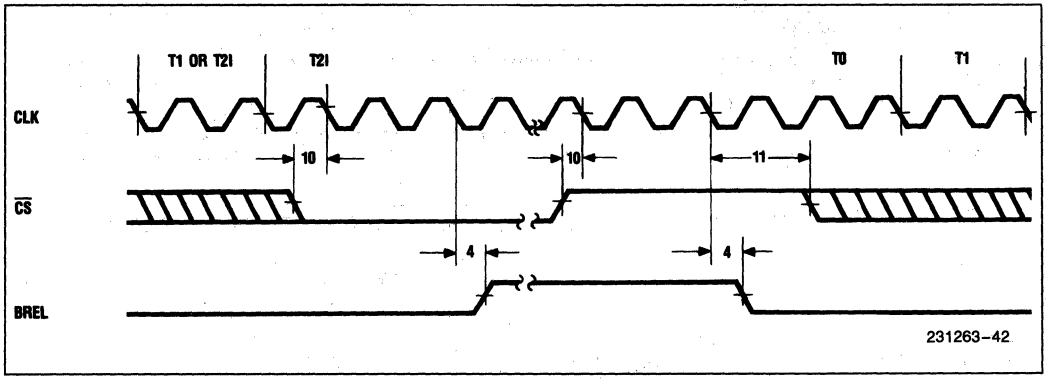


Figure 43. BREL, Bus Tristate Timing (Remote Mode)

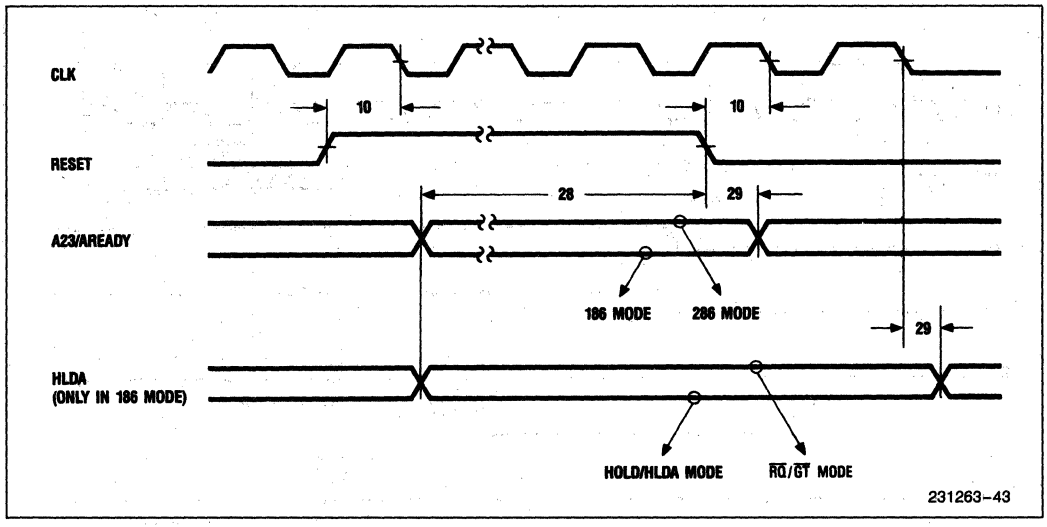
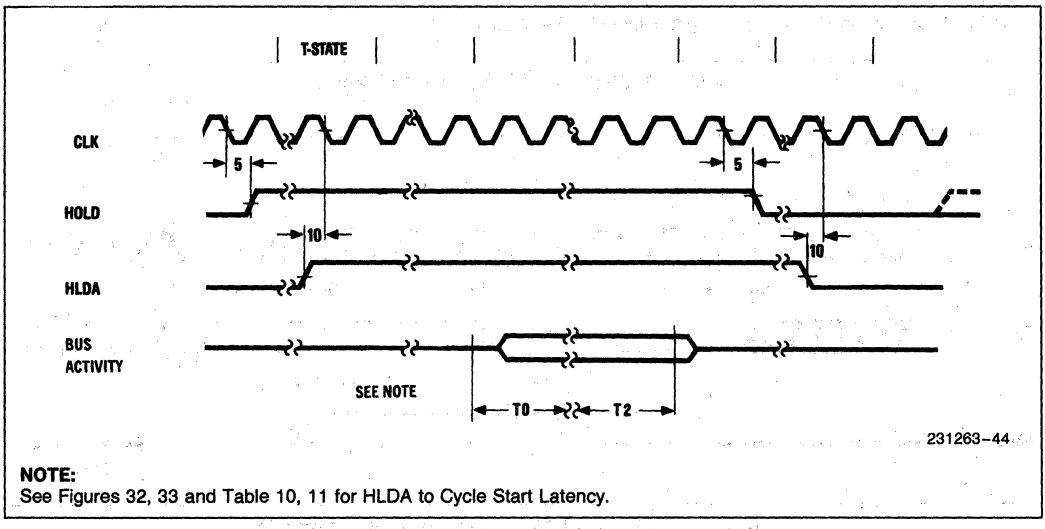
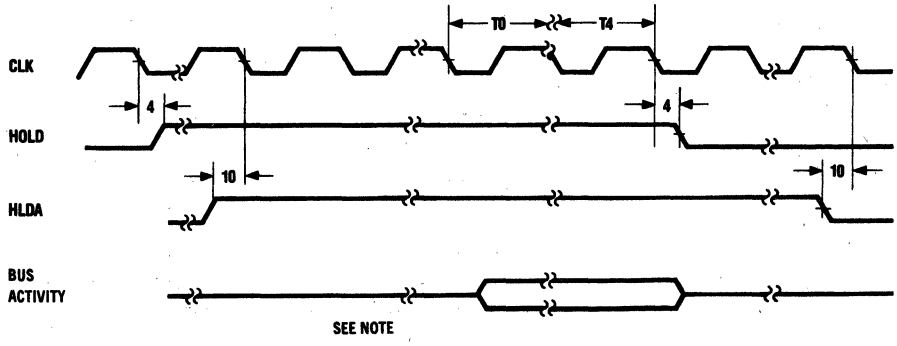


Figure 44. RESET Timing (All Modes)



NOTE:
See Figures 32, 33 and Table 10, 11 for HLDA to Cycle Start Latency.

Figure 45. HOLD, HLDA Timing (286 and Remote Modes)



NOTE:
See Figure 32 and Table 10 for HLDA to Cycle Start Latency.

Figure 46. HOLD, HLDA Timing (186 Mode)

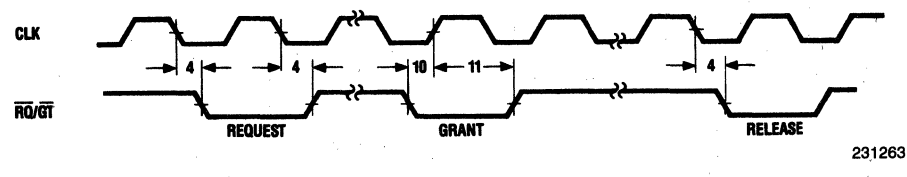


Figure 47. $\overline{RQ}/\overline{GT}$ Timing (8086 Mode)

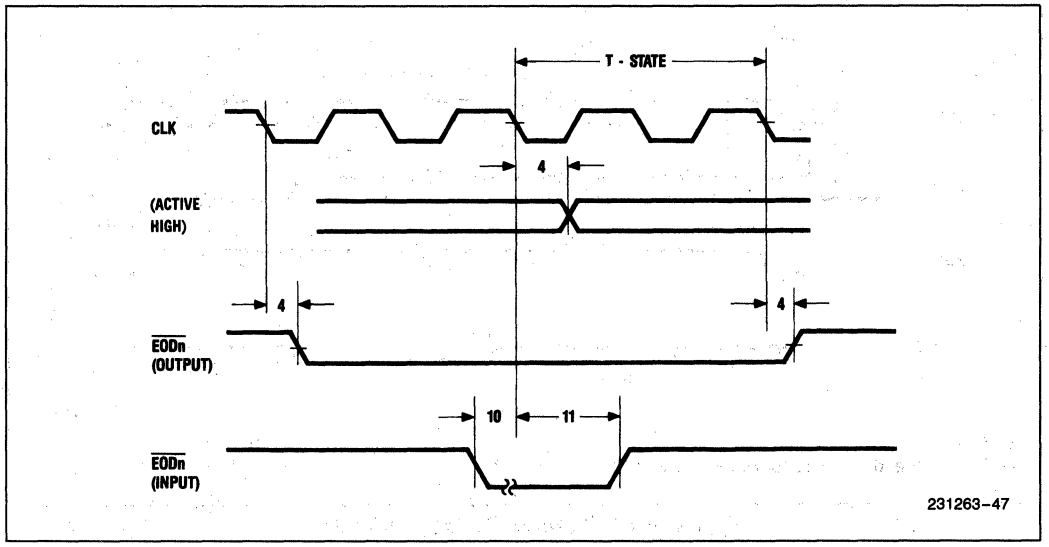


Figure 48. INTOUT, $\overline{EOD}$ Timing (286 and Remote Modes)

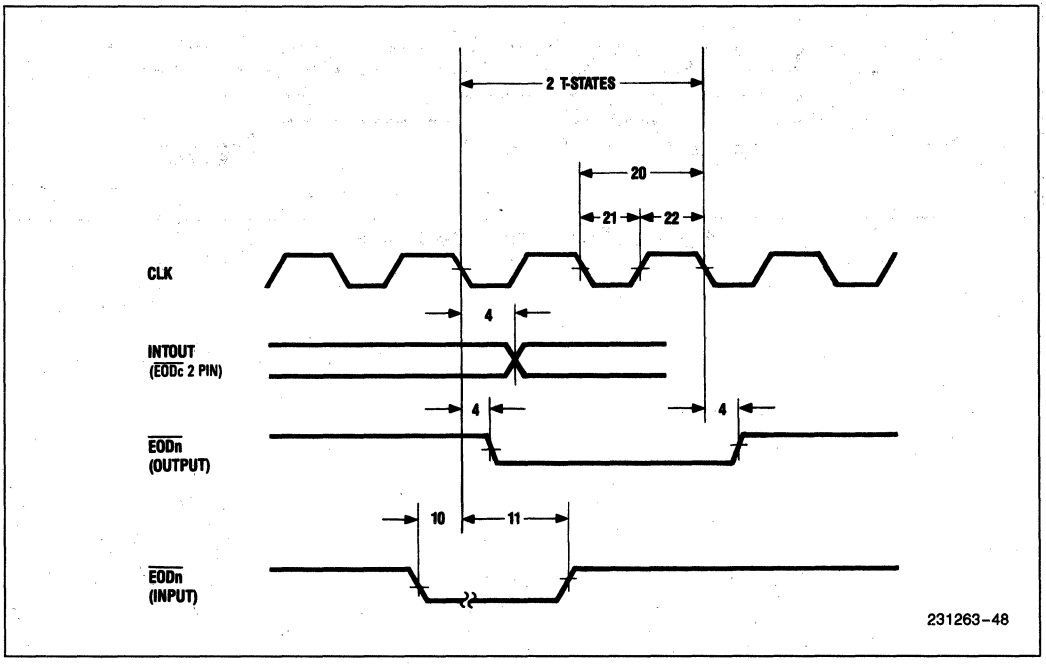


Figure 49. INTOUT, $\overline{EOD}$ Timing (186 and 8086 Modes)

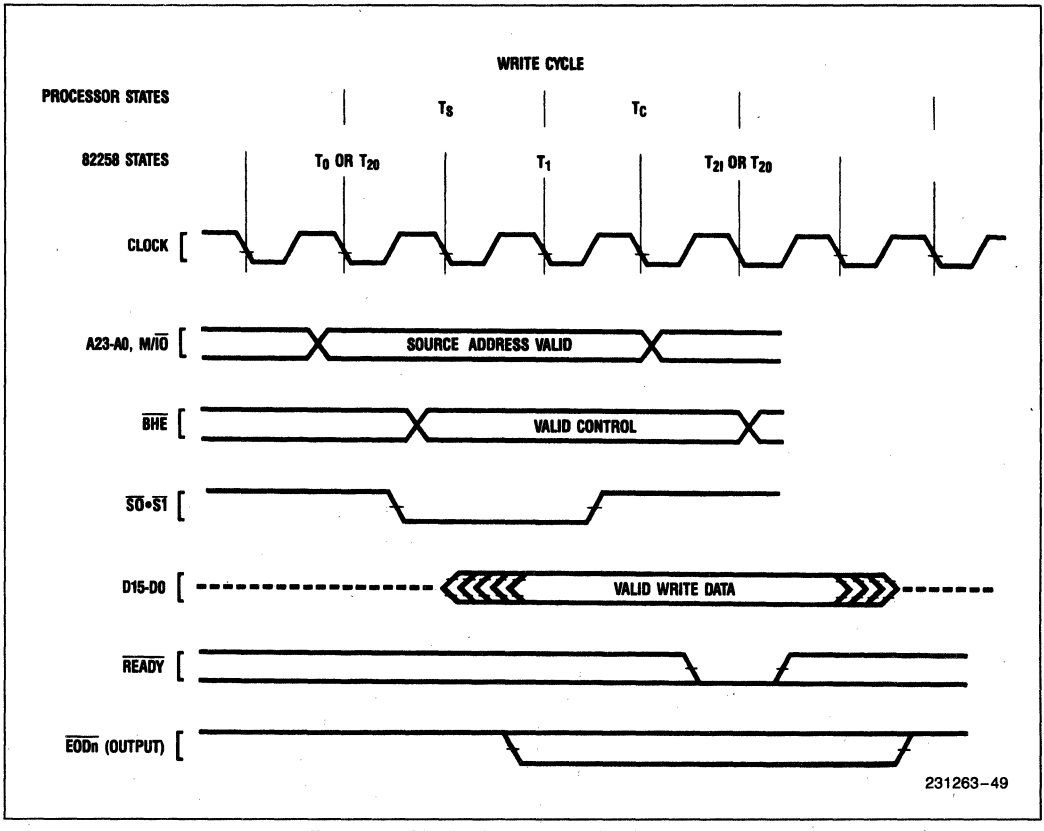


Figure 50. Single Cycle Transfer (286 Mode)

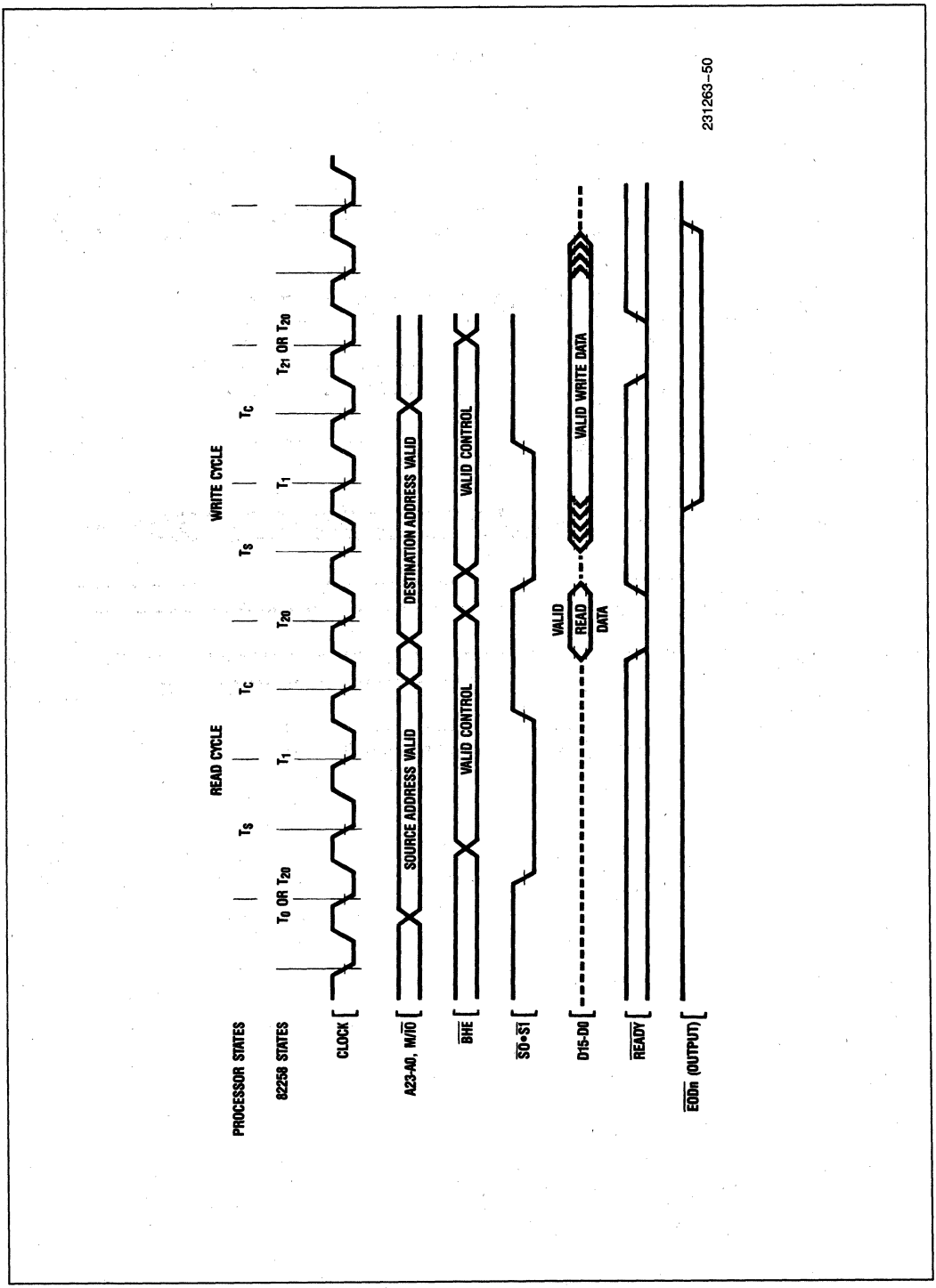
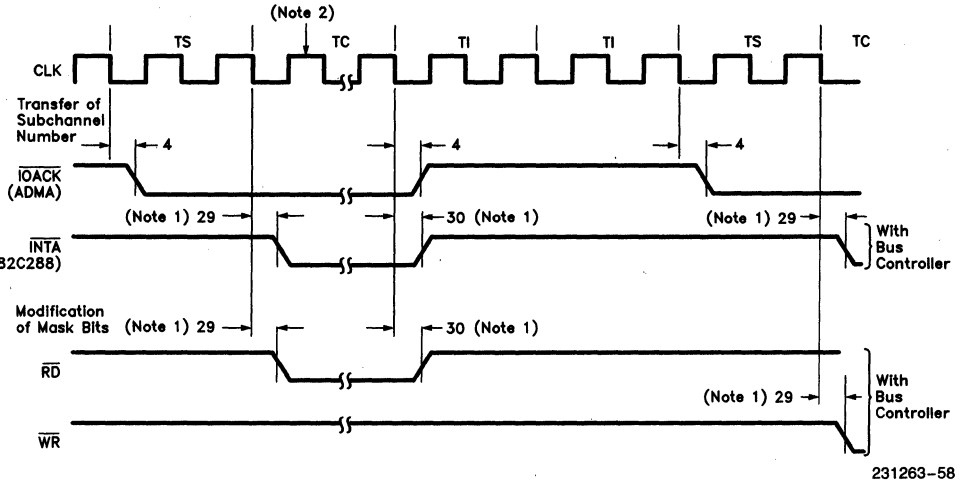
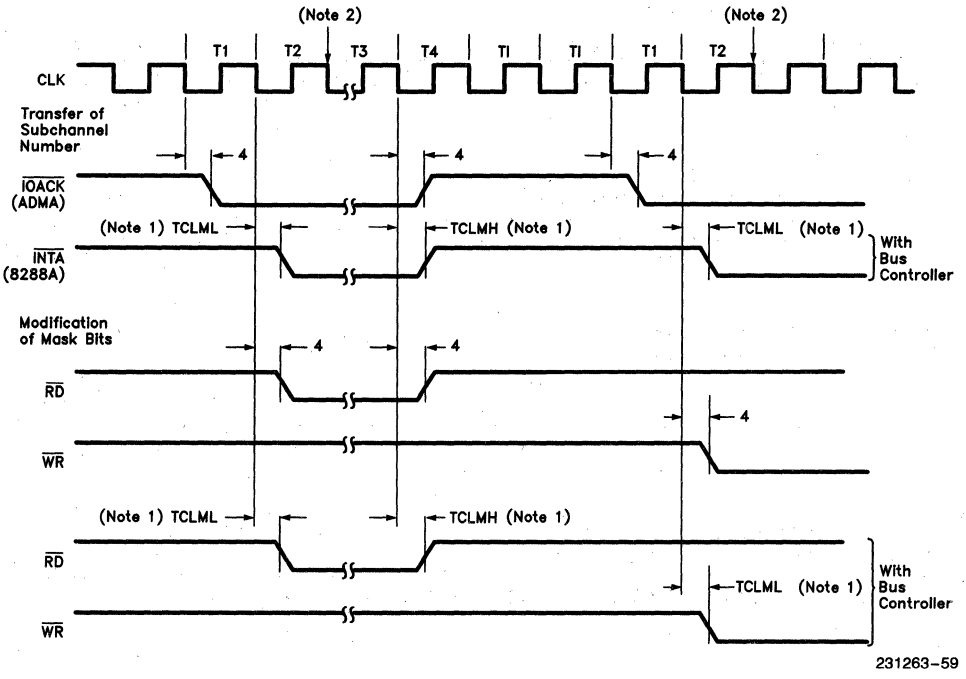


Figure 51. Two Cycle Transfer (286 Mode)



NOTES:
1. These timings are 82C288 timings.
2. Additional wait states may be inserted.

Figure 52. Access to 8259A in 80286 and Remote Modes.



NOTES:
1. These timings are 8288A timings.
2. Additional wait states may be inserted.

Figure 53. Access to 8259A in 8086 and 80186 Modes

DATA SHEET REVISION REVIEW

The following list represents key differences between this and the -003 82258 data sheet. Please review this summary carefully.

1. Figure 35 was updated. The new timing diagram now illustrates DACKN#, IOACK#, and EODn# timings during active bus cycles in the 80286 and remote modes.
2. Figure 37 was updated. The new timing diagram now illustrates the READY# signal during a synchronous access to the 82258.
3. Figure 41 was updated. The new timing diagram completely separates the DREQ, DACK# timings from "without bus arbitration" and "with bus arbitration".
4. Two new timing illustrations were added to the 82258 data sheet. Figure 52 illustrates bus accesses to the 8259A in 80286 and remote modes, and Figure 53 illustrates bus accesses to the 8259A in 80186 and 8086 modes.
5. A note to the DREQ pin description was added to advise designers to leave unused DREQn inputs left floating.

82288
BUS CONTROLLER FOR 80286 PROCESSORS (82288-12, 82288-10, 82288-8)

- **Provides Commands and Controls for Local and System Bus**
- **Wide Flexibility in System Configurations**
- **Flexible Command Timing**
- **Optional MULTIBUS® Compatible Timing**
- **Single +5V Supply**
- **Available in 20 Pin Cerdip Package**
  (See Packaging Spec, Order #231369)

The Intel 82288 Bus Controller is a 20-pin HMOS component for use in 80286 microsystems. The bus controller provides command and control outputs with flexible timing options. Separate command outputs are used for memory and I/O devices. The data bus is controlled with separate data enable and direction control signals.

Two modes of operation are possible via a strapping option: MULTIBUS® I compatible bus cycles, and high speed bus cycles.

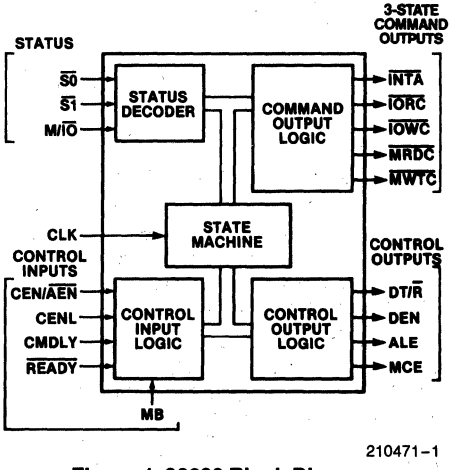


Figure 1. 82288 Block Diagram

20 Pin Cerdip Package

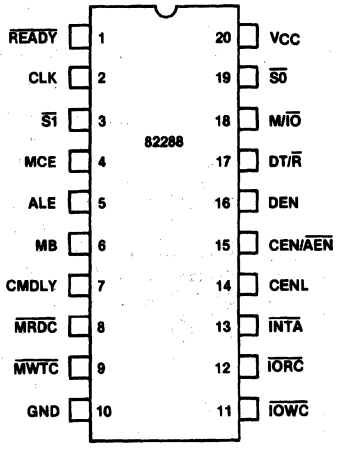


Figure 2. 82288 Pin Configuration

November 1988
Order Number: 210471-007

Table 1. Pin Description

The following pin function descriptions are for the 82288 bus controller.

Symbol	Type	Name and Function
CLK	I	**SYSTEM CLOCK** provides the basic timing control for the 82288 in an 80286 microsystem. Its frequency is twice the internal processor clock frequency. The falling edge of this input signal establishes when inputs are sampled and command and control outputs change.
$\overline{S0}$, $\overline{S1}$	I	**BUS CYCLE STATUS** starts a bus cycle and, along with M/$\overline{IO}$, defines the type of bus cycle. These inputs are active LOW. A bus cycle is started when either $\overline{S1}$ or $\overline{S0}$ is sampled LOW at the falling edge of CLK. Setup and hold times must be met for proper operation.

80286 Bus Cycle Status Definition

M/$\overline{IO}$	$\overline{S1}$	$\overline{S0}$	Type of Bus Cycle
0	0	0	Interrupt Acknowledge
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	None; Idle
1	0	0	Halt or Shutdown
1	0	1	Memory Read
1	1	0	Memory Write
1	1	1	None; Idle

M/$\overline{IO}$	I	**MEMORY OR I/O SELECT** determines whether the current bus cycle is in the memory space or I/O space. When LOW, the current bus cycle is in the I/O space. Setup and hold times must be met for proper operation.
MB	I	**MULTIBUS MODE SELECT** determines timing of the command and control outputs. When HIGH, the bus controller operates with MULTIBUS I compatible timings. When LOW, the bus controller optimizes the command and control output timing for short bus cycles. The function of the CEN/$\overline{AEN}$ input pin is selected by this signal. This input is typically a strapping option and not dynamically changed.
CENL	I	**COMMAND ENABLE LATCHED** is a bus controller select signal which enables the bus controller to respond to the current bus cycle being initiated. CENL is an active HIGH input latched internally at the end of each T_S cycle. CENL is used to select the appropriate bus controller for each bus cycle in a system where the CPU has more than one bus it can use. This input may be connected to V_{CC} to select this 82288 for all transfers. No control inputs affect CENL. Setup and hold times must be met for proper operation.
CMDLY	I	**COMMAND DELAY** allows delaying the start of a command. CMDLY is an active HIGH input. If sampled HIGH, the command output is not activated and CMDLY is again sampled at the next CLK cycle. When sampled LOW the selected command is enabled. If $\overline{READY}$ is detected LOW before the command output is activated, the 82288 will terminate the bus cycle, even if no command was issued. Setup and hold times must be satisfied for proper operation. This input may be connected to GND if no delays are required before starting a command. This input has no effect on 82288 control outputs.
$\overline{READY}$	I	**READY** indicates the end of the current bus cycle. $\overline{READY}$ is an active LOW input. MULTIBUS I mode requires at least one wait state to allow the command outputs to become active. $\overline{READY}$ must be LOW during reset, to force the 82288 into the idle state. Setup and hold times must be met for proper operation. The 82C284 drives $\overline{READY}$ LOW during RESET.

Table 1. Pin Description (Continued)

Symbol	Type	Name and Function
CEN/$\overline{AEN}$	I	**COMMAND ENABLE/ADDRESS ENABLE** controls the command and DEN outputs of the bus controller. CEN/$\overline{AEN}$ inputs may be asynchronous to CLK. Setup and hold times are given to assure a guaranteed response to synchronous inputs. This input may be connected to V_{CC} or GND. When MB is HIGH this pin has the $\overline{AEN}$ function. $\overline{AEN}$ is an active LOW input which indicates that the CPU has been granted use of a shared bus and the bus contoller command outputs may exit 3-state OFF and become inactive (HIGH). $\overline{AEN}$ HIGH indicates that the CPU does not have control of the shared bus and forces the command outputs into 3-state OFF and DEN inactive (LOW). When MB is LOW this pin has the CEN function. CEN is an unlatched active HIGH input which allows the bus controller to activate its command and DEN outputs. With MB LOW, CEN LOW forces the command and DEN outputs inactive but does not tristate them.
ALE	O	**ADDRESS LATCH ENABLE** controls the address latches used to hold an address stable during a bus cycle. This control output is active HIGH. ALE will not be issued for the halt bus cycle and is not affected by any of the control inputs.
MCE	O	**MASTER CASCADE ENABLE** signals that a cascade address from a master 8259A interrupt controller may be placed onto the CPU address bus for latching by the address latches under ALE control. The CPU's address bus may then be used to broadcast the cascade address to slave interrupt controllers so only one of them will respond to the interrupt acknowledge cycle. This control output is active HIGH. MCE is only active during interrupt acknowledge cycles and is not affected by any control input. Using MCE to enable cascade address drivers requires latches which save the cascade address on the falling edge of ALE.
DEN	O	**DATA ENABLE** controls when data transceivers connected to the local data bus should be enabled. DEN is an active HIGH control output. DEN is delayed for write cycles in the MULTIBUS I mode.
DT/$\overline{R}$	O	**DATA TRANSMIT/RECEIVE** establishes the direction of data flow to or from the local data bus. When HIGH, this control output indicates that a write bus cycle is being performed. A LOW indicates a read bus cycle. DEN is always inactive when DT/$\overline{R}$ changes states. This output is HIGH when no bus cycle is active. DT/$\overline{R}$ is not affected by any of the control inputs.
$\overline{IOWC}$	O	**I/O WRITE COMMAND** instructs an I/O device to read the data on the data bus. This command output is active LOW. The MB and CMDLY inputs control when this output becomes active. $\overline{READY}$ controls when it becomes inactive.
$\overline{IORC}$	O	**I/O READ COMMAND** instructs an I/O device to place data onto the data bus. This command output is active LOW. The MB and CMDLY inputs control when this output becomes active. $\overline{READY}$ controls when it becomes inactive.
$\overline{MWTC}$	O	**MEMORY WRITE COMMAND** instructs a memory device to read the data on the data bus. This command output is active LOW. The MB and CMDLY inputs control when this output becomes active. $\overline{READY}$ controls when it becomes inactive.
$\overline{MRDC}$	O	**MEMORY READ COMMAND** instructs the memory device to place data onto the data bus. This command output is active LOW. The MB and CMDLY inputs control when this output becomes active. $\overline{READY}$ controls when it becomes inactive.

Table 1. Pin Description (Continued)

Symbol	Type	Name and Function
$\overline{INTA}$	O	**INTERRUPT ACKNOWLEDGE** tells an interrupting device that its interrupt request is being acknowledged. This command output is active LOW. The MB and CMDLY inputs control when this output becomes active. $\overline{READY}$ controls when it becomes inactive.
V_{CC}		**System Power:** +5V Power Supply
GND		**System Ground:** 0V

Table 2. Command and Control Outputs for Each Type of Bus Cycle

Type of Bus Cycle	M/$\overline{IO}$	$\overline{S1}$	$\overline{S0}$	Command Activated	DT/$\overline{R}$ State	ALE, DEN Issued?	MCE Issued?
Interrupt Acknowledge	0	0	0	$\overline{INTA}$	LOW	YES	YES
I/O Read	0	0	1	$\overline{IORC}$	LOW	YES	NO
I/O Write	0	1	0	$\overline{IOWC}$	HIGH	YES	NO
None; Idle	0	1	1	None	HIGH	NO	NO
Halt/Shutdown	1	0	0	None	HIGH	NO	NO
Memory Read	1	0	1	$\overline{MRDC}$	LOW	YES	NO
Memory Write	1	1	0	$\overline{MWTC}$	HIGH	YES	NO
None; Idle	1	1	1	None	HIGH	NO	NO

Operating Modes

Two types of buses are supported by the 82288: MULTIBUS I and non-MULTIBUS I. When the MB input is strapped HIGH, MULTIBUS I timing is used. In MULTIBUS I mode, the 82288 delays command and data activation to meet IEEE-796 requirements on address to command active and write data to command active setup timing. MULTIBUS I mode requires at least one wait state in the bus cycle since the command outputs are delayed. The non-MULTIBUS I mode does not delay any outputs and does not require wait states. The MB input affects the timing of the command and DEN outputs.

Command and Control Outputs

The type of bus cycle performed by the local bus master is encoded in the M/$\overline{IO}$, $\overline{S1}$, and $\overline{S0}$ inputs. Different command and control outputs are activated depending on the type of bus cycle. Table 2 indicates the cycle decode done by the 82288 and the effect on command, DT/$\overline{R}$, ALE, DEN, and MCE outputs.

Bus cycles come in three forms: read, write, and halt. Read bus cycles include memory read, I/O read, and interrupt acknowledge. The timing of the associated read command outputs ($\overline{MRDC}$, $\overline{IORC}$, and $\overline{INTA}$), control outputs (ALE, DEN, DT/$\overline{R}$) and control inputs (CEN/$\overline{AEN}$, CENL, CMDLY, MB, and $\overline{READY}$) are identical for all read bus cycles. Read cycles differ only in which command output is activated. The MCE control output is only asserted during interrupt acknowledge cycles.

Write bus cycles activate different control and command outputs with different timing than read bus cycles. Memory write and I/O write are write bus cycles whose timing for command outputs ($\overline{MWTC}$ and $\overline{IOWC}$), control outputs (ALE, DEN, DT/$\overline{R}$) and control inputs (CEN/$\overline{AEN}$, CENL, CMDLY, MB, and $\overline{READY}$) are identical. They differ only in which command output is activated.

Halt bus cycles are different because no command or control output is activated. All control inputs are ignored until the next bus cycle is started via $\overline{S1}$ and $\overline{S0}$.

FUNCTIONAL DESCRIPTION

Introduction

The 82288 bus controller is used in 80286 systems to provide address latch control, data transceiver control, and standard level-type command outputs. The command outputs are timed and have sufficient drive capabilities for large TTL buses and meet all IEEE-796 requirements for MULTIBUS I. A special MULTIBUS I mode is provided to satisfy all address/data setup and hold time requirements. Command timing may be tailored to special needs via a CMDLY input to determine the start of a command and $\overline{\text{READY}}$ to determine the end of a command.

Connection to multiple buses are supported with a latched enable input (CENL). An address decoder can determine which, if any, bus controller should be enabled for the bus cycle. This input is latched to allow an address decoder to take full advantage of the pipelined timing on the 80286 local bus.

Buses shared by several bus controllers are supported. An $\overline{\text{AEN}}$ input prevents the bus controller from driving the shared bus command and data signals except when enabled by an external MULTIBUS I type bus arbiter.

Separate DEN and DT/$\overline{\text{R}}$ outputs control the data transceivers for all buses. Bus contention is eliminated by disabling DEN before changing DT/$\overline{\text{R}}$. The DEN timing allows sufficient time for tristate bus drivers to enter 3-state OFF before enabling other drivers onto the same bus.

The term CPU refers to any 80286 processor or 80286 support component which may become an 80286 local bus master and thereby drive the 82288 status inputs.

Processor Cycle Definition

Any CPU which drives the local bus uses an internal clock which is one half the frequency of the system clock (CLK) (see Figure 3). Knowledge of the phase of the local bus master internal clock is required for proper operation of the 80286 local bus. The local bus master informs the bus controller of its internal clock phase when it asserts the status signals. Status signals are always asserted beginning in Phase 1 of the local bus master's internal clock.

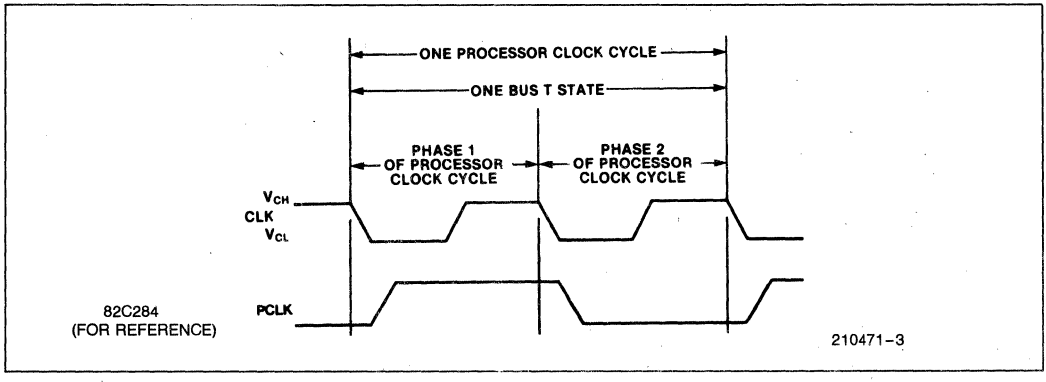


Figure 3. CLK Relationship to the Processor Clock and Bus T-States

Bus State Definition

The 82288 bus controller has three bus states (see Figure 4): Idle (T_I) Status (T_S) and Command (T_C). Each bus state is two CLK cycles long. Bus state phases correspond to the internal CPU processor clock phases.

The T_I bus state occurs when no bus cycle is currently active on the 80286 local bus. This state may be repeated indefinitely. When control of the local bus is being passed between masters, the bus remains in the T_I state.

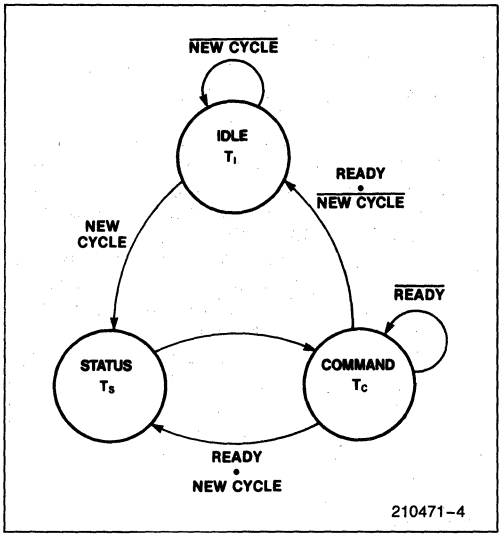


Figure 4. 82288 Bus States

Bus Cycle Definition

The $\overline{S1}$ and $\overline{S0}$ inputs signal the start of a bus cycle. When either input becomes LOW, a bus cycle is started. The T_S bus state is defined to be the two CLK cycles during which either $\overline{S1}$ or $\overline{S0}$ are active (see Figure 5). These inputs are sampled by the 82288 at every falling edge of CLK. When either $\overline{S1}$ or $\overline{S0}$ are sampled LOW, the next CLK cycle is considered the second phase of the internal CPU clock cycle.

The local bus enters the T_C bus state after the T_S state. The shortest bus cycle may have one T_S state and one T_C state. Longer bus cycles are formed by repeating T_C state. A repeated T_C bus state is called a wait state.

The $\overline{READY}$ input determines whether the current T_C bus state is to be repeated. The $\overline{READY}$ input has the same timing and effect for all bus cycles. $\overline{READY}$ is sampled at the end of each T_C bus state to see if it is active. If sampled HIGH, the T_C bus state is repeated. This is called inserting a wait state. The control and command outputs do not change during wait states.

When $\overline{READY}$ is sampled LOW, the current bus cycle is terminated. Note that the bus controller may enter the T_S bus state directly from T_C if the status lines are sampled active at the next falling edge of CLK.

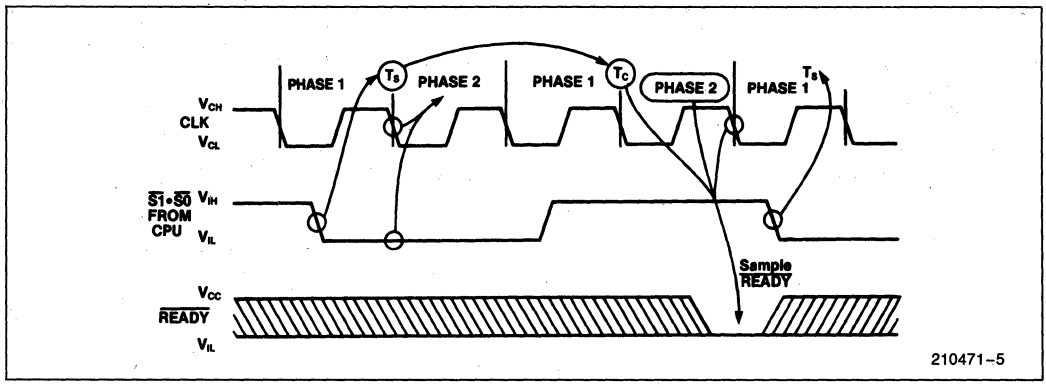


Figure 5. Bus Cycle Definition

Figures 6 through 10 show the basic command and control output timing for read and write bus cycles. Halt bus cycles are not shown since they activate no outputs. The basic idle-read-idle and idle-write-idle bus cycles are shown. The signal label CMD represents the appropriate command output for the bus cycle. For Figures 6 through 10, the CMDLY input is connected to GND and CENL to V_{CC}. The effects of CENL and CMDLY are described later in the section on control inputs.

Figures 6, 7 and 8 show non-MULTIBUS I cycles. MB is connected to GND while CEN is connected to V_{CC}. Figure 6 shows a read cycle with no wait states while Figure 7 shows a write cycle with one wait state. The $\overline{READY}$ input is shown to illustrate how wait states are added.

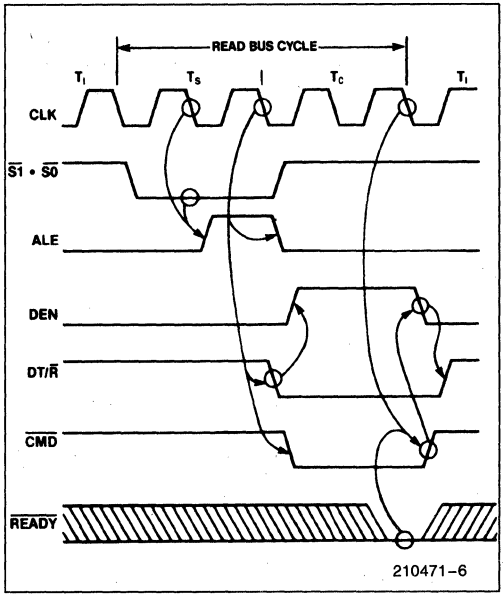


Figure 6. Idle-Read-Idle Bus Cycles with MB = 0

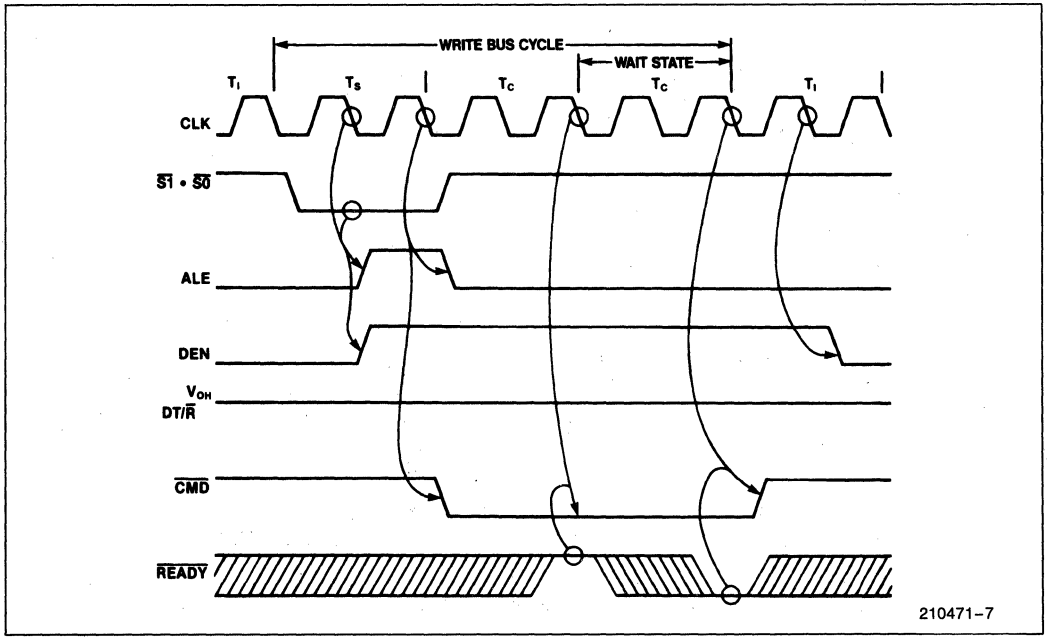


Figure 7. Idle-Write-Idle Bus Cycles with MB = 0

Bus cycles can occur back to back with no T_I bus states between T_C and T_S. Back to back cycles do not affect the timing of the command and control outputs. Command and control outputs always reach the states shown for the same clock edge (within T_S, T_C or following bus state) of a bus cycle.

A special case in control timing occurs for back to back write cycles with MB = 0. In this case, DT/$\overline{R}$ and DEN remain HIGH between the bus cycles (see Figure 8). The command and ALE output timing does not change.

Figures 9 and 10 show a MULTIBUS I cycle with MB = 1. $\overline{AEN}$ and CMDLY are connected to GND. The effects of CMDLY and $\overline{AEN}$ are described later in the section on control inputs. Figure 9 shows a read cycle with one wait state and Figure 10 shows a write cycle with two wait states. The second wait state of the write cycle is shown only for example purposes and is not required. The $\overline{READY}$ input is shown to illustrate how wait states are added.

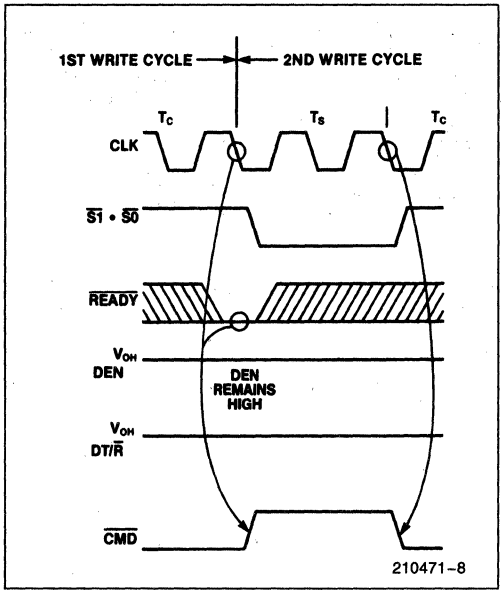


Figure 8. Write-Write Bus Cycles with MB = 0

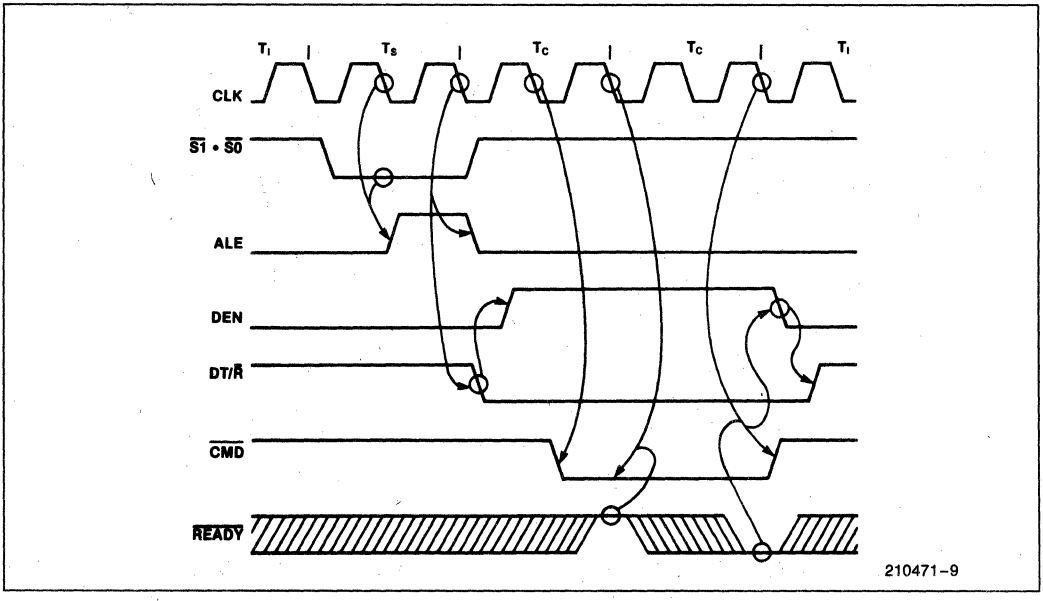


Figure 9. Idle-Read-Idle Bus Cycles with 1 Wait State and with MB = 1

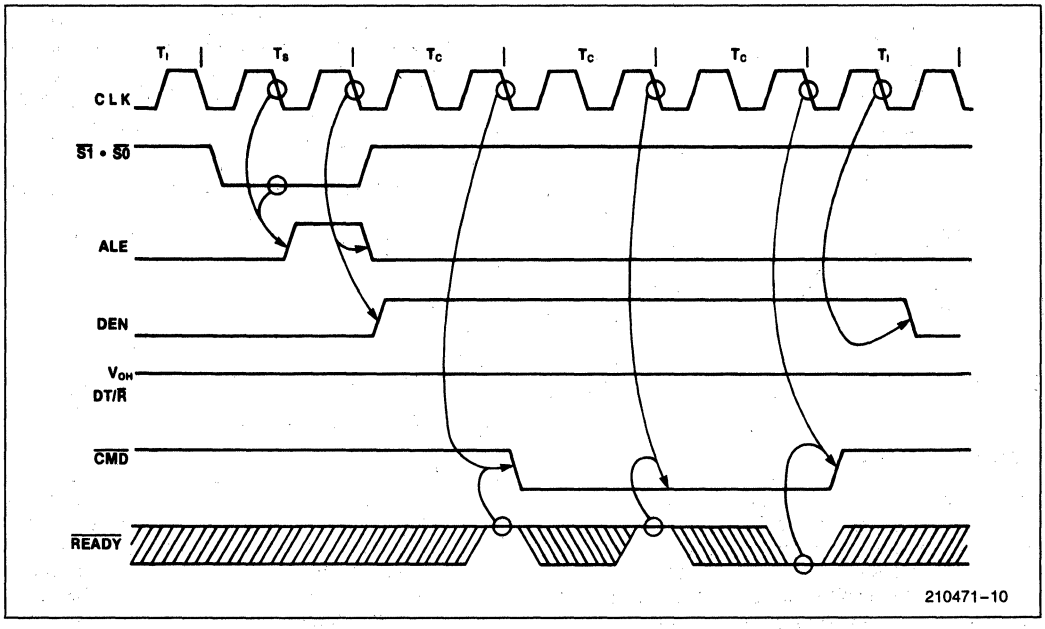


Figure 10. Idle-Write-Idle Bus Cycles with 2 Wait States and with MB = 1

The MB control input affects the timing of the command and DEN outputs. These outputs are automatically delayed in MULTIBUS I mode to satisfy three requirements:

1) 50 ns minimum setup time for valid address before any command output becomes active.
2) 50 ns minimum setup time for valid write data before any write command output becomes active.
3) 65 ns maximum time from when any read command becomes inactive until the slave's read data drivers reach 3-state OFF.

Three signal transitions are delayed by MB = 1 as compared to MB = 0:

1) The HIGH to LOW transition of the read command outputs ($\overline{IORC}$, $\overline{MRDC}$, and $\overline{INTA}$) are delayed one CLK cycle.
2) The HIGH to LOW transition of the write command outputs ($\overline{IOWC}$ and $\overline{MWTC}$) are delayed two CLK cycles.
3) The LOW to HIGH transition of DEN for write cycles is delayed one CLK cycle.

Back to back bus cycles with MB = 1 do not change the timing of any of the command or control outputs. DEN always becomes inactive between bus cycles with MB = 1.

Except for a halt or shutdown bus cycle, ALE will be issued during the second half of T_S for any bus cycle. ALE becomes inactive at the end of the T_S to allow latching the address to keep it stable during the entire bus cycle. The address outputs may change during Phase 2 of any T_C bus state. ALE is not affected by any control input.

Figure 11 shows how MCE is timed during interrupt acknowledge (INTA) bus cycles. MCE is one CLK cycle longer than ALE to hold the cascade address from a master 8259A valid after the falling edge of ALE. With the exception of the MCE control output, an INTA bus cycle is identical in timing to a read bus cycle. MCE is not affected by any control input.

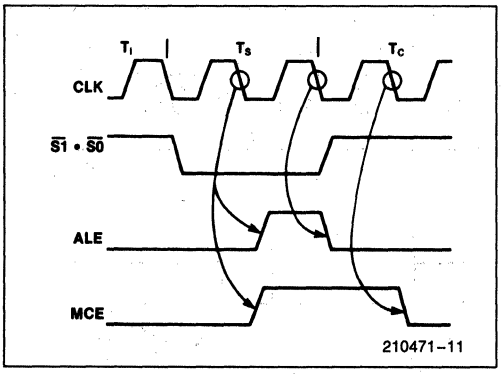


Figure 11. MCE Operation for an INTA Bus Cycle

Control Inputs

The control intputs can alter the basic timing of command outputs, allow interfacing to multiple buses, and share a bus between different masters. For many 80286 systems, each CPU will have more than one bus which may be used to perform a bus cycle. Normally, a CPU will only have one bus controller active for each bus cycle. Some buses may be shared by more than one CPU (i.e. MULTIBUS) requiring only one of them use the bus at a time.

Systems with multiple and shared buses use two control input signals of the 82288 bus controller, CENL and $\overline{AEN}$ (see Figure 12). CENL enables the bus controller to control the current bus cycle. The $\overline{AEN}$ input prevents a bus controller from driving its command outputs. $\overline{AEN}$ HIGH means that another bus controller may be driving the shared bus.

In Figure 12, two buses are shown: a local bus and a MULTIBUS I. Only one bus is used for each CPU bus cycle. The CENL inputs of the bus controller select which bus controller is to perform the bus cycle. An address decoder determines which bus to use for each bus cycle. The 82288 connected to the shared MULTIBUS I must be selected by CENL and be given access to the MULTIBUS I by $\overline{AEN}$ before it will begin a MULTIBUS I operation.

CENL must be sampled HIGH at the end of the T_S bus state (see waveforms) to enable the bus controller to activate its command and control outputs. If sampled LOW the commands and DEN will not go active and DT/$\overline{R}$ will remain HIGH. The bus controller will ignore the CMDLY, CEN, and $\overline{READY}$ inputs until another bus cycle is started via $\overline{S1}$ and $\overline{S0}$. Since an address decoder is commonly used to identify which bus is required for each bus cycle, CENL is latched to avoid the need for latching its input.

The CENL input can affect the DEN control output. When MB = 0, DEN normally becomes active during Phase 2 of T_S in write bus cycles. This transition occurs before CENL is sampled. If CENL is sampled LOW, the DEN output will be forced LOW during T_C as shown in the timing waveforms.

When MB = 1, CEN/$\overline{AEN}$ becomes $\overline{AEN}$. $\overline{AEN}$ controls when the bus controller command outputs enter and exit 3-state OFF. $\overline{AEN}$ is intended to be driven by a MULTIBUS I type bus arbiter, which assures only one bus controller is driving the shared bus at any time. When $\overline{AEN}$ makes a LOW to HIGH transition, the command outputs immediately enter 3-state OFF and DEN is forced inactive. An inactive DEN should force the local data transceivers connected to the shared data bus into 3-state OFF (see Figure 12). The LOW to HIGH transition of $\overline{AEN}$ should only occur during T_I or T_S bus states.

The HIGH to LOW transition of $\overline{AEN}$ signals that the bus controller may now drive the shared bus command signals. Since a bus cycle may be active or be in the process of starting, $\overline{AEN}$ can become active during any T-state. $\overline{AEN}$ LOW immediately allows DEN to go to the appropriate state. Three CLK edges later, the command outputs will go active (see timing waveforms). The MULTIBUS I requires this delay for the address and data to be valid on the bus before the command becomes active.

When MB = 0, CEN/$\overline{AEN}$ becomes CEN. CEN is an asynchronous input which immediately affects the command and DEN outputs. When CEN makes a HIGH to LOW transition, the commands and DEN

are immediately forced inactive. When CEN makes a LOW to HIGH transition, the commands and DEN outputs immediately go to the appropriate state (see timing waveforms). READY must still become active to terminate a bus cycle if CEN remains LOW for a selected bus controller (CENL was latched HIGH).

Some memory or I/O systems may require more address or write data setup time to command active than provided by the basic command output timing. To provide flexible command timing, the CMDLY input can delay the activation of command outputs. The CMDLY input must be sampled LOW to activate the command outputs. CMDLY does not affect the control outputs ALE, MCE, DEN, and DT/R.

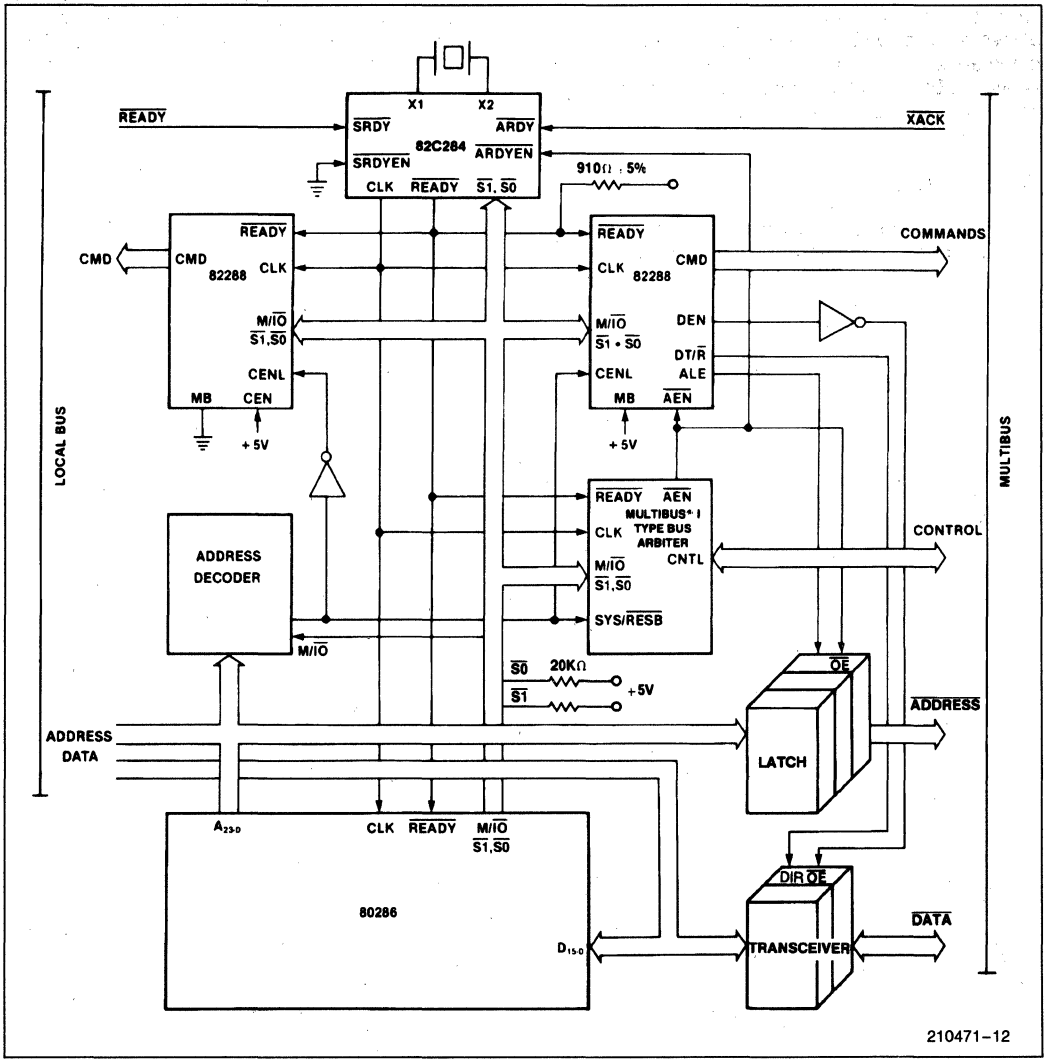


Figure 12. System Use of AEN and CENL

CMDLY is first sampled on the falling edge of the CLK ending T_S. If sampled HIGH, the command output is not activated, and CMDLY is again sampled on the next falling edge of CLK. Once sampled LOW, the proper command output becomes active immediately if MB = 0. If MB = 1, the proper command goes active no earlier than shown in Figures 9 and 10.

$\overline{\text{READY}}$ can terminate a bus cycle before CMDLY allows a command to be issued. In this case no commands are issued an the bus controller will deactivate DEN and DT/$\overline{\text{R}}$ in the same manner as if a command had been issued.

Waveforms Discussion

The waveforms show the timing relationships of inputs and outputs and do not show all possible transitions of all signals in all modes. Instead, all signal timing relationships are shown via the general cases. Special cases are shown when needed. The waveforms provide some functional descriptions of the 82288; however, most functional descriptions are provided in Figures 5 through 11.

To find the timing specification for a signal transition in a particular mode, first look for a special case in the waveforms. If no special case applies, then use a timing specification for the same or related function in another mode.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
Voltage on Any Pin with Respect to GND	−0.5V to +7V
Power Dissipation	1 Watt

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS V_{CC} = 5V ±5%, T_{CASE} = 0°C to 85°C*, or T_A = 0°C to +70°C

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input LOW Voltage	−0.5	0.8	V	
V_{IH}	Input HIGH Voltage	2.0	V_{CC} + 0.5	V	
V_{ILC}	CLK Input LOW Voltage	−0.5	0.6	V	
V_{IHC}	CLK Input HIGH Voltage	3.8	V_{CC} + 0.5	V	
V_{OL}	Output LOW Voltage				
	Command Outputs		0.45	V	I_{OL} = 32 mA (Note 1)
	Control Outputs		0.45	V	I_{OL} = 16 mA (Note 2)
V_{OH}	Output HIGH Voltage				
	Command Outputs	2.4		V	I_{OH} = −5 mA (Note 1)
	Control Outputs	2.4		V	I_{OH} = −1 mA (Note 2)
I_F	Input Current ($\overline{S0}$ and $\overline{S1}$ Inputs)		−0.5	mA	V_f = 0.45V
I_{IL}	Input Leakage Current (All Other Inputs)		±10	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{LO}	Output Leakage Current		±10	μA	0.45V ≤ V_{OUT} ≤ V_{CC}
I_{CC}	Power Supply Current		140	mA	
C_{CLK}	CLK Input Capacitance		12	pF	F_C = 1 MHz
C_I	Input Capacitance		10	pF	F_C = 1 MHz
C_O	Input/Output Capacitance		20	pF	F_C = 1 MHz

*T_A is guaranteed from 0°C to +70°C as long as T_{CASE} is not exceeded.

NOTES:
1. Command Outputs are $\overline{INTA}$, $\overline{IORC}$, $\overline{IOWC}$, $\overline{MRDC}$, $\overline{MWRC}$.
2. Control Outputs are DT/$\overline{R}$, DEN, ALE and MCE.

A.C. CHARACTERISTICS

V_{CC} = 5V, ±5%, T_{CASE} = 0°C to +85°C.* AC timings are referenced to 0.8V and 2.0V points of signals as illustrated in data sheet waveforms, unless otherwise noted.

Symbol	Parameter	8 MHz (Advance)		10 MHz (Advance)		12.5 MHz (Advance)		Unit	Test Condition
		-8 Min	-8 Max	-10 Min	-10 Min	-12 Min	-12 Max		
1	CLK Period	62	250	50	250	40	250	ns	
2	CLK HIGH Time	20	235	16	238	13	239	ns	at 3.6V
3	CLK LOW Time	15	230	12	234	11	237	ns	at 1.0V
4	CLK Rise Time		10		8		8	ns	1.0V to 3.6V
5	CLK Fall Time		10		8		8	ns	3.6V to 1.0V
6	M/$\overline{IO}$ and Status Setup Time	22		18		15		ns	
7	M/$\overline{IO}$ and Status Hold Time	1		1		1		ns	
8	CENL Setup Time	20		15		15		ns	
9	CENL Hold Time	1		1		1		ns	
10	$\overline{READY}$ Setup Time	38		26		18		ns	
11	$\overline{READY}$ Hold Time	25		25		20		ns	
12	CMDLY Setup Time	20		15		15		ns	
13	CMDLY Hold Time	1		1		1		ns	
14	$\overline{AEN}$ Setup Time	20		15		15		ns	(Note 3)
15	$\overline{AEN}$ Hold Time	0		0		0		ns	(Note 3)
16	ALE, MCE Active Delay from CLK	3	20	3	16	3	16	ns	(Note 4)
17	ALE, MCE Inactive Delay from CLK		25		19		19	ns	(Note 4)
18	DEN (Write) Inactive from CENL		35		23		23	ns	(Note 4)
19	DT/$\overline{R}$ LOW from CLK		25		23		23	ns	(Note 4)
20	DEN (Read) Active from DT/$\overline{R}$	5	35	5	21	5	21	ns	(Note 4)
21	DEN (Read) Inactive Dly from CLK	3	35	3	21	3	19	ns	(Note 4)
22	DT/$\overline{R}$ HIGH from DEN Inactive	5	35	5	20	5	18	ns	(Note 4)
23	DEN (Write) Active Delay from CLK		30		23		23	ns	(Note 4)
24	DEN (Write) Inactive Dly from CLK	3	30	3	19	3	19	ns	(Note 4)

*T_A is guaranteed from 0°C to +70°C as long as T_{CASE} is not exceeded.

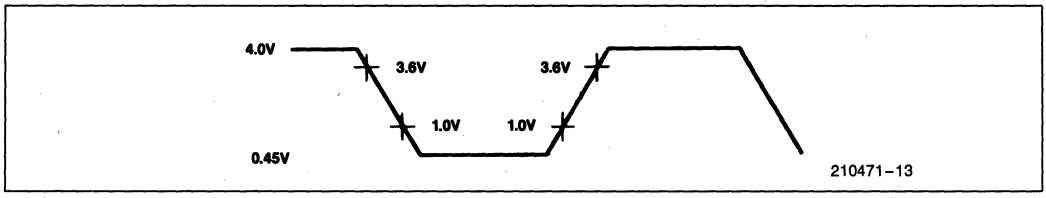
A.C. CHARACTERISTICS

V_{CC} = 5V, ±5%, T_{CASE} = 0°C to +85°C.* AC timings are referenced to 0.8V and 2.0V points of signals as illustrated in data sheet waveforms, unless otherwise noted. (Continued)

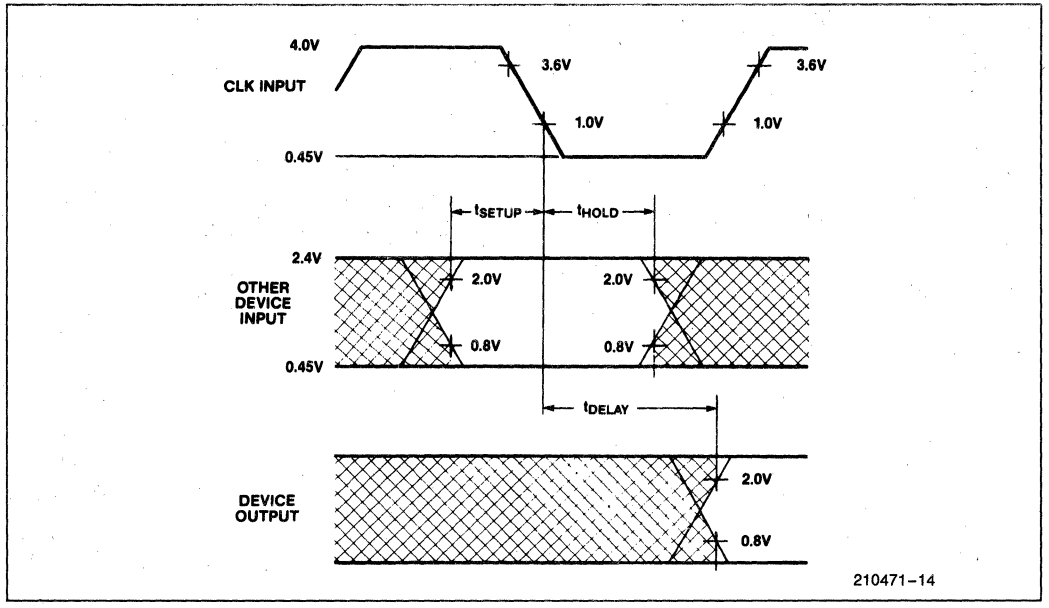
Symbol	Parameter	8 MHz (Advance)		10 MHz (Advance)		12.5 MHz (Advance)		Unit	Test Condition
		-8 Min	-8 Max	-10 Min	-10 Min	-12 Min	-12 Max		
25	DEN Inactive from CEN		30		25		25	ns	(Note 4)
26	DEN Active from CEN		30		24		24	ns	(Note 4)
27	DT/$\overline{R}$ HIGH from CLK (when CEN = LOW)		35		25		25	ns	(Note 4)
28	DEN Active from $\overline{AEN}$		30		26		26	ns	(Note 4)
29	$\overline{CMD}$ Active Delay from CLK	3	25	3	21	3	21	ns	(Note 5)
30	$\overline{CMD}$ Inactive Delay from CLK	5	25	5	20	5	20	ns	(Note 5)
31	$\overline{CMD}$ Active from CEN		25		25		25	ns	(Note 5)
32	$\overline{CMD}$ Inactive from CEN		25		25		25	ns	(Note 5)
33	$\overline{CMD}$ Inactive Enable from $\overline{AEN}$		40		40		40	ns	(Note 5)
34	$\overline{CMD}$ Float Delay from $\overline{AEN}$		40		40		40	ns	(Note 6)
35	MB Setup Time	20		20		20		ns	
36	MB Hold Time	0		0		0		ns	
37	Command Inactive Enable from MB ↓		40		40		40	ns	(Note 5)
38	Command Float Time from MB ↑		40		40		40	ns	(Note 6)
39	DEN Inactive from MB ↑		30		26		26	ns	(Note 4)
40	DEN Active from MB ↓		30		30		30	ns	(Note 4)

*T_A is guaranteed from 0°C to +70°C as long as T_{CASE} is not exceeded.

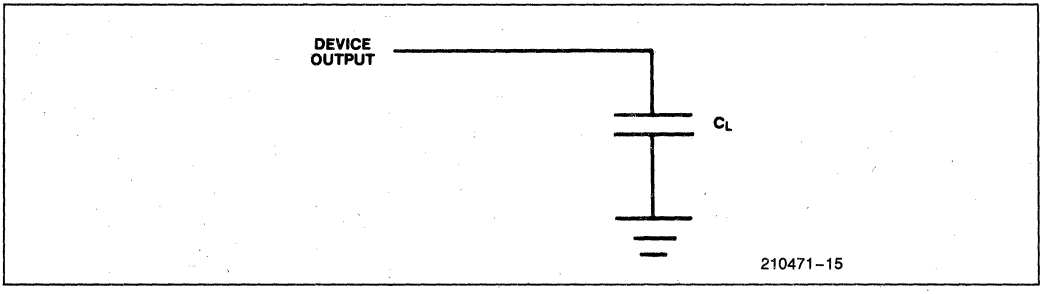
NOTES:
3. $\overline{AEN}$ is an asynchronous input. This specification is for testing purposes only, to assure recognition at a specific CLK edge.
4. Control output load: Cl = 150 pF.
5. Command output load: Cl = 300 pF.
6. Float condition occurs when output current is less than I_{LO} in magnitude.



Note 7: AC Drive and Measurement Points—CLK Input



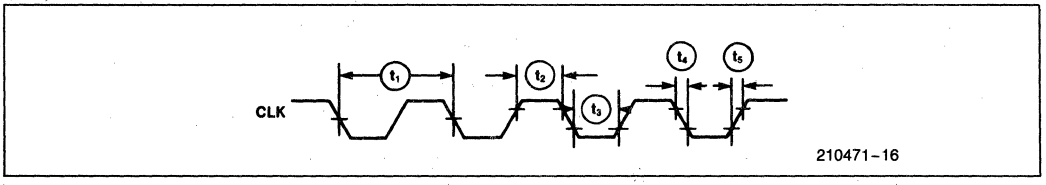
Note 8: AC Setup, Hold and Delay Time Measurement—General



Note 9: AC Test Loading on Outputs

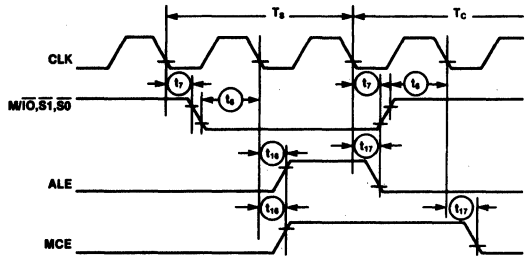
WAVEFORMS

CLK CHARACTERISTICS



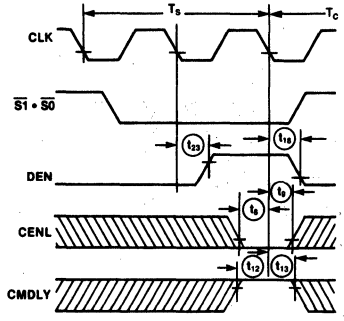
WAVEFORMS (Continued)

STATUS, ALE, MCE, CHARACTERISTICS



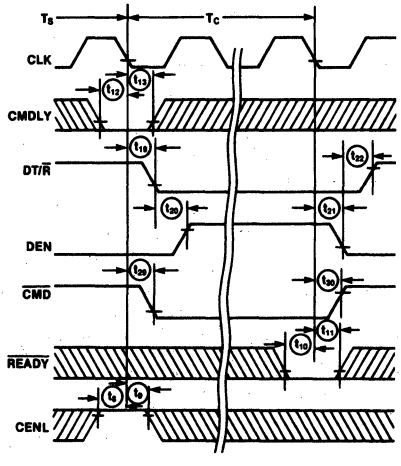
210471-17

CENL, CMDLY, DEN CHARACTERISTICS WITH MB = 0 AND CEN = 1 DURING WRITE CYCLE



210471-18

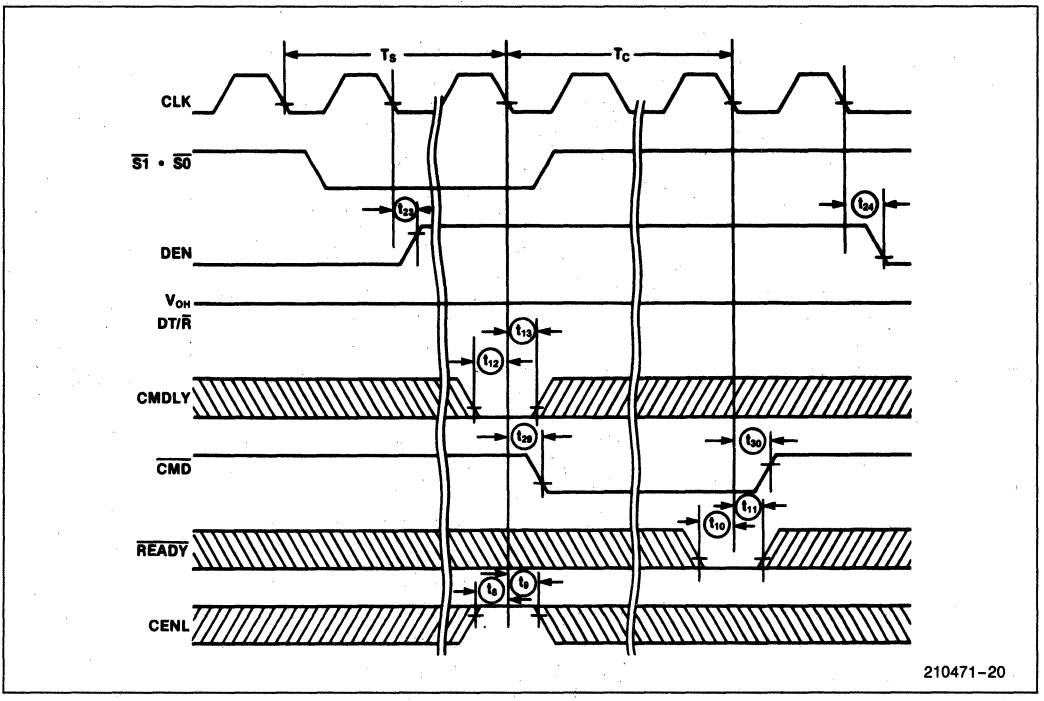
READ CYCLE CHARACTERISTICS WITH MB = 0 AND CEN = 1



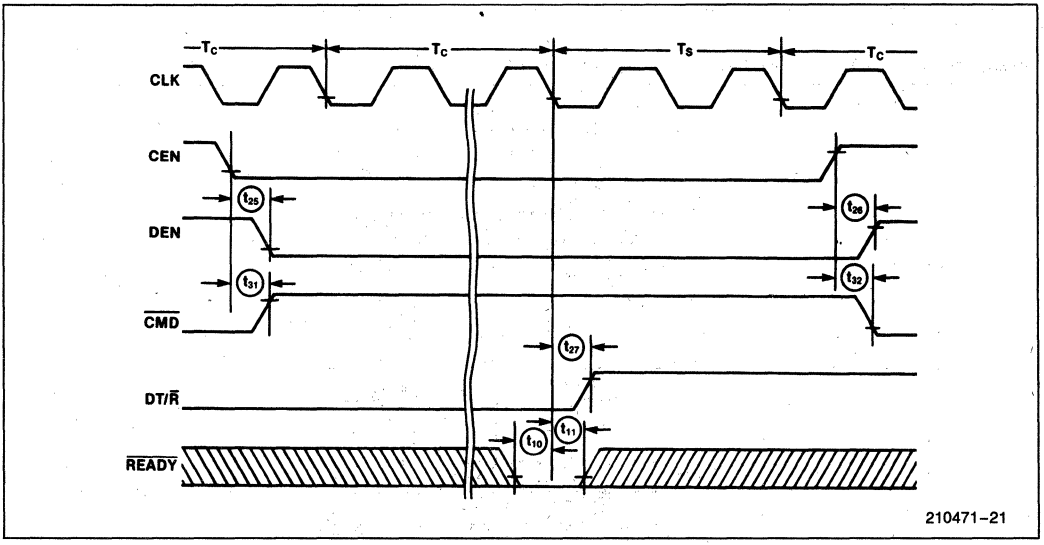
210471-19

WAVEFORMS (Continued)

WRITE CYCLE CHARACTERISTIC WITH MB = 0 AND CEN = 1

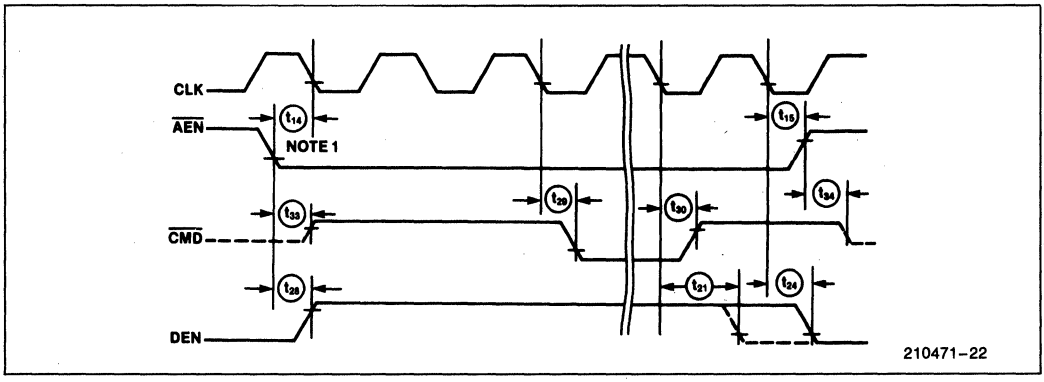


CEN CHARACTERISTICS WITH MB = 0



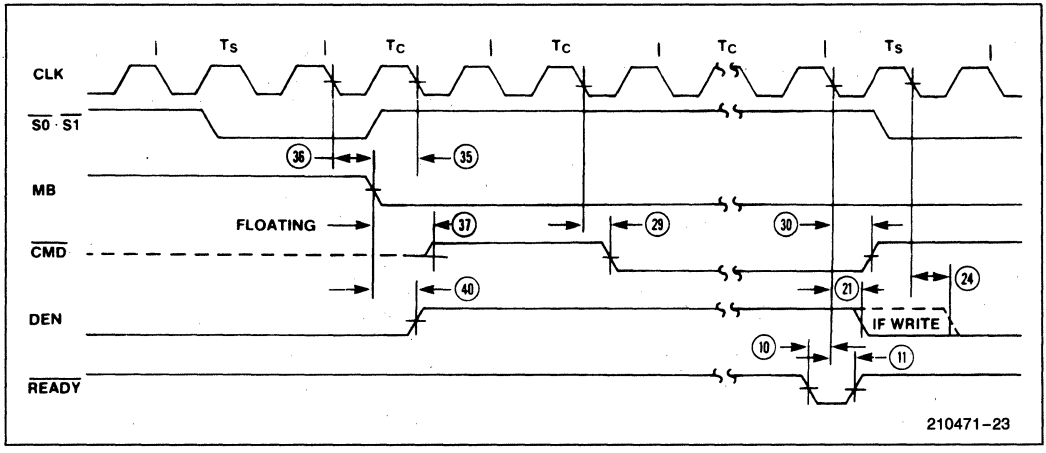
WAVEFORMS (Continued)

$\overline{AEN}$ CHARACTERISTICS WITH MB = 1



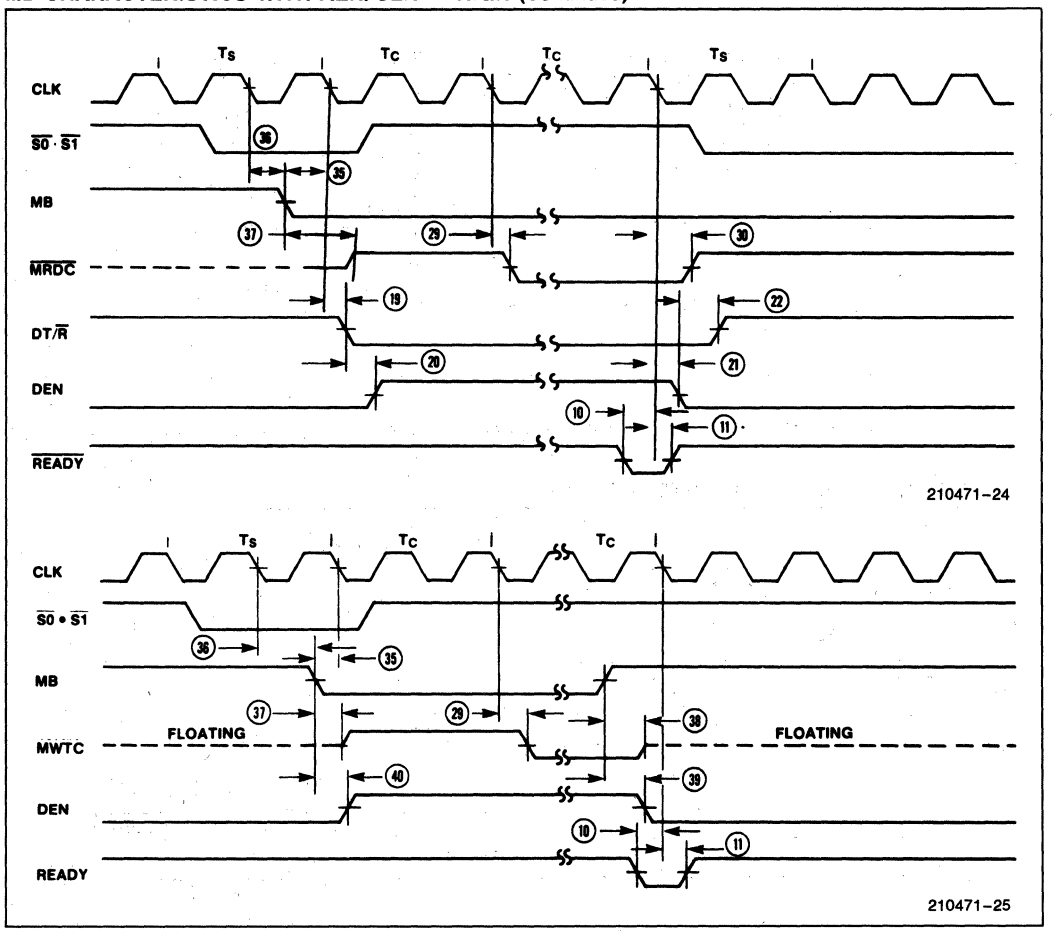
NOTE:
1. $\overline{AEN}$ is an asynchronous input. $\overline{AEN}$ setup and hold time is specified to guarantee the response shown in the waveforms.

MB CHARACTERISTICS WITH $\overline{AEN}$/CEN = HIGH



WAVEFORMS (Continued)

MB CHARACTERISTICS WITH $\overline{AEN}$/CEN = HIGH (Continued)



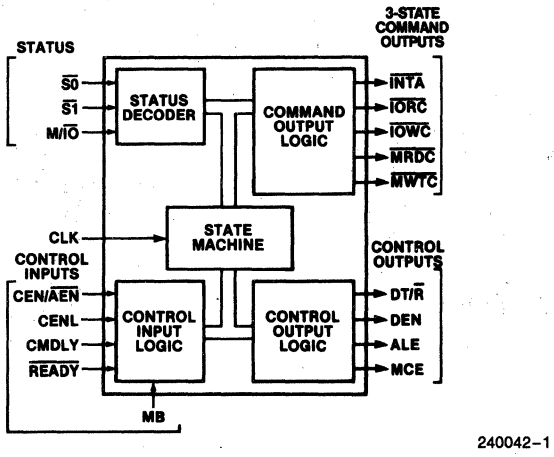
NOTES:
1. MB is an asynchronous input. MB setup and hold times specified to guarantee the response shown in the waveforms.
2. If the setup time, t35, is met two clock cycles will occur before $\overline{CMD}$ becomes active after the falling edge of MB.

82C288
BUS CONTROLLER FOR 80286 PROCESSORS
(82C288-12, 82C288-10, 82C288-8)

- **Provides Commands and Controls for Local and System Bus**
- **Wide Flexibility in System Configurations**
- **Implemented in High Speed CHMOS III Technology**
- **Fully Compatible with the HMOS 82288**
- **Fully Static Device**
- **Single +5V Supply**
- **Available in 20 Pin PLCC (Plastic Leaded Chip Carrier) and 20 Pin Cerdip Packages**
  (See Packaging Spec, Order #231369)

The Intel 82C288 Bus Controller is a 20-pin CHMOS III component for use in 80286 microsystems. The 82C288 is fully compatible with its predecessor the HMOS 82288. The bus controller is fully static and supports a low power mode. The bus controller provides command and control outputs with flexible timing options. Separate command outputs are used for memory and I/O devices. The data bus is controlled with separate data enable and direction control signals.

Two modes of operation are possible via a strapping option: MULTIBUS® I compatible bus cycles, and high speed bus cycles.

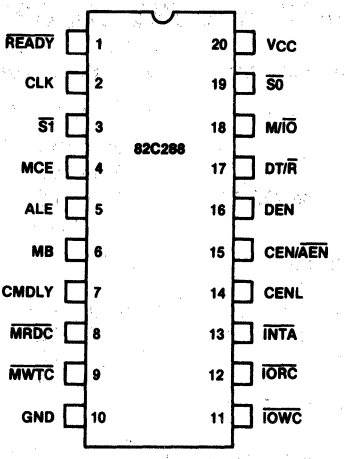


240042-1

Figure 1. 82C288 Block Diagram

September 1988
Order Number: 240042-002

20 Pin Cerdip Package

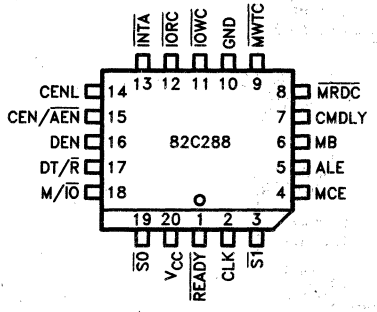


240042-2

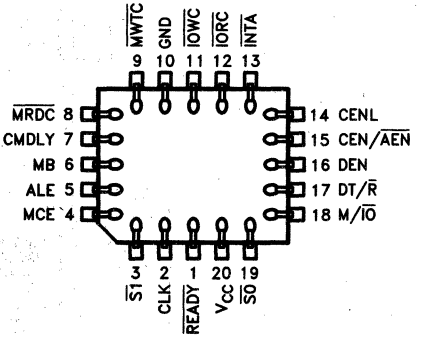
P.C. Board Views—As viewed from the component side of the P.C. board.

Component Pad Views—As viewed from underside of component when mounted on the board.

20 Pin PLCC Package



240042-3



240042-4

Figure 2. 82C288 Pin Configuration

Table 1. Pin Description

The following pin function descriptions are for the 82C288 bus controller.

Symbol	Type	Name and Function
CLK	I	**SYSTEM CLOCK** provides the basic timing control for the 82C288 in an 80286 microsystem. Its frequency is twice the internal processor clock frequency. The falling edge of this input signal establishes when inputs are sampled and command and control outputs change.
$\overline{S0}$, $\overline{S1}$	I	**BUS CYCLE STATUS** starts a bus cycle and, along with M/$\overline{IO}$, defines the type of bus cycle. These inputs are active LOW. A bus cycle is started when either $\overline{S1}$ or $\overline{S0}$ is sampled LOW at the falling edge of CLK. Setup and hold times must be met for proper operation.

80286 Bus Cycle Status Definition

M/$\overline{IO}$	$\overline{S1}$	$\overline{S0}$	Type of Bus Cycle
0	0	0	Interrupt Acknowledge
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	None; Idle
1	0	0	Halt or Shutdown
1	0	1	Memory Read
1	1	0	Memory Write
1	1	1	None; Idle

M/$\overline{IO}$	I	**MEMORY OR I/O SELECT** determines whether the current bus cycle is in the memory space or I/O space. When LOW, the current bus cycle is in the I/O space. Setup and hold times must be met for proper operation.
MB	I	**MULTIBUS MODE SELECT** determines timing of the command and control outputs. When HIGH, the bus controller operates with MULTIBUS I compatible timings. When LOW, the bus controller optimizes the command and control output timing for short bus cycles. The function of the CEN/$\overline{AEN}$ input pin is selected by this signal. This input is typically a strapping option and not dynamically changed.
CENL	I	**COMMAND ENABLE LATCHED** is a bus controller select signal which enables the bus controller to respond to the current bus cycle being initiated. CENL is an active HIGH input latched internally at the end of each T_S cycle. CENL is used to select the appropriate bus controller for each bus cycle in a system where the CPU has more than one bus it can use. This input may be connected to V_{CC} to select this 82C288 for all transfers. No control inputs affect CENL. Setup and hold times must be met for proper operation.
CMDLY	I	**COMMAND DELAY** allows delaying the start of a command. CMDLY is an active HIGH input. If sampled HIGH, the command output is not activated and CMDLY is again sampled at the next CLK cycle. When sampled LOW the selected command is enabled. If $\overline{READY}$ is detected LOW before the command output is activated, the 82C288 will terminate the bus cycle, even if no command was issued. Setup and hold times must be satisfied for proper operation. This input may be connected to GND if no delays are required before starting a command. This input has no effect on 82C288 control outputs.
$\overline{READY}$	I	**READY** indicates the end of the current bus cycle. $\overline{READY}$ is an active LOW input. MULTIBUS I mode requires at least one wait state to allow the command outputs to become active. $\overline{READY}$ must be LOW during reset, to force the 82C288 into the idle state. Setup and hold times must be met for proper operation. The 82C284 drives $\overline{READY}$ LOW during RESET.

Table 1. Pin Description (Continued)

Symbol	Type	Name and Function
CEN/$\overline{AEN}$	I	**COMMAND ENABLE/ADDRESS ENABLE** controls the command and DEN outputs of the bus controller. CEN/$\overline{AEN}$ inputs may be asynchronous to CLK. Setup and hold times are given to assure a guaranteed response to synchronous inputs. This input may be connected to V_{CC} or GND. When MB is HIGH this pin has the $\overline{AEN}$ function. $\overline{AEN}$ is an active LOW input which indicates that the CPU has been granted use of a shared bus and the bus contoller command outputs may exit 3-state OFF and become inactive (HIGH). $\overline{AEN}$ HIGH indicates that the CPU does not have control of the shared bus and forces the command outputs into 3-state OFF and DEN inactive (LOW). When MB is LOW this pin has the CEN function. CEN is an unlatched active HIGH input which allows the bus controller to activate its command and DEN outputs. With MB LOW, CEN LOW forces the command and DEN outputs inactive but does not tristate them.
ALE	O	**ADDRESS LATCH ENABLE** controls the address latches used to hold an address stable during a bus cycle. This control output is active HIGH. ALE will not be issued for the halt bus cycle and is not affected by any of the control inputs.
MCE	O	**MASTER CASCADE ENABLE** signals that a cascade address from a master 8259A interrupt controller may be placed onto the CPU address bus for latching by the address latches under ALE control. The CPU's address bus may then be used to broadcast the cascade address to slave interrupt controllers so only one of them will respond to the interrupt acknowledge cycle. This control output is active HIGH. MCE is only active during interrupt acknowledge cycles and is not affected by any control input. Using MCE to enable cascade address drivers requires latches which save the cascade address on the falling edge of ALE.
DEN	O	**DATA ENABLE** controls when data transceivers connected to the local data bus should be enabled. DEN is an active HIGH control output. DEN is delayed for write cycles in the MULTIBUS I mode.
DT/$\overline{R}$	O	**DATA TRANSMIT/RECEIVE** establishes the direction of data flow to or from the local data bus. When HIGH, this control output indicates that a write bus cycle is being performed. A LOW indicates a read bus cycle. DEN is always inactive when DT/$\overline{R}$ changes states. This output is HIGH when no bus cycle is active. DT/$\overline{R}$ is not affected by any of the control inputs.
$\overline{IOWC}$	O	**I/O WRITE COMMAND** instructs an I/O device to read the data on the data bus. This command output is active LOW. The MB and CMDLY inputs control when this output becomes active. $\overline{READY}$ controls when it becomes inactive.
$\overline{IORC}$	O	**I/O READ COMMAND** instructs an I/O device to place data onto the data bus. This command output is active LOW. The MB and CMDLY inputs control when this output becomes active. $\overline{READY}$ controls when it becomes inactive.
$\overline{MWTC}$	O	**MEMORY WRITE COMMAND** instructs a memory device to read the data on the data bus. This command output is active LOW. The MB and CMDLY inputs control when this output becomes active. $\overline{READY}$ controls when it becomes inactive.
$\overline{MRDC}$	O	**MEMORY READ COMMAND** instructs the memory device to place data onto the data bus. This command output is active LOW. The MB and CMDLY inputs control when this output becomes active. $\overline{READY}$ controls when it becomes inactive.

Table 1. Pin Description (Continued)

Symbol	Type	Name and Function
$\overline{INTA}$	O	**INTERRUPT ACKNOWLEDGE** tells an interrupting device that its interrupt request is being acknowledged. This command output is active LOW. The MB and CMDLY inputs control when this output becomes active. $\overline{READY}$ controls when it becomes inactive.
V_{CC}		**System Power:** +5V Power Supply
GND		**System Ground:** 0V

Table 2. Command and Control Outputs for Each Type of Bus Cycle

Type of Bus Cycle	M/$\overline{IO}$	$\overline{S1}$	$\overline{S0}$	Command Activated	DT/$\overline{R}$ State	ALE, DEN Issued?	MCE Issued?
Interrupt Acknowledge	0	0	0	$\overline{INTA}$	LOW	YES	YES
I/O Read	0	0	1	$\overline{IORC}$	LOW	YES	NO
I/O Write	0	1	0	$\overline{IOWC}$	HIGH	YES	NO
None; Idle	0	1	1	None	HIGH	NO	NO
Halt/Shutdown	1	0	0	None	HIGH	NO	NO
Memory Read	1	0	1	$\overline{MRDC}$	LOW	YES	NO
Memory Write	1	1	0	$\overline{MWTC}$	HIGH	YES	NO
None; Idle	1	1	1	None	HIGH	NO	NO

Operating Modes

Two types of buses are supported by the 82C288: MULTIBUS I and non-MULTIBUS I. When the MB input is strapped HIGH, MULTIBUS I timing is used. In MULTIBUS I mode, the 82C288 delays command and data activation to meet IEEE-796 requirements on address to command active and write data to command active setup timing. MULTIBUS I mode requires at least one wait state in the bus cycle since the command outputs are delayed. The non-MULTIBUS I mode does not delay any outputs and does not require wait states. The MB input affects the timing of the command and DEN outputs.

Command and Control Outputs

The type of bus cycle performed by the local bus master is encoded in the M/$\overline{IO}$, $\overline{S1}$, and $\overline{S0}$ inputs. Different command and control outputs are activated depending on the type of bus cycle. Table 2 indicates the cycle decode done by the 82C288 and the effect on command, DT/$\overline{R}$, ALE, DEN, and MCE outputs.

Bus cycles come in three forms: read, write, and halt. Read bus cycles include memory read, I/O read, and interrupt acknowledge. The timing of the associated read command outputs ($\overline{MRDC}$, $\overline{IORC}$, and $\overline{INTA}$), control outputs (ALE, DEN, DT/$\overline{R}$) and control inputs (CEN/$\overline{AEN}$, CENL, CMDLY, MB, and $\overline{READY}$) are identical for all read bus cycles. Read cycles differ only in which command output is activated. The MCE control output is only asserted during interrupt acknowledge cycles.

Write bus cycles activate different control and command outputs with different timing than read bus cycles. Memory write and I/O write are write bus cycles whose timing for command outputs ($\overline{MWTC}$ and $\overline{IOWC}$), control outputs (ALE, DEN, DT/$\overline{R}$) and control inputs (CEN/$\overline{AEN}$, CENL, CMDLY, MB, and $\overline{READY}$) are identical. They differ only in which command output is activated.

Halt bus cycles are different because no command or control output is activated. All control inputs are ignored until the next bus cycle is started via $\overline{S1}$ and $\overline{S0}$.

Static Operation

All 82C288 circuitry is of static design. Internal registers and logic are static and require no refresh as with dynamic circuit design. This eliminates the minimum operating frequency restriction placed on the HMOS 82288. The CHMOS III 82C288 can operate from DC to the appropriate upper frequency limit.

The clock may be stopped in either state (HIGH/LOW) and held there indefinitely.

Power dissipation is directly related to operating frequency. As the system frequency is reduced, so is the operating power. When the clock is stopped to the 82C288, power dissipation is at a minimum. This is useful for low-power and portable applications.

FUNCTIONAL DESCRIPTION

Introduction

The 82C288 bus controller is used in 80286 systems to provide address latch control, data transceiver control, and standard level-type command outputs. The command outputs are timed and have sufficient drive capabilities for large TTL buses and meet all IEEE-796 requirements for MULTIBUS I. A special MULTIBUS I mode is provided to satisfy all address/data setup and hold time requirements. Command timing may be tailored to special needs via a CMDLY input to determine the start of a command and $\overline{\text{READY}}$ to determine the end of a command.

Connection to multiple buses are supported with a latched enable input (CENL). An address decoder can determine which, if any, bus controller should be enabled for the bus cycle. This input is latched to allow an address decoder to take full advantage of the pipelined timing on the 80286 local bus.

Buses shared by several bus controllers are supported. An $\overline{\text{AEN}}$ input prevents the bus controller from driving the shared bus command and data signals except when enabled by an external MULTIBUS I type bus arbiter.

Separate DEN and DT/$\overline{\text{R}}$ outputs control the data transceivers for all buses. Bus contention is eliminated by disabling DEN before changing DT/$\overline{\text{R}}$. The DEN timing allows sufficient time for tristate bus drivers to enter 3-state OFF before enabling other drivers onto the same bus.

The term CPU refers to any 80286 processor or 80286 support component which may become an 80286 local bus master and thereby drive the 82C288 status inputs.

Processor Cycle Definition

Any CPU which drives the local bus uses an internal clock which is one half the frequency of the system clock (CLK) (see Figure 3). Knowledge of the phase of the local bus master internal clock is required for proper operation of the 80286 local bus. The local bus master informs the bus controller of its internal clock phase when it asserts the status signals. Status signals are always asserted beginning in Phase 1 of the local bus master's internal clock.

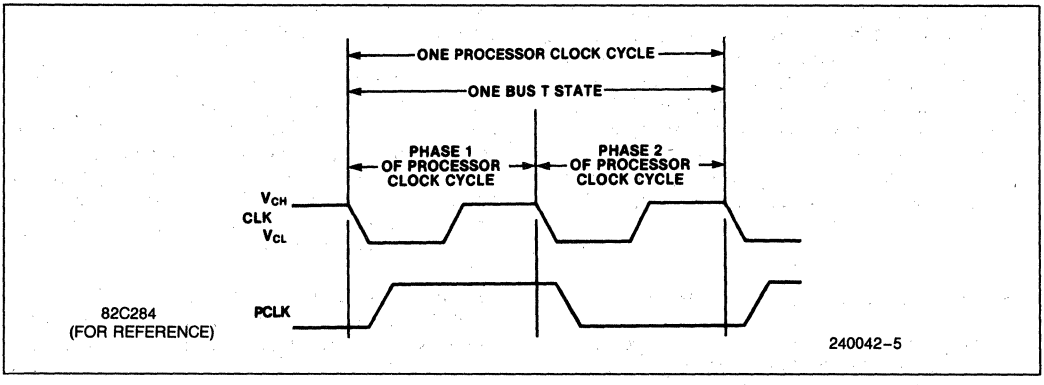


Figure 3. CLK Relationship to the Processor Clock and Bus T-States

Bus State Definition

The 82C288 bus controller has three bus states (see Figure 4): Idle (T_I) Status (T_S) and Command (T_C). Each bus state is two CLK cycles long. Bus state phases correspond to the internal CPU processor clock phases.

The T_I bus state occurs when no bus cycle is currently active on the 80286 local bus. This state may be repeated indefinitely. When control of the local bus is being passed between masters, the bus remains in the T_I state.

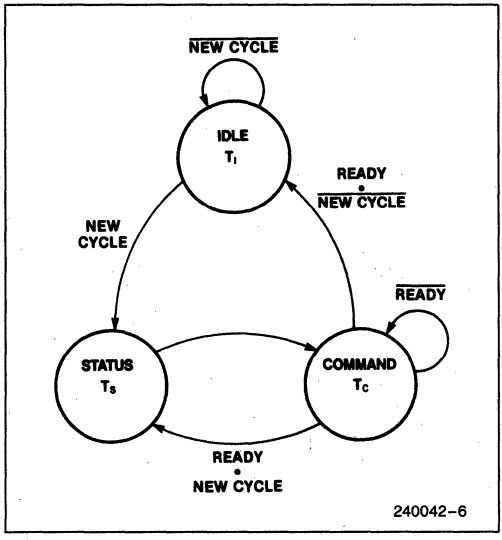


Figure 4. 82C288 Bus States

Bus Cycle Definition

The $\overline{S1}$ and $\overline{S0}$ inputs signal the start of a bus cycle. When either input becomes LOW, a bus cycle is started. The T_S bus state is defined to be the two CLK cycles during which either $\overline{S1}$ or $\overline{S0}$ are active (see Figure 5). These inputs are sampled by the 82C288 at every falling edge of CLK. When either $\overline{S1}$ or $\overline{S0}$ are sampled LOW, the next CLK cycle is considered the second phase of the internal CPU clock cycle.

The local bus enters the T_C bus state after the T_S state. The shortest bus cycle may have one T_S state and one T_C state. Longer bus cycles are formed by repeating T_C state. A repeated T_C bus state is called a wait state.

The $\overline{READY}$ input determines whether the current T_C bus state is to be repeated. The $\overline{READY}$ input has the same timing and effect for all bus cycles. $\overline{READY}$ is sampled at the end of each T_C bus state to see if it is active. If sampled HIGH, the T_C bus state is repeated. This is called inserting a wait state. The control and command outputs do not change during wait states.

When $\overline{READY}$ is sampled LOW, the current bus cycle is terminated. Note that the bus controller may enter the T_S bus state directly from T_C if the status lines are sampled active at the next falling edge of CLK.

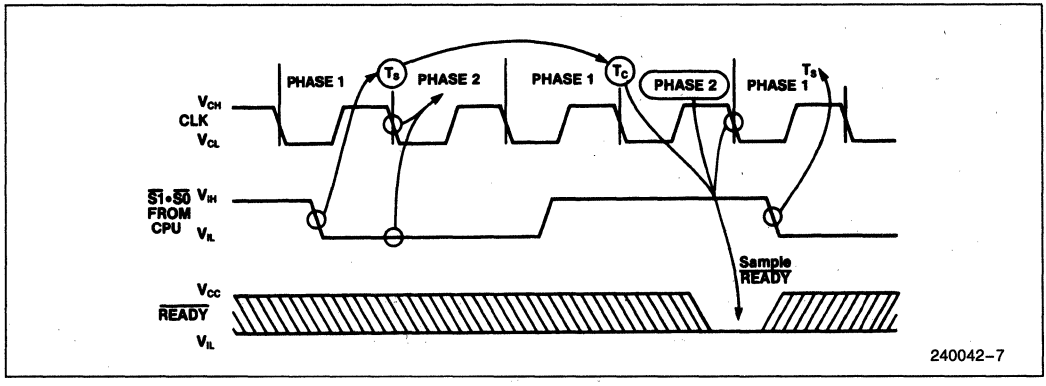


Figure 5. Bus Cycle Definition

Figures 6 through 10 show the basic command and control output timing for read and write bus cycles. Halt bus cycles are not shown since they activate no outputs. The basic idle-read-idle and idle-write-idle bus cycles are shown. The signal label CMD represents the appropriate command output for the bus cycle. For Figures 6 through 10, the CMDLY input is connected to GND and CENL to V_{CC}. The effects of CENL and CMDLY are described later in the section on control inputs.

Figures 6, 7 and 8 show non-MULTIBUS I cycles. MB is connected to GND while CEN is connected to V_{CC}. Figure 6 shows a read cycle with no wait states while Figure 7 shows a write cycle with one wait state. The $\overline{READY}$ input is shown to illustrate how wait states are added.

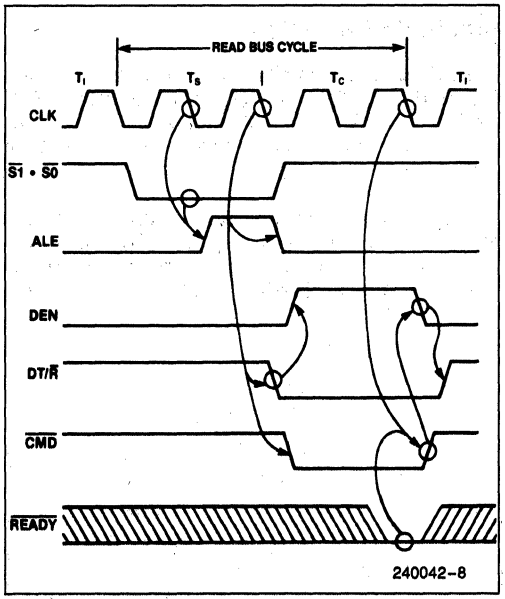


Figure 6. Idle-Read-Idle Bus Cycles with MB = 0

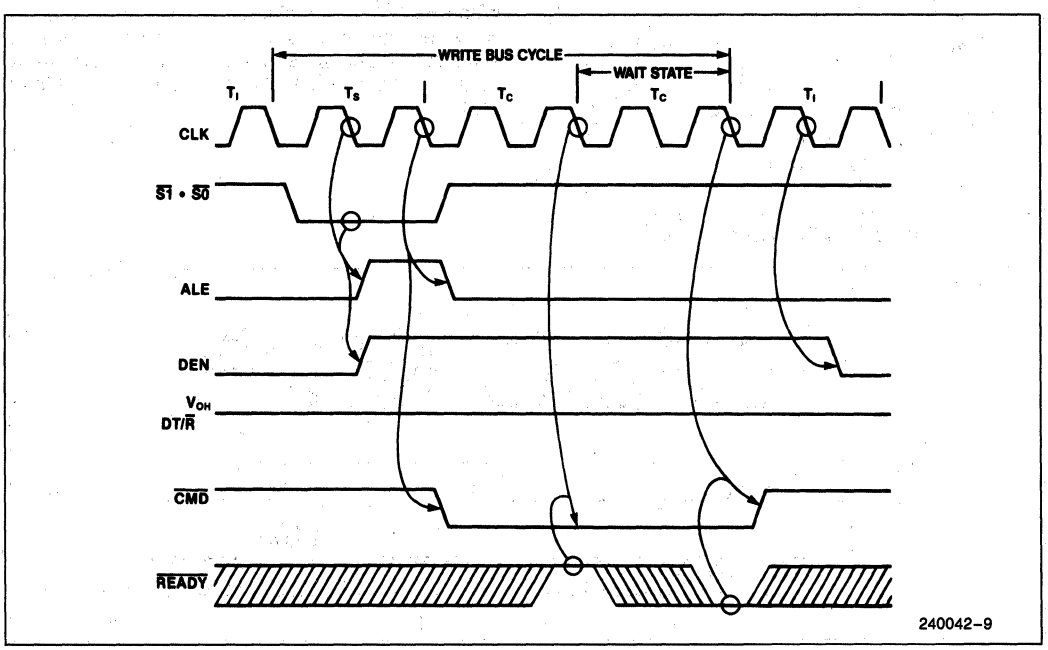


Figure 7. Idle-Write-Idle Bus Cycles with MB = 0

Bus cycles can occur back to back with no T_I bus states between T_C and T_S. Back to back cycles do not affect the timing of the command and control outputs. Command and control outputs always reach the states shown for the same clock edge (within T_S, T_C or following bus state) of a bus cycle.

A special case in control timing occurs for back to back write cycles with MB = 0. In this case, DT/$\overline{R}$ and DEN remain HIGH between the bus cycles (see Figure 8). The command and ALE output timing does not change.

Figures 9 and 10 show a MULTIBUS I cycle with MB = 1. $\overline{AEN}$ and CMDLY are connected to GND. The effects of CMDLY and $\overline{AEN}$ are described later in the section on control inputs. Figure 9 shows a read cycle with one wait state and Figure 10 shows a write cycle with two wait states. The second wait state of the write cycle is shown only for example purposes and is not required. The $\overline{READY}$ input is shown to illustrate how wait states are added.

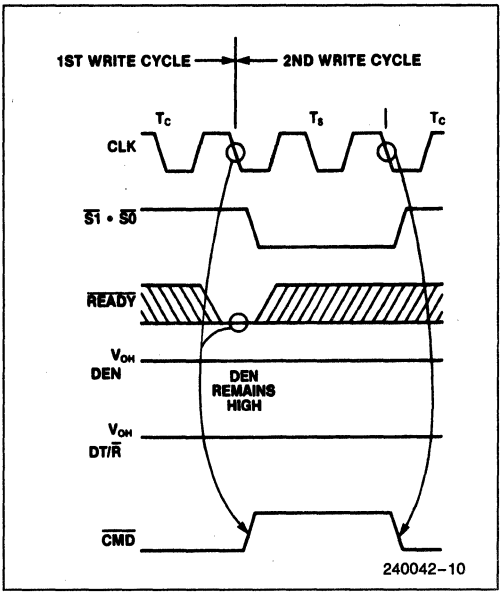


Figure 8. Write-Write Bus Cycles with MB = 0

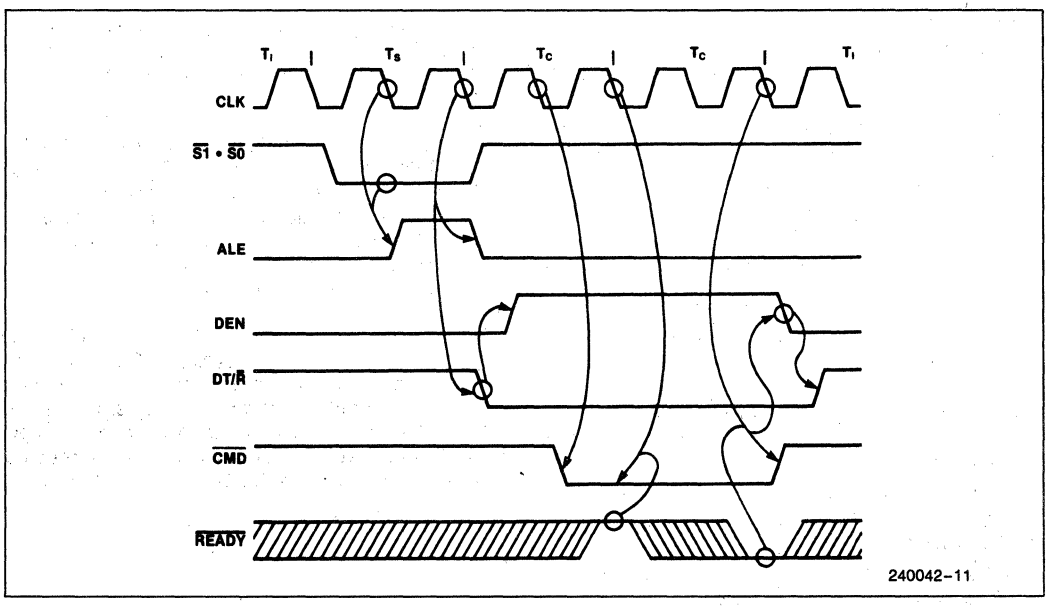


Figure 9. Idle-Read-Idle Bus Cycles with 1 Wait State and with MB = 1

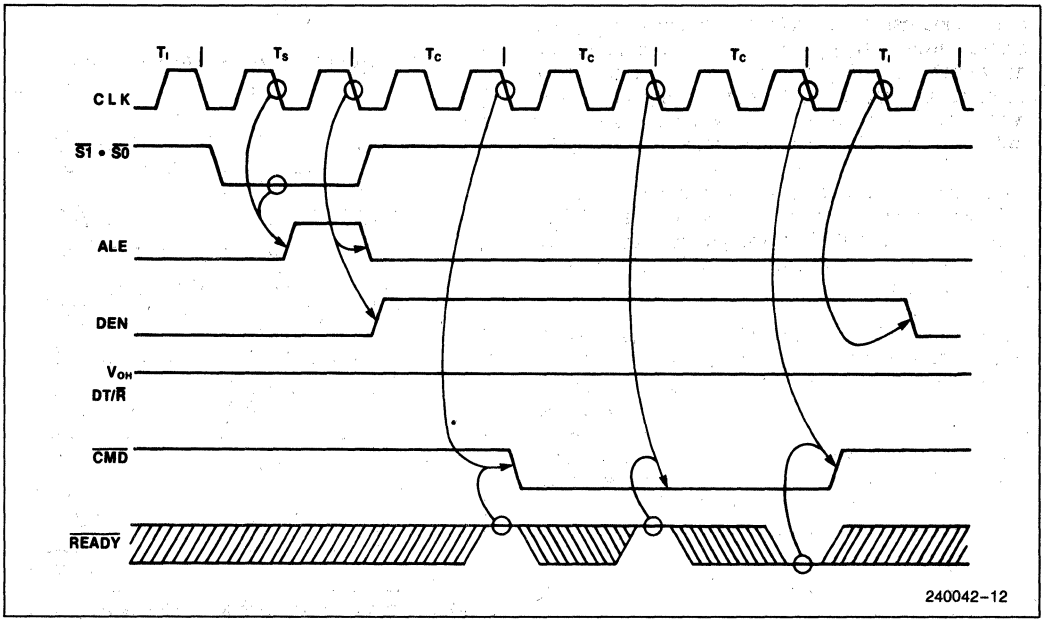


Figure 10. Idle-Write-Idle Bus Cycles with 2 Wait States and with MB = 1

The MB control input affects the timing of the command and DEN outputs. These outputs are automatically delayed in MULTIBUS I mode to satisfy three requirements:

1) 50 ns minimum setup time for valid address before any command output becomes active.
2) 50 ns minimum setup time for valid write data before any write command output becomes active.
3) 65 ns maximum time from when any read command becomes inactive until the slave's read data drivers reach 3-state OFF.

Three signal transitions are delayed by MB = 1 as compared to MB = 0:

1) The HIGH to LOW transition of the read command outputs ($\overline{IORC}$, $\overline{MRDC}$, and $\overline{INTA}$) are delayed one CLK cycle.
2) The HIGH to LOW transition of the write command outputs ($\overline{IOWC}$ and $\overline{MWTC}$) are delayed two CLK cycles.
3) The LOW to HIGH transition of DEN for write cycles is delayed one CLK cycle.

Back to back bus cycles with MB = 1 do not change the timing of any of the command or control outputs. DEN always becomes inactive between bus cycles with MB = 1.

Except for a halt or shutdown bus cycle, ALE will be issued during the second half of T_S for any bus cycle. ALE becomes inactive at the end of the T_S to allow latching the address to keep it stable during the entire bus cycle. The address outputs may change during Phase 2 of any T_C bus state. ALE is not affected by any control input.

Figure 11 shows how MCE is timed during interrupt acknowledge (INTA) bus cycles. MCE is one CLK cycle longer than ALE to hold the cascade address from a master 8259A valid after the falling edge of ALE. With the exception of the MCE control output, an INTA bus cycle is identical in timing to a read bus cycle. MCE is not affected by any control input.

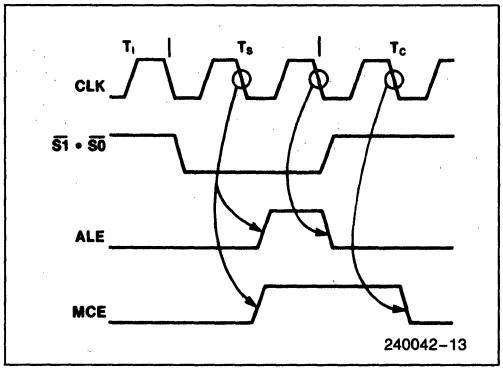


Figure 11. MCE Operation for an INTA Bus Cycle

Control Inputs

The control intputs can alter the basic timing of command outputs, allow interfacing to multiple buses, and share a bus between different masters. For many 80286 systems, each CPU will have more than one bus which may be used to perform a bus cycle. Normally, a CPU will only have one bus controller active for each bus cycle. Some buses may be shared by more than one CPU (i.e. MULTIBUS) requiring only one of them use the bus at a time.

Systems with multiple and shared buses use two control input signals of the 82C288 bus controller, CENL and $\overline{AEN}$ (see Figure 12). CENL enables the bus controller to control the current bus cycle. The $\overline{AEN}$ input prevents a bus controller from driving its command outputs. $\overline{AEN}$ HIGH means that another bus controller may be driving the shared bus.

In Figure 12, two buses are shown: a local bus and a MULTIBUS I. Only one bus is used for each CPU bus cycle. The CENL inputs of the bus controller select which bus controller is to perform the bus cycle. An address decoder determines which bus to use for each bus cycle. The 82C288 connected to the shared MULTIBUS I must be selected by CENL and be given access to the MULTIBUS I by $\overline{AEN}$ before it will begin a MULTIBUS I operation.

CENL must be sampled HIGH at the end of the T_S bus state (see waveforms) to enable the bus controller to activate its command and control outputs. If sampled LOW the commands and DEN will not go active and DT/$\overline{R}$ will remain HIGH. The bus controller will ignore the CMDLY, CEN, and $\overline{READY}$ inputs until another bus cycle is started via $\overline{S1}$ and $\overline{S0}$. Since an address decoder is commonly used to identify which bus is required for each bus cycle, CENL is latched to avoid the need for latching its input.

The CENL input can affect the DEN control output. When MB = 0, DEN normally becomes active during Phase 2 of T_S in write bus cycles. This transition occurs before CENL is sampled. If CENL is sampled LOW, the DEN output will be forced LOW during T_C as shown in the timing waveforms.

When MB = 1, CEN/$\overline{AEN}$ becomes $\overline{AEN}$. $\overline{AEN}$ controls when the bus controller command outputs enter and exit 3-state OFF. $\overline{AEN}$ is intended to be driven by a MULTIBUS I type bus arbiter, which assures only one bus controller is driving the shared bus at any time. When $\overline{AEN}$ makes a LOW to HIGH transition, the command outputs immediately enter 3-state OFF and DEN is forced inactive. An inactive DEN should force the local data transceivers connected to the shared data bus into 3-state OFF (see Figure 12). The LOW to HIGH transition of $\overline{AEN}$ should only occur during T_I or T_S bus states.

The HIGH to LOW transition of $\overline{AEN}$ signals that the bus controller may now drive the shared bus command signals. Since a bus cycle may be active or be in the process of starting, $\overline{AEN}$ can become active during any T-state. $\overline{AEN}$ LOW immediately allows DEN to go to the appropriate state. Three CLK edges later, the command outputs will go active (see timing waveforms). The MULTIBUS I requires this delay for the address and data to be valid on the bus before the command becomes active.

When MB = 0, CEN/$\overline{AEN}$ becomes CEN. CEN is an asynchronous input which immediately affects the command and DEN outputs. When CEN makes a HIGH to LOW transition, the commands and DEN

are immediately forced inactive. When CEN makes a LOW to HIGH transition, the commands and DEN outputs immediately go to the appropriate state (see timing waveforms). $\overline{\text{READY}}$ must still become active to terminate a bus cycle if CEN remains LOW for a selected bus controller (CENL was latched HIGH).

Some memory or I/O systems may require more address or write data setup time to command active than provided by the basic command output timing. To provide flexible command timing, the CMDLY input can delay the activation of command outputs. The CMDLY input must be sampled LOW to activate the command outputs. CMDLY does not affect the control outputs ALE, MCE, DEN, and DT/$\overline{\text{R}}$.

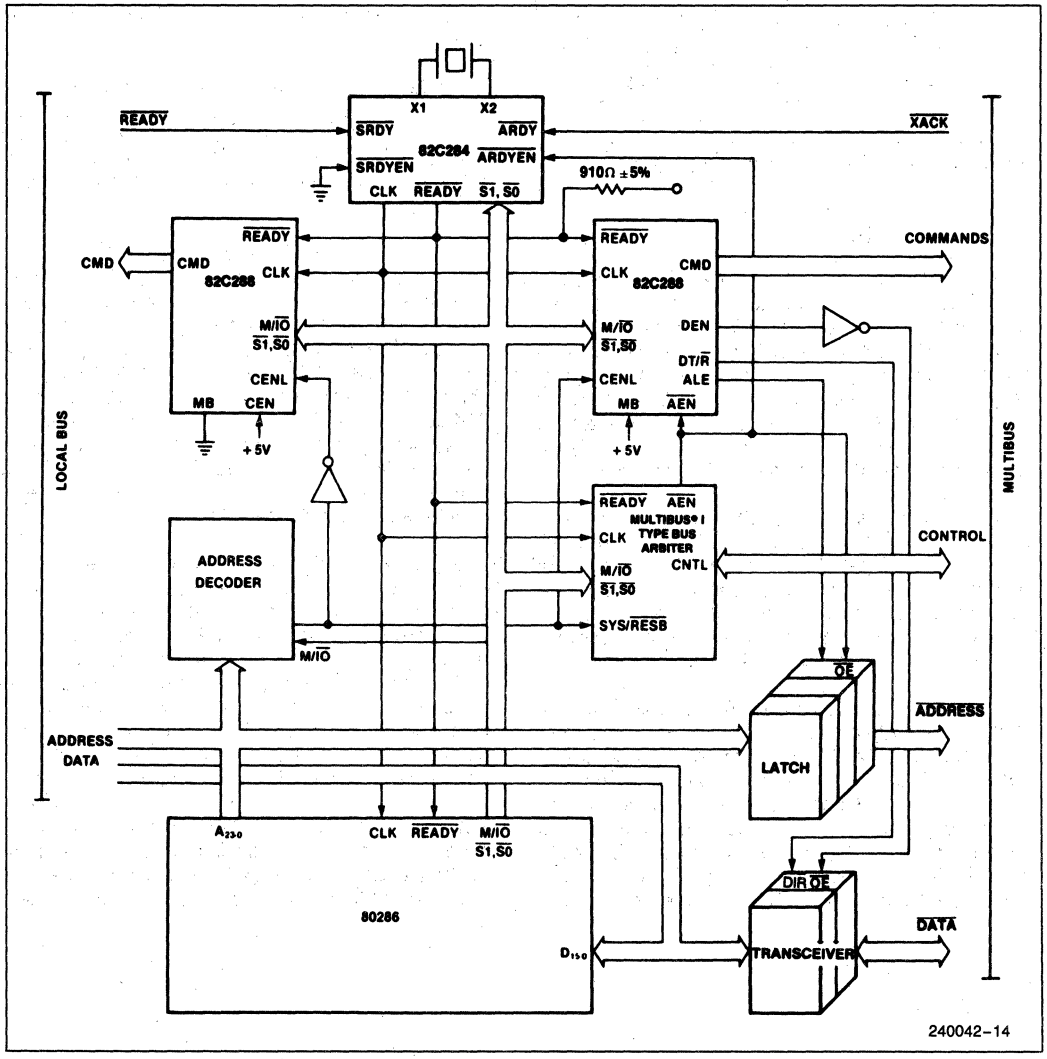


Figure 12. System Use of $\overline{\text{AEN}}$ and CENL

CMDLY is first sampled on the falling edge of the CLK ending T_S. If sampled HIGH, the command output is not activated, and CMDLY is again sampled on the next falling edge of CLK. Once sampled LOW, the proper command output becomes active immediately if MB = 0. If MB = 1, the proper command goes active no earlier than shown in Figures 9 and 10.

$\overline{\text{READY}}$ can terminate a bus cycle before CMDLY allows a command to be issued. In this case no commands are issued an the bus controller will deactivate DEN and DT/$\overline{\text{R}}$ in the same manner as if a command had been issued.

Waveforms Discussion

The waveforms show the timing relationships of inputs and outputs and do not show all possible transitions of all signals in all modes. Instead, all signal timing relationships are shown via the general cases. Special cases are shown when needed. The waveforms provide some functional descriptions of the 82C288; however, most functional descriptions are provided in Figures 5 through 11.

To find the timing specification for a signal transition in a particular mode, first look for a special case in the waveforms. If no special case applies, then use a timing specification for the same or related function in another mode.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
Voltage on Any Pin with Respect to GND	−0.5V to +7V
Power Dissipation	1 Watt

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS V_{CC} = 5V ±5%, T_{CASE} = 0°C to 85°C*

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input LOW Voltage	−0.5	0.8	V	
V_{IH}	Input HIGH Voltage	2.0	V_{CC} + 0.5	V	
V_{ILC}	CLK Input LOW Voltage	−0.5	0.6	V	
V_{IHC}	CLK Input HIGH Voltage	3.8	V_{CC} + 0.5	V	
V_{OL}	Output LOW Voltage				
	Command Outputs		0.45	V	I_{OL} = 32 mA (Note 1)
	Control Outputs		0.45	V	I_{OL} = 16 mA (Note 2)
V_{OH}	Output HIGH Voltage				
	Command Outputs	2.4		V	I_{OH} = −5 mA (Note 1)
		V_{CC} − 0.5		V	I_{OH} = −1 mA (Note 1)
	Control Outputs	2.4		V	I_{OH} = −1 mA (Note 2)
		V_{CC} − 0.5		V	I_{OH} = −0.2 mA (Note 2)
I_{IL}	Input Leakage Current		±10	µA	0V ≤ V_{IN} ≤ V_{CC}
I_{LO}	Output Leakage Current		±10	µA	0.45V ≤ V_{OUT} ≤ V_{CC}
I_{CC}	Power Supply Current		75	mA	
I_{CCS}	Power Supply Current (Static)		1	mA	(Note 3)
C_{CLK}	CLK Input Capacitance		12	pF	F_C = 1 MHz
C_I	Input Capacitance		10	pF	F_C = 1 MHz
C_O	Input/Output Capacitance		20	pF	F_C = 1 MHz

*T_A is guaranteed from 0°C to +70°C as long as T_{CASE} is not exceeded.

NOTES:
1. Command Outputs are $\overline{INTA}$, $\overline{IORC}$, $\overline{IOWC}$, $\overline{MRDC}$ and $\overline{MWRC}$.
2. Control Outputs are DT/$\overline{R}$, DEN, ALE and MCE.
3. Tested while outputs are unloaded, and inputs at V_{CC} or V_{SS}.

A.C. CHARACTERISTICS

V_{CC} = 5V, ±5%, T_{CASE} = 0°C to +85°C.* AC timings are referenced to 0.8V and 2.0V points of signals as illustrated in data sheet waveforms, unless otherwise noted.

Symbol	Parameter	8 MHz (Advance)		10 MHz (Advance)		12.5 MHz (Advance)		Unit	Test Condition
		-8 Min	-8 Max	-10 Min	-10 Min	-12 Min	-12 Max		
1	CLK Period	62	250	50	250	40	250	ns	
2	CLK HIGH Time	20		16		13		ns	at 3.6V
3	CLK LOW Time	15		12		11		ns	at 1.0V
4	CLK Rise Time		10		8		8	ns	1.0V to 3.6V
5	CLK Fall Time		10		8		8	ns	3.6V to 1.0V
6	M/$\overline{IO}$ and Status Setup Time	22		18		15		ns	
7	M/$\overline{IO}$ and Status Hold Time	1		1		1		ns	
8	CENL Setup Time	20		15		15		ns	
9	CENL Hold Time	1		1		1		ns	
10	$\overline{READY}$ Setup Time	38		26		18		ns	
11	$\overline{READY}$ Hold Time	25		25		20		ns	
12	CMDLY Setup Time	20		15		15		ns	
13	CMDLY Hold Time	1		1		1		ns	
14	$\overline{AEN}$ Setup Time	20		15		15		ns	(Note 3)
15	$\overline{AEN}$ Hold Time	0		0		0		ns	(Note 3)
16	ALE, MCE Active Delay from CLK	3	20	3	16	3	16	ns	(Note 4)
17	ALE, MCE Inactive Delay from CLK		25		19		19	ns	(Note 4)
18	DEN (Write) Inactive from CENL		35		23		23	ns	(Note 4)
19	DT/$\overline{R}$ LOW from CLK		25		23		23	ns	(Note 4)
20	DEN (Read) Active$\overline{R}$ from DT/	5	35	5	21	5	21	ns	(Note 4)
21	DEN (Read) Inactive Dly from CLK	3	35	3	21	3	19	ns	(Note 4)
22	DT/$\overline{R}$ HIGH from DEN Inactive	5	35	5	20	5	18	ns	(Note 4)
23	DEN (Write) Active Delay from CLK		30		23		23	ns	(Note 4)
24	DEN (Write) Inactive Dly from CLK	3	30	3	19	3	19	ns	(Note 4)

*T_A is guaranteed from 0°C to +70°C as long as T_{CASE} is not exceeded.

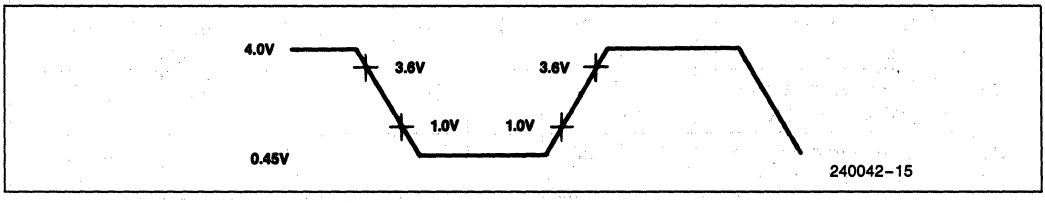
A.C. CHARACTERISTICS

V_{CC} = 5V, ±5%, T_{CASE} = 0°C to +85°C.* AC timings are referenced to 0.8V and 2.0V points of signals as illustrated in data sheet waveforms, unless otherwise noted. (Continued)

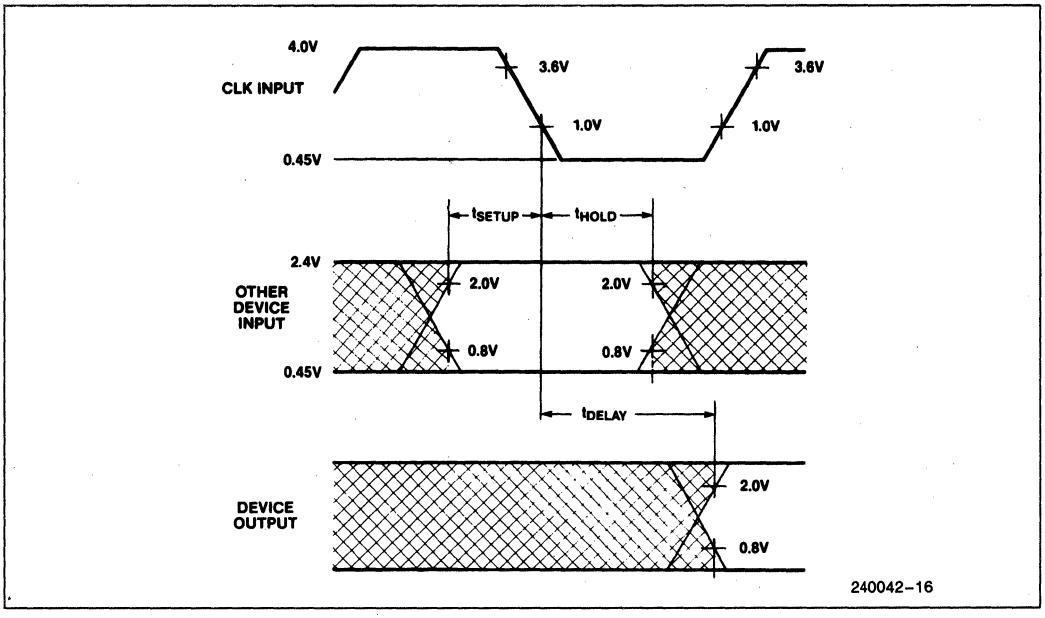
Symbol	Parameter	8 MHz (Advance)		10 MHz (Advance)		12.5 MHz (Advance)		Unit	Test Condition
		-8 Min	-8 Max	-10 Min	-10 Min	-12 Min	-12 Max		
25	DEN Inactive from CEN		30		25		25	ns	(Note 4)
26	DEN Active from CEN		30		24		24	ns	(Note 4)
27	DT/$\overline{R}$ HIGH from CLK (when CEN = LOW)		35		25		25	ns	(Note 4)
28	DEN Active from $\overline{AEN}$		30		26		26	ns	(Note 4)
29	$\overline{CMD}$ Active Delay from CLK	3	25	3	21	3	21	ns	(Note 5)
30	$\overline{CMD}$ Inactive Delay from CLK	5	20	5	20	5	20	ns	(Note 5)
31	$\overline{CMD}$ Active from CEN		25		25		25	ns	(Note 5)
32	$\overline{CMD}$ Inactive from CEN		25		25		25	ns	(Note 5)
33	$\overline{CMD}$ Inactive Enable from $\overline{AEN}$		40		40		40	ns	(Note 5)
34	$\overline{CMD}$ Float Delay from $\overline{AEN}$		40		40		40	ns	(Note 6)
35	MB Setup Time	20		20		20		ns	
36	MB Hold Time	0		0		0		ns	
37	Command Inactive Enable from MB ↓		40		40		40	ns	(Note 5)
38	Command Float Time from MB ↑		40		40		40	ns	(Note 6)
39	DEN Inactive from MB ↑		30		26		26	ns	(Note 4)
40	DEN Active from MB ↓		30		30		30	ns	(Note 4)

*T_A is guaranteed from 0°C to +70°C as long as T_{CASE} is not exceeded.

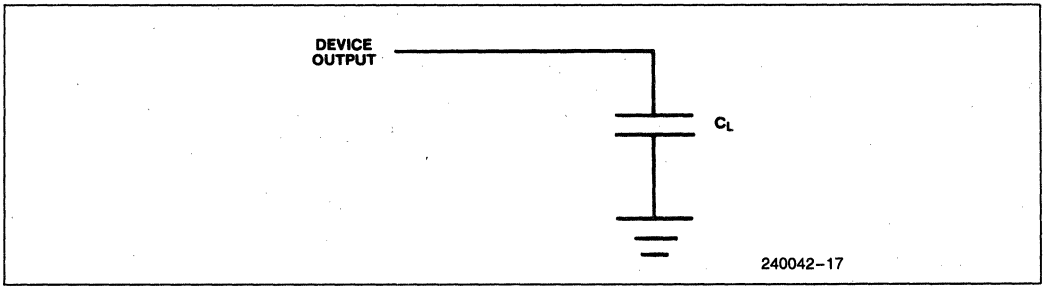
NOTES:
3. $\overline{AEN}$ is an asynchronous input. This specification is for testing purposes only, to assure recognition at a specific CLK edge.
4. Control output load: Cl = 150 pF.
5. Command output load: Cl = 300 pF.
6. Float condition occurs when output current is less than I_{LO} in magnitude.



Note 7: AC Drive and Measurement Points—CLK Input



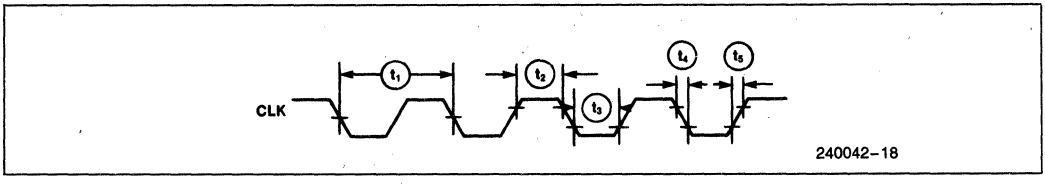
Note 8: AC Setup, Hold and Delay Time Measurement—General



Note 9: AC Test Loading on Outputs

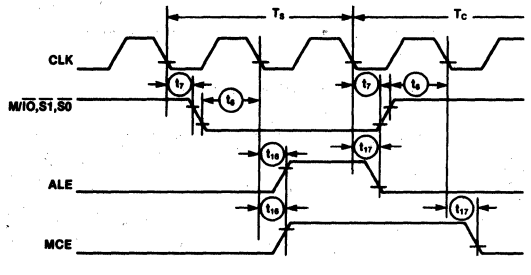
WAVEFORMS

CLK CHARACTERISTICS

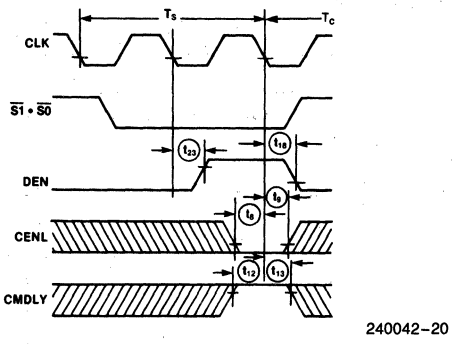


WAVEFORMS (Continued)

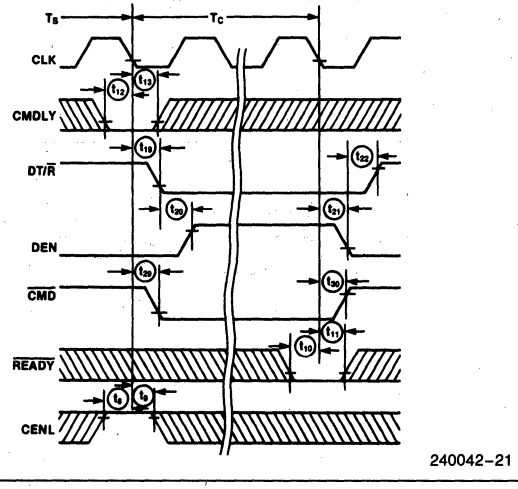
STATUS, ALE, MCE, CHARACTERISTICS



CENL, CMDLY, DEN CHARACTERISTICS WITH MB = 0 AND CEN = 1 DURING WRITE CYCLE

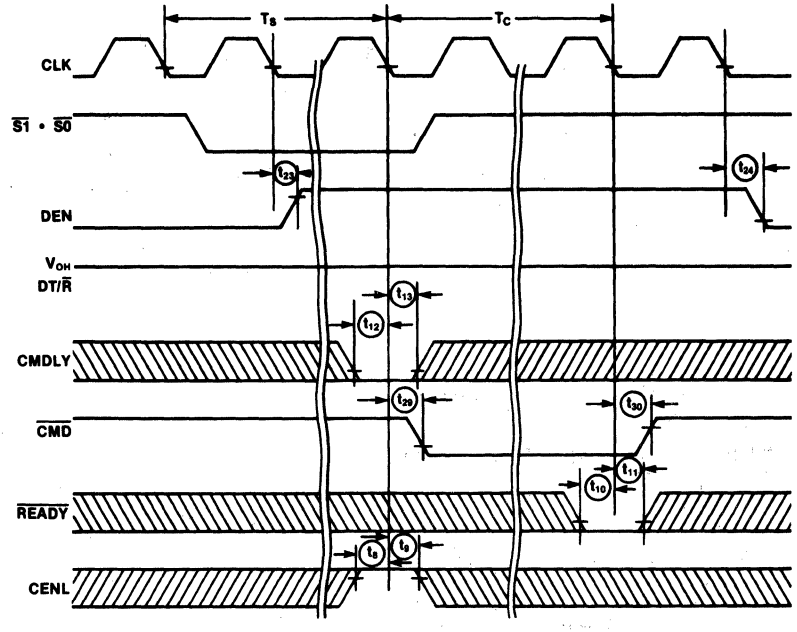


READ CYCLE CHARACTERISTICS WITH MB = 0 AND CEN = 1



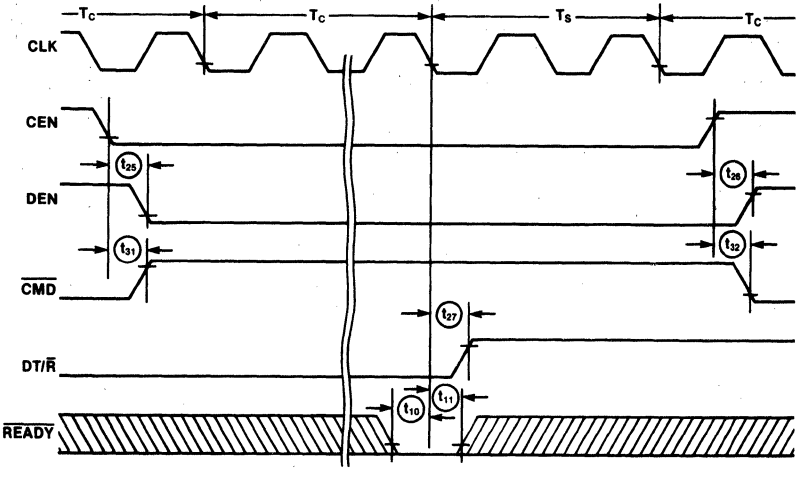
WAVEFORMS (Continued)

WRITE CYCLE CHARACTERISTIC WITH MB = 0 AND CEN = 1



240042-22

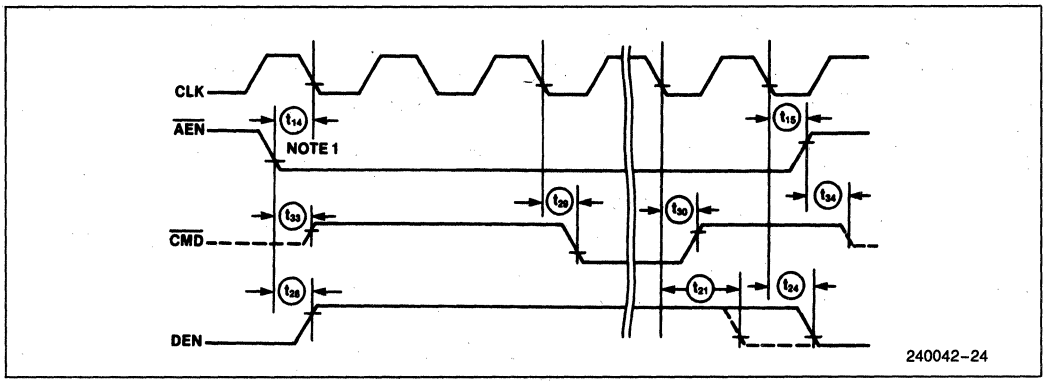
CEN CHARACTERISTICS WITH MB = 0



240042-23

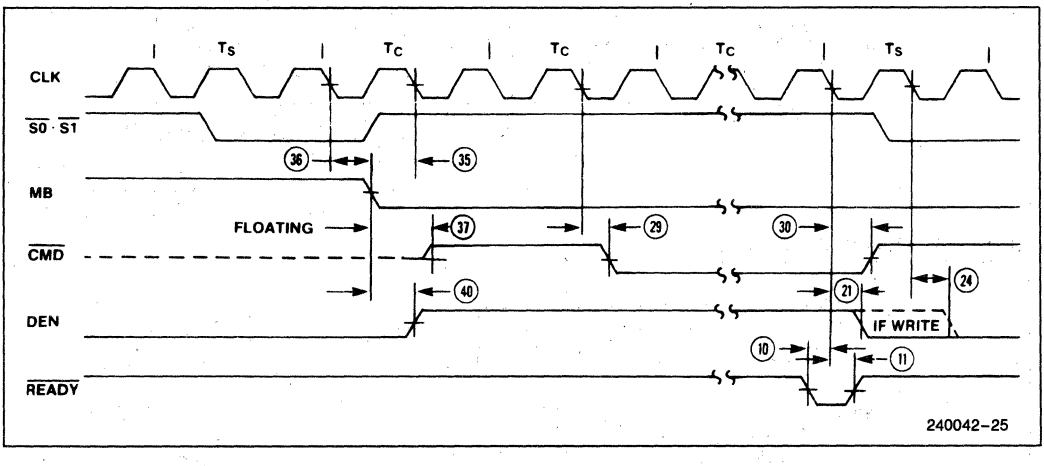
WAVEFORMS (Continued)

$\overline{AEN}$ CHARACTERISTICS WITH MB = 1



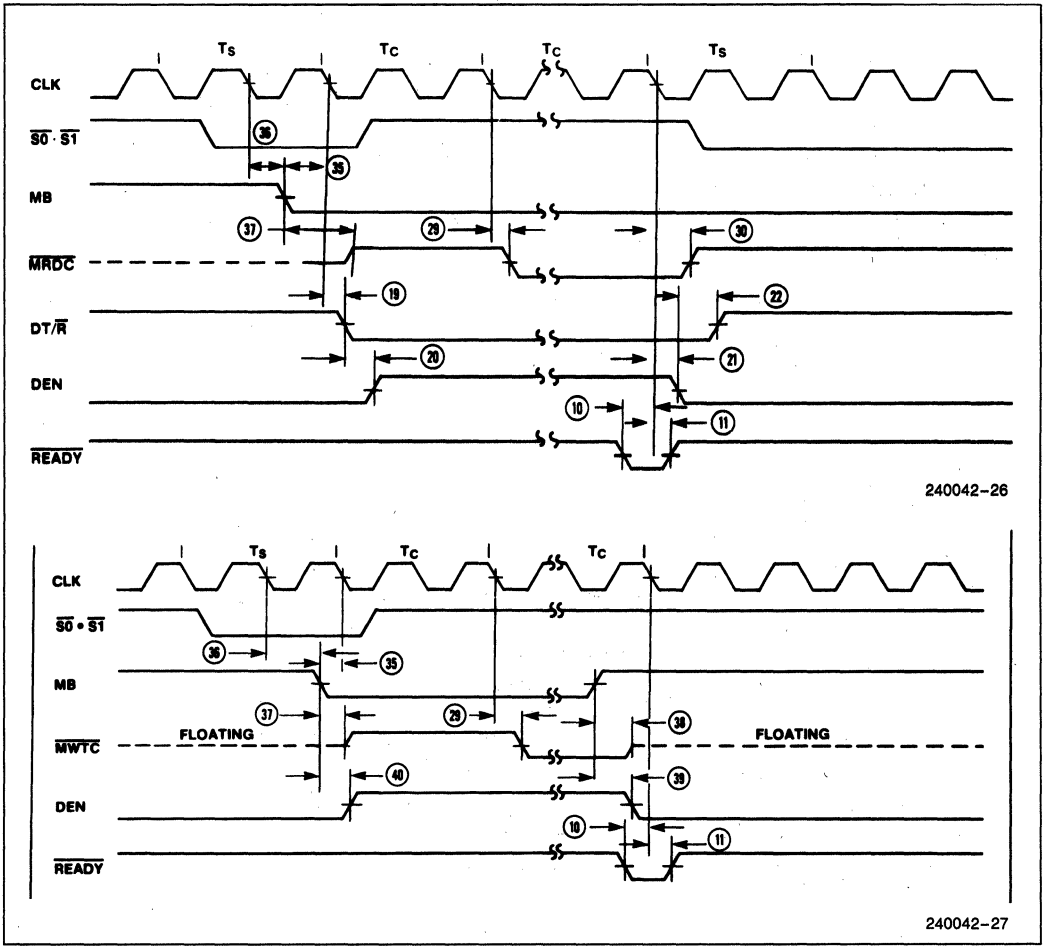
NOTE:
1. $\overline{AEN}$ is an asynchronous input. $\overline{AEN}$ setup and hold time is specified to guarantee the response shown in the waveforms.

MB CHARACTERISTICS WITH $\overline{AEN}$/CEN = HIGH



WAVEFORMS (Continued)

MB CHARACTERISTICS WITH $\overline{AEN}$/CEN = HIGH (Continued)



NOTES:
1. MB is an asynchronous input. MB setup and hold times specified to guarantee the response shown in the waveforms.
2. If the setup time, t35, is met two clock cycles will occur before $\overline{CMD}$ becomes active after the falling edge of MB.

DATA SHEET REVISION REVIEW

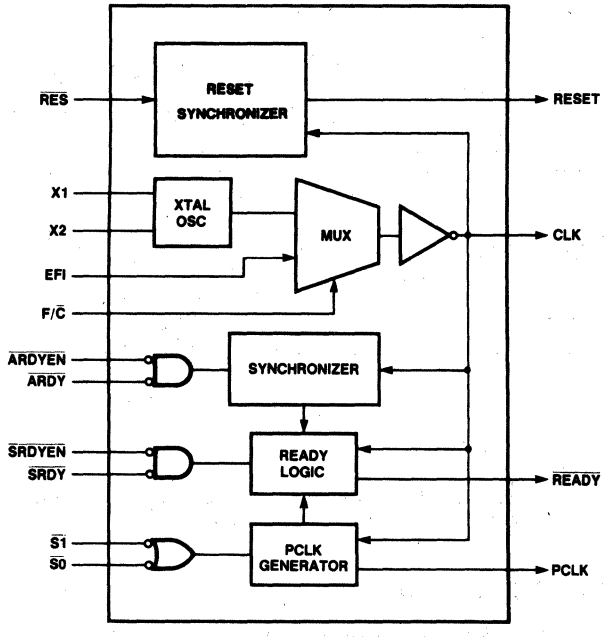
The following list represents key differences between this and the -001 data sheet. Please review this summary carefully.

1. The 82C288 data sheet has been upgraded from "ADVANCED" to "PRELIMINARY".

82C284
CLOCK GENERATOR AND READY INTERFACE FOR 80286 PROCESSORS (82C284-12, 82C284-10, 82C284-8)

- **Generates System Clock for 80286 Processors**
- **Uses Crystal or TTL Signal for Frequency Source**
- **Provides Local READY and MULTIBUS®I READY Synchronization**
- **Single +5V Power Supply**
- **CHMOS III Technology**
- **Generates System Reset Output**
- **Available in 18-Lead Cerdip and 20-Pin PLCC (Plastic Leaded Chip Carrier) Packages**
  (See Packaging Spec, Order #231369)

The 82C284 is a clock generator/driver which provides clock signals for 80286 processors and support components. It also contains logic to supply $\overline{\text{READY}}$ to the CPU from either asynchronous or synchronous sources and synchronous RESET from an asynchronous input.

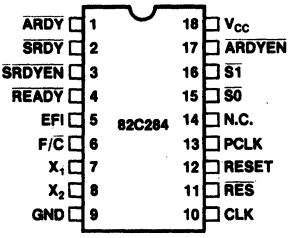


210453-1

Figure 1. 82C284 Block Diagram

September 1988
Order Number: 210453-009

18-Lead Cerdip

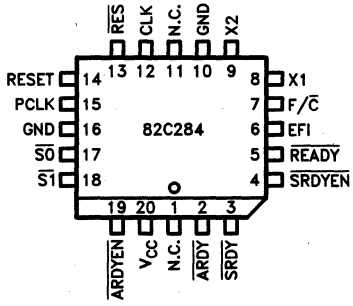


210453-2

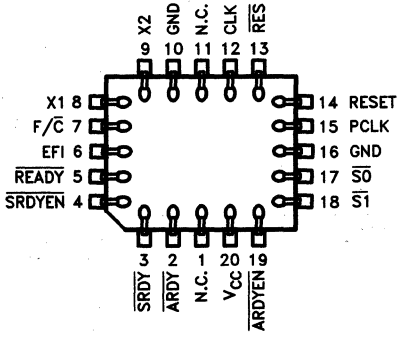
P.C. Board Views—As viewed from the component side of the P.C. Board.

Component Pad Views—As viewed from underside of component when mounted on the board.

20 Pin PLCC



210453-18



210453-19

NOTE:
1. N.C. Signals must not be connected.

Figure 2. 82C284 Pin Configuration

Table 1. Pin Description

The following pin function descriptions are for the 82C284 clock generator.

Symbol	Type	Name and Function
CLK	O	**SYSTEM CLOCK** is the signal used by the processor and support devices which must be synchronous with the processor. The frequency of the CLK output has twice the desired internal processor clock frequency. CLK can drive both TTL and MOS level inputs.
F/$\overline{C}$	I	**FREQUENCY/CRYSTAL SELECT** is a strapping option to select the source for the CLK output. When F/$\overline{C}$ is strapped LOW, the internal crystal oscillator drives CLK. When F/$\overline{C}$ is strapped HIGH, the EFI input drives the CLK output.
X1, X2	I	**CRYSTAL IN** are the pins to which a parallel resonant fundamental mode crystal is attached for the internal oscillator. When F/$\overline{C}$ is LOW, the internal oscillator will drive the CLK output at the crystal frequency. The crystal frequency must be twice the desired internal processor clock frequency.
EFI	I	**EXTERNAL FREQUENCY IN** drives CLK when the F/$\overline{C}$ input is strapped HIGH. The EFI input frequency must be twice the desired internal processor clock frequency.
PCLK	O	**PERIPHERAL CLOCK** is an output which provides a 50% duty cycle clock with 1/2 the frequency of CLK. PCLK will be in phase with the internal processor clock following the first bus cycle after the processor has been reset.
$\overline{ARDYEN}$	I	**ASYNCHRONOUS READY ENABLE** is an active LOW input which qualifies the $\overline{ARDY}$ input. $\overline{ARDYEN}$ selects $\overline{ARDY}$ as the source of ready for the current bus cycle. Inputs to $\overline{ARDYEN}$ may be applied asynchronously to CLK. Setup and hold times are given to assure a guaranteed response to synchronous inputs.
$\overline{ARDY}$	I	**ASYNCHRONOUS READY** is an active LOW input used to terminate the current bus cycle. The $\overline{ARDY}$ input is qualified by $\overline{ARDYEN}$. Inputs to $\overline{ARDY}$ may be applied asynchronously to CLK. Setup and hold times are given to assure a guaranteed response to synchronous outputs.
$\overline{SRDYEN}$	I	**SYNCHRONOUS READY ENABLE** is an active LOW input which qualifies $\overline{SRDY}$. $\overline{SRDYEN}$ selects $\overline{SRDY}$ as the source for $\overline{READY}$ to the CPU for the current bus cycle. Setup and hold times must be satisfied for proper operation.
$\overline{SRDY}$	I	**SYNCHRONOUS READY** is an active LOW input used to terminate the current bus cycle. The $\overline{SRDY}$ input is qualified by the $\overline{SRDYEN}$ input. Setup and hold times must be satisfied for proper operation.
$\overline{READY}$	O	**READY** is an active LOW output which signals the current bus cycle is to be completed. The $\overline{SRDY}$, $\overline{SRDYEN}$, $\overline{ARDY}$, $\overline{ARDYEN}$, $\overline{S1}$, $\overline{S0}$ and $\overline{RES}$ inputs control $\overline{READY}$ as explained later in the $\overline{READY}$ generator section. $\overline{READY}$ is an open drain output requiring an external pull-up resistor.

Table 1. Pin Description (Continued)

The following pin function descriptions are for the 82C284 clock generator.

Symbol	Type	Name and Function
$\overline{S0}$, $\overline{S1}$	I	**STATUS** input prepare the 82C284 for a subsequent bus cycle. $\overline{S0}$ and $\overline{S1}$ synchronize PCLK to the internal processor clock and control $\overline{READY}$. These inputs have internal pull-up resistors to keep them HIGH if nothing is driving them. Setup and hold times must be satisfied for proper operation.
RESET	O	**RESET** is an active HIGH output which is derived from the $\overline{RES}$ input. RESET is used to force the system into an initial state. When RESET is active, $\overline{READY}$ will be active (LOW).
$\overline{RES}$	I	**RESET IN** is an active LOW input which generates the system reset signal, RESET. Signals to $\overline{RES}$ may be applied asynchronously to CLK. Setup and hold times are given to assure a guaranteed response to synchronous inputs.
V_{CC}		**SYSTEM POWER:** +5V Power Supply
GND		**SYSTEM GROUND:** 0V

FUNCTIONAL DESCRIPTION

Introduction

The 82C284 generates the clock, ready, and reset signals required for 80286 processors and support components. The 82C284 is packaged in an 18-pin DIP and contains a crystal controlled oscillator, clock generator, peripheral clock generator, Multibus ready synchronization logic and system reset generation logic.

Clock Generator

The CLK output provides the basic timing control for an 80286 system. CLK has output characteristics sufficient to drive MOS devices. CLK is generated by either an internal crystal oscillator or an external source as selected by the F/$\overline{C}$ strapping option. When F/$\overline{C}$ is LOW, the crystal oscillator drives the CLK output. When F/$\overline{C}$ is HIGH, the EFI input drives the CLK output.

The 82C284 provides a second clock output, PCLK, for peripheral devices. PCLK is CLK divided by two. PCLK has a duty cycle of 50% and MOS output drive characteristics. PCLK is normally synchronized to the internal processor clock.

After reset, the PCLK signal may be out of phase with the internal processor clock. The $\overline{S1}$ and $\overline{S0}$ signals of the first bus cycle are used to synchronize PCLK to the internal processor clock. The phase of the PCLK output changes by extending its HIGH time beyond one system clock (see waveforms). PCLK is forced HIGH whenever either $\overline{S0}$ or $\overline{S1}$ were active (LOW) for the two previous CLK cycles. PCLK continues to oscillate when both $\overline{S0}$ and $\overline{S1}$ are HIGH.

Since the phase of the internal processor clock will not change except during reset, the phase of PCLK will not change except during the first bus cycle after reset.

Oscillator

The oscillator circuit of the 82C284 is a linear Pierce oscillator which requires an external parallel resonant, fundamental mode, crystal. The output of the oscillator is internally buffered. The crystal frequency chosen should be twice the required internal processor clock frequency. The crystal should have a typical load capacitance of 32 pF.

X1 and X2 are the oscillator crystal connections. For stable operation of the oscillator, two loading capacitors are recommended, as shown in Table 2. The sum of the board capacitance and loading capacitance should equal the values shown. It is advisable to limit stray board capacitances (not including the effect of the loading capacitors or crystal capacitance) to less than 10 pF between the X1 and X2 pins. Decouple V_{CC} and GND as close to the 82C284 as possible.

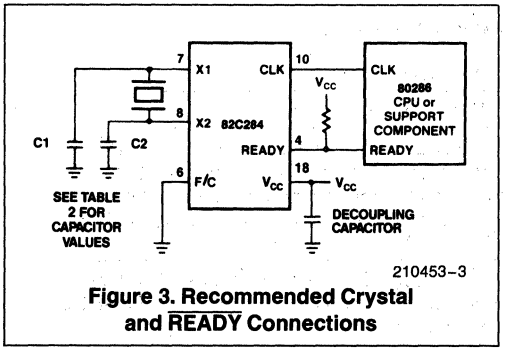


Figure 3. Recommended Crystal and $\overline{\text{READY}}$ Connections

CLK Termination

Due to the CLK output having a very fast rise and fall time, it is recommended to properly terminate the CLK line at frequencies above 10 MHz to avoid signal reflections and ringing. Termination is accomplished by inserting a small resistor (typically 10Ω–74Ω) in series with the output, as shown in Figure 4. This is known as series termination. The resistor value plus the circuit output impedance should be made equal to the impedance of the transmission line.

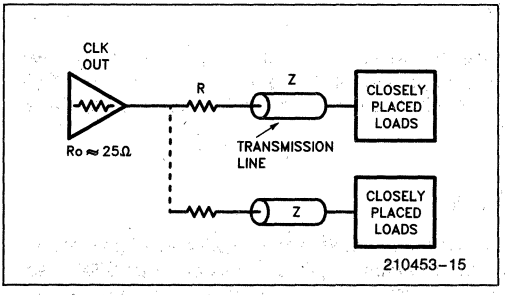


Figure 4. Series Termination

Reset Operation

The reset logic provides the RESET output to force the system into a known, initial state. When the $\overline{\text{RES}}$ input is active (LOW), the RESET output becomes active (HIGH). $\overline{\text{RES}}$ is synchronized internally at the falling edge of CLK before generating the RESET output (see waveforms). Synchronization of the $\overline{\text{RES}}$ input introduces a one or two CLK delay before affecting the RESET output.

At power up, a system does not have a stable V_{CC} and CLK. To prevent spurious activity, $\overline{\text{RES}}$ should be asserted until V_{CC} and CLK stabilize at their operating values. 80286 processors and support components also require their RESET inputs be HIGH a minimum of 16 CLK cycles. A network such as shown in Figure 5 will keep $\overline{\text{RES}}$ LOW long enough to satisfy both needs.

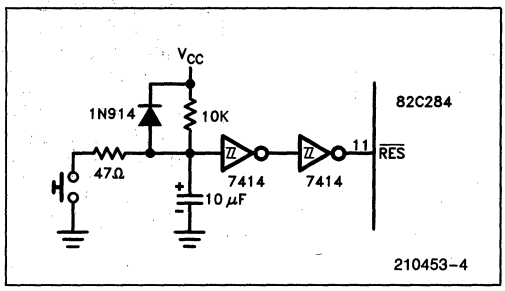


Figure 5. Typical $\overline{\text{RES}}$ Timing Circuit

Ready Operation

The 82C284 accepts two ready sources for the system ready signal which terminates the current bus cycle. Either a synchronous ($\overline{\text{SRDY}}$) or asynchronous ready ($\overline{\text{ARDY}}$) source may be used. Each ready input has an enable ($\overline{\text{SRDYEN}}$ and $\overline{\text{ARDYEN}}$) for selecting the type of ready source required to terminate the current bus cycle. An address decoder would normally select one of the enable inputs.

$\overline{\text{READY}}$ is enabled (LOW), if either $\overline{\text{SRDY}}$ + $\overline{\text{SRDYEN}}$ = 0 or $\overline{\text{ARDY}}$ + $\overline{\text{ARDYEN}}$ = 0 when sampled by the 82C284 $\overline{\text{READY}}$ generation logic. $\overline{\text{READY}}$ will remain active for at least two CLK cycles.

The $\overline{\text{READY}}$ output has an open-drain driver allowing other ready circuits to be wire or'ed with it, as shown in Figure 3. The $\overline{\text{READY}}$ signal of an 80286 system requires an external pull-up resistor. To force the $\overline{\text{READY}}$ signal inactive (HIGH) at the start of a bus cycle, the $\overline{\text{READY}}$ output floats when either $\overline{\text{S1}}$ or $\overline{\text{S0}}$ are sampled LOW at the falling edge of CLK. Two system clock periods are allowed for the pull-up resistor to pull the $\overline{\text{READY}}$ signal to V_{IH}. When RESET is active, $\overline{\text{READY}}$ is forced active one CLK later (see waveforms).

Figure 6 illustrates the operation of $\overline{\text{SRDY}}$ and $\overline{\text{SRDYEN}}$. These inputs are sampled on the falling edge of CLK when $\overline{\text{S1}}$ and $\overline{\text{S0}}$ are inactive and PCLK

is HIGH. $\overline{READY}$ is forced active when both $\overline{SRDY}$ and $\overline{SRDYEN}$ are sampled as LOW.

Figure 7 shows the operation of $\overline{ARDY}$ and $\overline{ARDYEN}$. These inputs are sampled by an internal synchronizer at each falling edge of CLK. The output of the synchronizer is then sampled when PCLK is HIGH. If the synchronizer resolved both the $\overline{ARDY}$ and $\overline{ARDYEN}$ as active, the $\overline{SRDY}$ and $\overline{SRDYEN}$ inputs are ignored. Either $\overline{ARDY}$ or $\overline{ARDYEN}$ must be HIGH at the end of T_S (see Figure 7).

$\overline{READY}$ remains active until either $\overline{S1}$ or $\overline{S0}$ are sampled LOW, or the ready inputs are sampled as inactive.

Table 2. 82C284 Crystal Loading Capacitance Values

Crystal Frequency	C1 Capacitance (Pin 7)	C2 Capacitance (Pin 8)
1 to 8 MHz	60 pF	40 pF
8 to 20 MHz	25 pF	15 pF
Above 20 MHz	15 pF	15 pF

NOTE:
Capacitance values must include stray board capacitance.

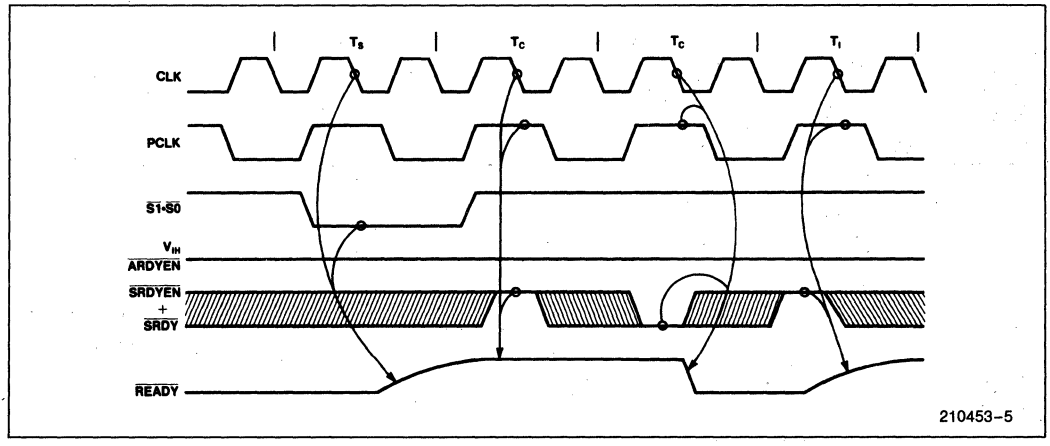


Figure 6. Synchronous Ready Operation

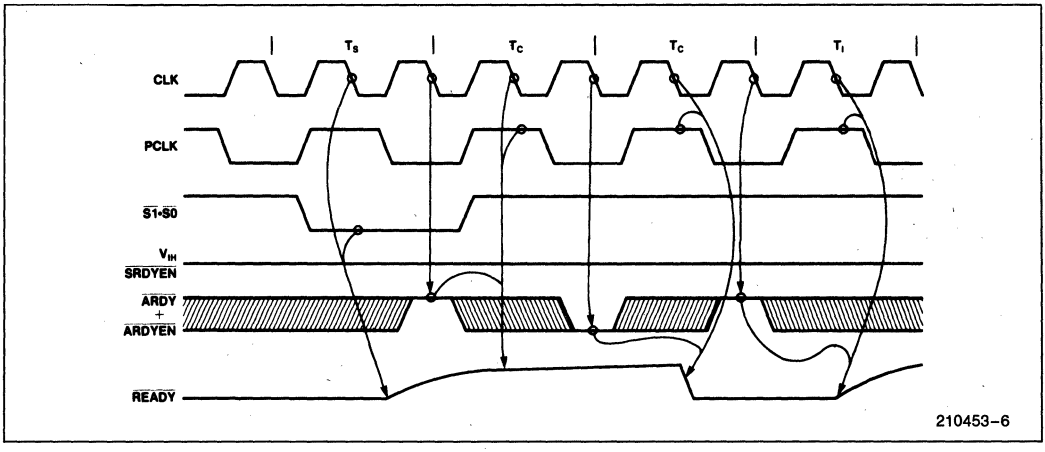


Figure 7. Asynchronous Ready Operation

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias .....0°C to +70°C
Storage Temperature .....−65°C to +150°C
All Output and Supply Voltages .....−0.5V to +7V
All Input Voltages.....−1.0V to +5.5V
Power Dissipation .....1 Watt

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS T_{CASE} = 0°C to +85°C,* V_{CC} = 5V ±5%

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{IL}	Input LOW Voltage		0.8	V	
V_{IH}	Input HIGH Voltage	2.0		V	
V_{IHR}	$\overline{RES}$ and EFI Input HIGH Voltage	2.6		V	
V_{OL}	RESET, PCLK Output LOW Voltage		0.45	V	I_{OL} = 5 mA
V_{OH}	RESET, PCLK Output HIGH Voltage	2.4		V	I_{OH} = −1 mA
		V_{CC}−0.5		V	I_{OH} = −0.2 mA
V_{OLR}	$\overline{READY}$, Output LOW Voltage		0.45	V	I_{OL} = 9 mA
V_{OLC}	CLK Output LOW Voltage		0.45	V	I_{OL} = 5 mA
V_{OHC}	CLK Output HIGH Voltage	4.0		V	I_{OH} = − 800 μA
I_{IL}	Input Sustaining Current on $\overline{S0}$ and $\overline{S1}$ Pins	30	500	μA	V_{IN} = 0V
I_{LI}	Input Leakage Current		±10	μA	$0 \le V_{IN} \le V_{CC}$(1)
I_{CC}	Power Supply Current		75	mA	at 25 MHz Output CLK Frequency
C_I	Input Capacitance		10	pF	F_C = 1 MHz

*T_A is guaranteed from 0°C to +70°C as long as T_{CASE} is not exceeded.

NOTE:
1. Status lines S0 and S1 excluded because they have internal pull-up resistors.

A.C. CHARACTERISTICS V_{CC} = 5V ±5%, T_{CASE} = 0°C to +85°C.*

Timings are referenced to 0.8V and 2.0V points of signals as illustrated in the datasheet waveforms, unless otherwise noted.

82C284 A.C. Timing Parameters

Symbol	Parameter	8.0 MHz Preliminary		10.0 MHz Preliminary		12.5 MHz Preliminary		Units	Test Conditions
		Min	Max	Min	Max	Min	Max		
1	EFI to CLK Delay		25		25		25	ns	At 1.5V (1)
2	EFI LOW Time	28		22.5		13		ns	At 1.5V (1, 7)
3	EFI HIGH Time	28		22.5		22		ns	At 1.5V (1, 7)
4	CLK Period	62	500	50	500	40	500	ns	
5	CLK LOW Time	15		12		11		ns	At 1.0V (1, 2, 7, 8, 9, 10)
6	CLK HIGH Time	25		16		13		ns	At 3.6V (1, 2, 7, 8, 9, 10)
7	CLK Rise Time		10		8		8	ns	1.0V to 3.6V (1, 2, 10, 11)
8	CLK Fall Time		10		8		8	ns	3.6V to 1.0V (1, 9, 10, 11)
9	Status Setup Time	22		—		—		ns	(Note 1)
9a	Status Setup Time for Status Going Active	—		20		22		ns	(Note 1)
9b	Status Setup Time for Status Going Inactive	—		20		18		ns	(Note 1)
10	Status Hold Time	1		1		3		ns	(Note 1)
11	$\overline{SRDY}$ or $\overline{SRDYEN}$ Setup Time	17		15		15		ns	(Note 1)
12	$\overline{SRDY}$ or $\overline{SRDYEN}$ Hold Time	0		2		2		ns	(Notes 1, 11)
13	$\overline{ARDY}$ or $\overline{ARDYEN}$ Setup Time	0		0		0		ns	(Notes 1, 3)
14	$\overline{ARDY}$ or $\overline{ARDYEN}$ Hold Time	30		30		25		ns	(Notes 1, 3)
15	$\overline{RES}$ Setup Time	20		20		18		ns	(Notes 1, 3)
16	$\overline{RES}$ Hold Time	10		10		8		ns	(Notes 1, 3)
17	$\overline{READY}$ Inactive Delay	5		5		5		ns	At 0.8V (4)
18	$\overline{READY}$ Active Delay	0	24	0	24	0	18	ns	At 0.8V (4)
19	PCLK Delay	0	45	0	35	0	23	ns	(Note 5)
20	RESET Delay	5	34	5	27	3	22	ns	(Note 5)
21	PCLK LOW Time	t4–20		t4–20		T4–20		ns	(Notes 5, 6)
22	PCLK HIGH Time	t4–20		t4–20		T4–20		ns	(Notes 5, 6)

*T_A is guaranteed from 0°C to 70°C as long as T_{CASE} is not exceeded.

NOTES:

1. CLK loading: C_L = 100 pF. The 82C284's X1 and X2 inputs are designed primarily for parallel-resonant crystals. Serial-resonant crystals may also be used, however, they may oscillate up to 0.01% faster than their nominal frequencies when used with the 82C284. For either type of crystal, capacitive loading should be as specified by Table 2.
2. With the internal crystal oscillator using recommended crystal and capacitive loading; or with the EFI input meeting specifications t2 and t3. The recommended crystal loading for CLK frequencies of 8 MHz–20 MHz are 25 pF from pin X_1 to ground, and 15 pF from pin X_2 to ground; for CLK frequencies above 20 MHz 15 pF from pin X_1 to ground, and 15 pF from pin X_2 to ground. These recommended values are ±5 pF and include all stray capacitance. Decouple V_{CC} and GND as close to the 82C284 as possible.
3. This is an asynchronous input. This specification is given for testing purposes only, to assure recognition at specific CLK edge.

NOTES:

4. Pull-up Resistor values for READY Pin:

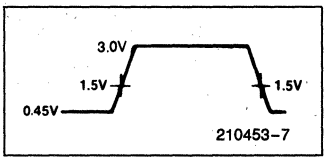
CPU Frequency	8 MHz	10 MHz	12.5 MHz
Resistor	910Ω	700Ω	600Ω
CL	150 pF	150 pF	150 pF
I_{OL}	7 mA	7 mA	9 mA

5. PCLK and RESET loading: C_L = 75 pF.
6. t4 refers to any allowable CLK period.
7. When driving the 82C284 with EFI, provide minimum EFI HIGH and LOW times as follows:

CLK Output Frequency	16 MHz	20 MHz	25 MHz
Min. Required EFI HIGH Time	28 ns	22.5 ns	22 ns
Min. Required EFI LOW Time	28 ns	22.5 ns	13 ns

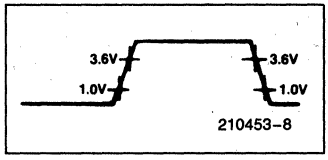
8. When using a crystal (with recommended capacitive loading per Table 2) appropriate for the speed of the 80286, CLK output HIGH and LOW times guaranteed to meet the 80286 requirements.

Reset Drive EFI Drive and Measurement Points



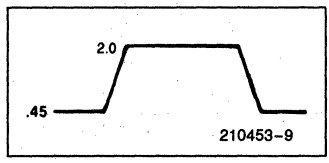
Note 9

CLK Output Measurement Points

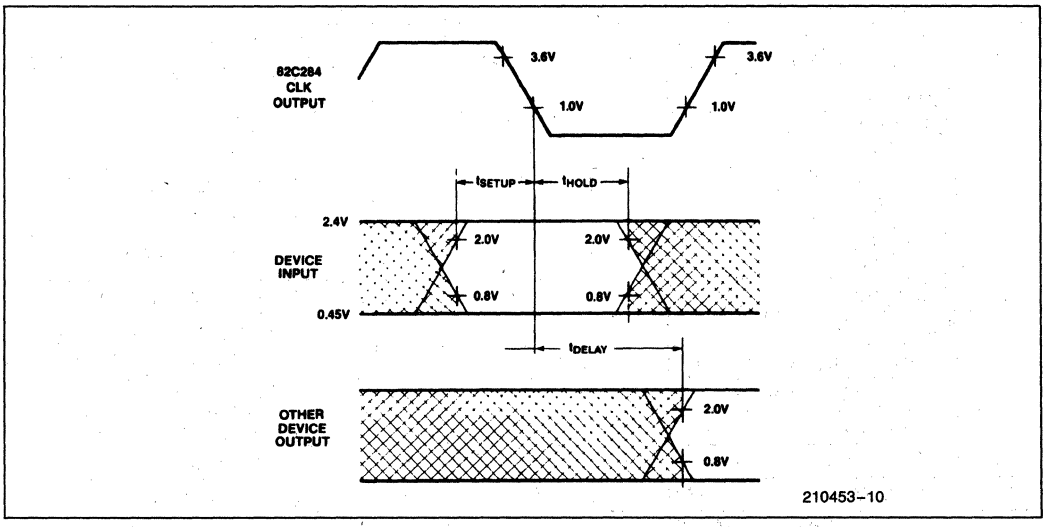


Note 10

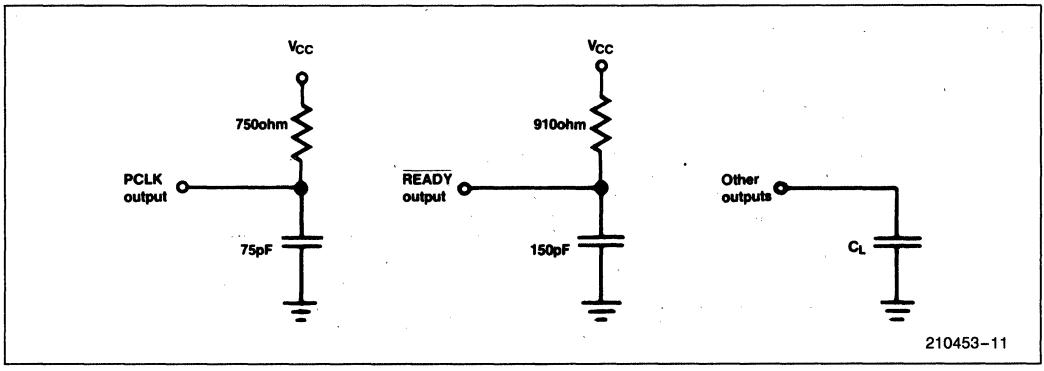
F/$\overline{C}$ Drive Points



Note 11



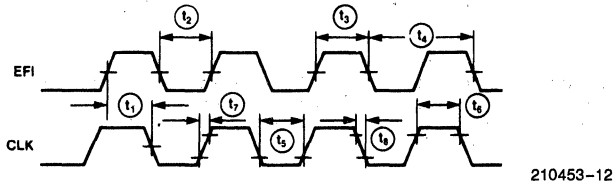
Note 12. AC Setup, Hold and Delay Time Measurement—General



Note 13. AC Test Loading on Outputs

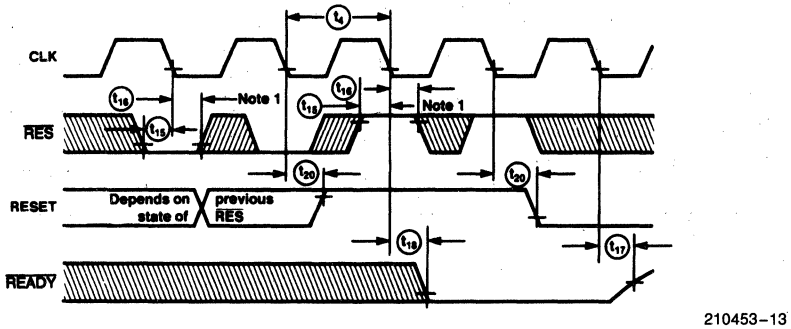
WAVEFORMS

CLK as a Function of EFI



NOTE:
The EFI input LOW and HIGH times as shown are required to guarantee the CLK LOW and HIGH times shown.

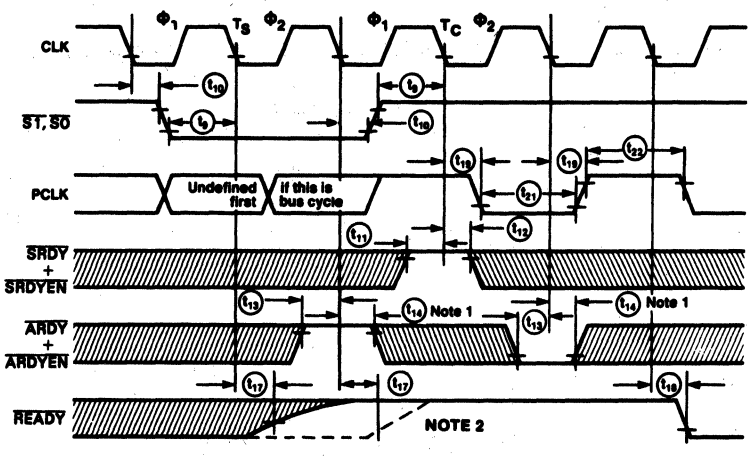
RESET and $\overline{\text{READY}}$ Timing as a Function of $\overline{\text{RES}}$ with $\overline{\text{S1}}$, $\overline{\text{S0}}$, $\overline{\text{ARDY}}$ + $\overline{\text{ARDYEN}}$, and $\overline{\text{SRDY}}$ + $\overline{\text{SRDYEN}}$ High



NOTE:
1. This is an asynchronous input. The setup and hold times shown are required to guarantee the response shown.

WAVEFORMS (Continued)

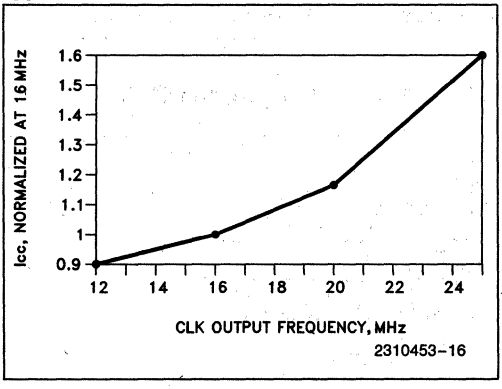
$\overline{\text{READY}}$ and PCLK Timing with $\overline{\text{RES}}$ High



210453-14

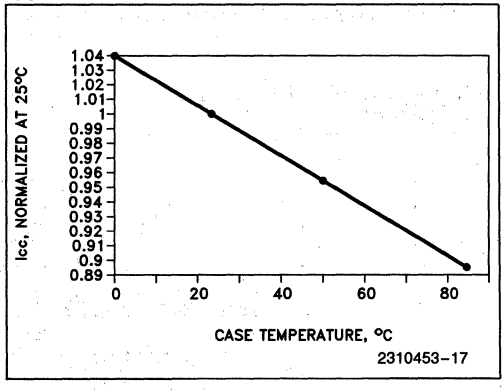
NOTES:
1. This is an asynchronous input. The setup and hold times shown are required to guarantee the response shown.
2. If $\overline{\text{SRDY}}$ + $\overline{\text{SRDYEN}}$ or $\overline{\text{ARDY}}$ + $\overline{\text{ARDYEN}}$ are active before and/or during the first bus cycle after RESET, $\overline{\text{READY}}$ may not be deasserted until after the falling edge of $\phi 2$ of T_S.

I_{CC} vs Frequency @ Nominal Conditions



2310453-16

I_{CC} vs Case Temperature @ 25 MHz



2310453-17

DATA SHEET REVISION REVIEW

The following list represents key differences between this and the -008 data sheet. Please review this summary carefully.

1. The "PRELIMINARY" markings have been removed from the data sheet.

INTEL386™ Family

4

386™ MICROPROCESSOR
HIGH PERFORMANCE 32-BIT CHMOS MICROPROCESSOR WITH INTEGRATED MEMORY MANAGEMENT

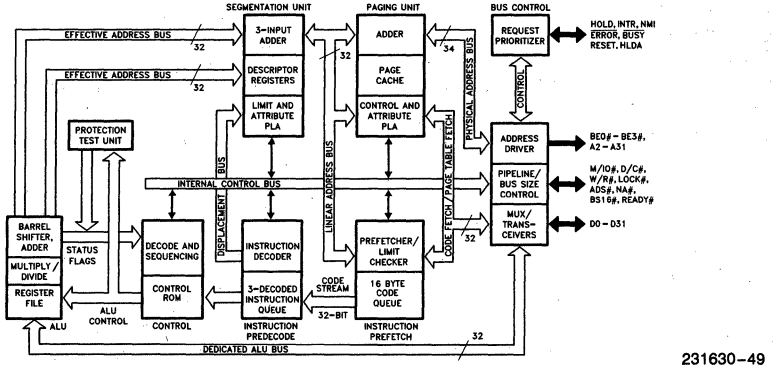
- **Flexible 32-Bit Microprocessor**
  - **— 8, 16, 32-Bit Data Types**
  - **— 8 General Purpose 32-Bit Registers**
- **Very Large Address Space**
  - **— 4 Gigabyte Physical**
  - **— 64 Terabyte Virtual**
  - **— 4 Gigabyte Maximum Segment Size**
- **Integrated Memory Management Unit**
  - **— Virtual Memory Support**
  - **— Optional On-Chip Paging**
  - **— 4 Levels of Protection**
  - **— Fully Compatible with 80286**
- **Object Code Compatible with All 8086 Family Microprocessors**
- **Virtual 8086 Mode Allows Running of 8086 Software in a Protected and Paged System**
- **Hardware Debugging Support**
- **Optimized for System Performance**
  - **— Pipelined Instruction Execution**
  - **— On-Chip Address Translation Caches**
  - **— 16, 20 and 25 MHz Clock**
  - **— 32, 40 and 50 Megabytes/Sec Bus Bandwidth**
- **High Speed Numerics Support via 80387 Coprocessor**
- **Complete System Development Support**
  - **— Software: C, PL/M, Assembler System Generation Tools**
  - **— Debuggers: PSCOPE, ICE™-386**
- **High Speed CHMOS III and CHMOS IV Technology**
- **132 Pin Grid Array Package**

(See Packaging Specification, Order #231369)

The 386™ Microprocessor is an advanced 32-bit microprocessor designed for applications needing very high performance and optimized for multitasking operating systems. The 32-bit registers and data paths support 32-bit addresses and data types. The processor addresses up to four gigabytes of physical memory and 64 terabytes (2**46) of virtual memory. The integrated memory management and protection architecture includes address translation registers, advanced multitasking hardware and a protection mechanism to support operating systems. In addition, the 386 Microprocessor allows the simultaneous running of multiple operating systems. Instruction pipelining, on-chip address translation, and high bus bandwidth ensure short average instruction execution times and high system throughput.

The 386 Microprocessor offers new testability and debugging features. Testability features include a self-test and direct access to the page translation cache. Four new breakpoint registers provide breakpoint traps on code execution or data accesses, for powerful debugging of even ROM-based systems.

Object-code compatibility with all 8086 family members (8086, 8088, 80186, 80188, 80286) means the 386 Microprocessor offers immediate access to the world's largest microprocessor software base.



231630-49

Figure 1-1. 386™ Microprocessor Pipelined 32-Bit Microarchitecture

386™ is Trademark of Intel Corporation.
UNIX™ is a Trademark of AT&T Bell Labs.
MS-DOS is a Trademark of MICROSOFT Corporation.

November 1988
Order Number: 231630-005

2. BASE ARCHITECTURE

2.1 INTRODUCTION

The 386™ Microprocessor consists of a central processing unit, a memory management unit and a bus interface.

The central processing unit consists of the execution unit and instruction unit. The execution unit contains the eight 32-bit general purpose registers which are used for both address calculation, data operations and a 64-bit barrel shifter used to speed shift, rotate, multiply, and divide operations. The multiply and divide logic uses a 1-bit per cycle algorithm. The multiply algorithm stops the iteration when the most significant bits of the multiplier are all zero. This allows typical 32-bit multiplies to be executed in under one microsecond. The instruction unit decodes the instruction opcodes and stores them in the decoded instruction queue for immediate use by the execution unit.

The memory management unit (MMU) consists of a segmentation unit and a paging unit. Segmentation allows the managing of the logical address space by providing an extra addressing component, one that allows easy code and data relocatability, and efficient sharing. The paging mechanism operates beneath and is transparent to the segmentation process, to allow management of the physical address space. Each segment is divided into one or more 4K byte pages. To implement a virtual memory system, the 386 Microprocessor supports full restartability for all page and segment faults.

Memory is organized into one or more variable length segments, each up to four gigabytes in size. A given region of the linear address space, a segment, can have attributes associated with it. These attributes include its location, size, type (i.e. stack, code or data), and protection characteristics. Each task on an 386 Microprocessor can have a maximum of 16,381 segments of up to four gigabytes each, thus providing 64 terabytes (trillion bytes) of virtual memory to each task.

The segmentation unit provides four-levels of protection for isolating and protecting applications and the operating system from each other. The hardware enforced protection allows the design of systems with a high degree of integrity.

The 386 Microprocessor has two modes of operation: Real Address Mode (Real Mode), and Protected Virtual Address Mode (Protected Mode). In Real Mode the 386 Microprocessor operates as a very fast 8086, but with 32-bit extensions if desired. Real Mode is required primarily to setup the processor for Protected Mode operation. Protected Mode provides access to the sophisticated memory management, paging and privilege capabilities of the processor.

Within Protected Mode, software can perform a task switch to enter into tasks designated as Virtual 8086 Mode tasks. Each such task behaves with 8086 semantics, thus allowing 8086 software (an application program, or an entire operating system) to execute. The Virtual 8086 tasks can be isolated and protected from one another and the host 386 Microprocessor operating system, by the use of paging, and the I/O Permission Bitmap.

Finally, to facilitate high performance system hardware designs, the 386 Microprocessor bus interface offers address pipelining, dynamic data bus sizing, and direct Byte Enable signals for each byte of the data bus. These hardware features are described fully beginning in Section 5.

2.2 REGISTER OVERVIEW

The 386 Microprocessor has 32 register resources in the following categories:

- General Purpose Registers
- Segment Registers
- Instruction Pointer and Flags
- Control Registers
- System Address Registers
- Debug Registers
- Test Registers.

The registers are a superset of the 8086, 80186 and 80286 registers, so all 16-bit 8086, 80186 and 80286 registers are contained within the 32-bit 386 Microprocessor.

Figure 2-1 shows all of 386 Microprocessor base architecture registers, which include the general address and data registers, the instruction pointer, and the flags register. The contents of these registers are task-specific, so these registers are automatically loaded with a new context upon a task switch operation.

The base architecture also includes six directly accessible segments, each up to 4 Gbytes in size. The segments are indicated by the selector values placed in 386 Microprocessor segment registers of Figure 2-1. Various selector values can be loaded as a program executes, if desired.

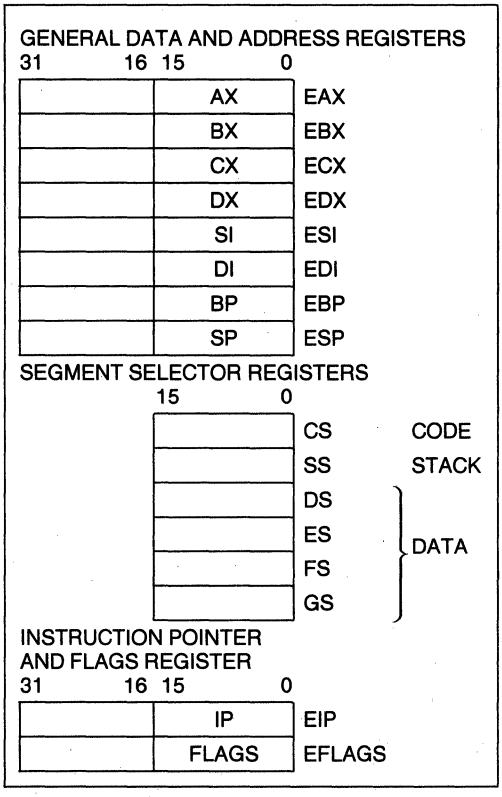


Figure 2-1. 386™ Microprocessor Base Architecture Registers

The selectors are also task-specific, so the segment registers are automatically loaded with new context upon a task switch operation.

The other types of registers, Control, System Address, Debug, and Test, are primarily used by system software.

2.3 REGISTER DESCRIPTIONS

2.3.1 General Purpose Registers

General Purpose Registers: The eight general purpose registers of 32 bits hold data or address quantities. The general registers, Figure 2-2, support data operands of 1, 8, 16, 32 and 64 bits, and bit fields of 1 to 32 bits. They support address operands of 16 and 32 bits. The 32-bit registers are named EAX, EBX, ECX, EDX, ESI, EDI, EBP, and ESP.

The least significant 16 bits of the registers can be accessed separately. This is done by using the 16-bit names of the registers AX, BX, CX, DX, SI, DI, BP, and SP. When accessed as a 16-bit operand, the upper 16 bits of the register are neither used nor changed.

Finally 8-bit operations can individually access the lowest byte (bits 0–7) and the higher byte (bits 8–15) of general purpose registers AX, BX, CX and DX. The lowest bytes are named AL, BL, CL and DL, respectively. The higher bytes are named AH, BH, CH and DH, respectively. The individual byte accessibility offers additional flexibility for data operations, but is not used for effective address calculation.

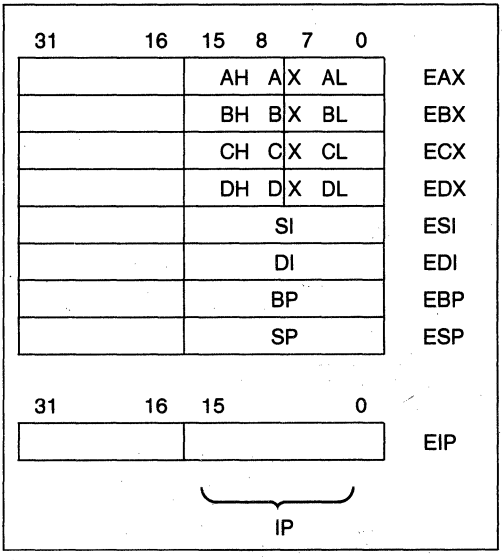


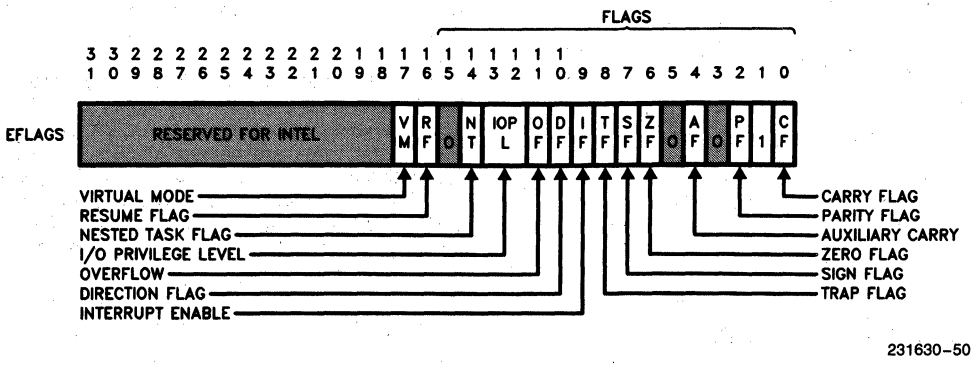
Figure 2-2. General Registers and Instruction Pointer

2.3.2 Instruction Pointer

The instruction pointer, Figure 2-2, is a 32-bit register named EIP. EIP holds the offset of the next instruction to be executed. The offset is always relative to the base of the code segment (CS). The lower 16 bits (bits 0–15) of EIP contain the 16-bit instruction pointer named IP, which is used by 16-bit addressing.

2.3.3 Flags Register

The Flags Register is a 32-bit register named EFLAGS. The defined bits and bit fields within EFLAGS, shown in Figure 2-3, control certain operations and indicate status of the 386 Microprocessor. The lower 16 bits (bit 0–15) of EFLAGS contain the 16-bit flag register named FLAGS, which is most useful when executing 8086 and 80286 code.



NOTE:
0 indicates Intel reserved: do not define; see section 2.3.10.

Figure 2-3. Flags Register

VM (Virtual 8086 Mode, bit 17)

The VM bit provides Virtual 8086 Mode within Protected Mode. If set while the 386 Microprocessor is in Protected Mode, the 386 Microprocessor will switch to Virtual 8086 operation, handling segment loads as the 8086 does, but generating exception 13 faults on privileged opcodes. The VM bit can be set only in Protected Mode, by the IRET instruction (if current privilege level = 0) and by task switches at any privilege level. The VM bit is unaffected by POPF. PUSHF always pushes a 0 in this bit, even if executing in virtual 8086 Mode. The EFLAGS image pushed during interrupt processing or saved during task switches will contain a 1 in this bit if the interrupted code was executing as a Virtual 8086 Task.

RF (Resume Flag, bit 16)

The RF flag is used in conjunction with the debug register breakpoints. It is checked at instruction boundaries before breakpoint processing. When RF is set, it causes any debug fault to be ignored on the next instruction. RF is then automatically reset at the successful completion of every instruction (no faults are signalled) except the IRET instruction, the POPF instruction, (and JMP, CALL, and INT instructions causing a task switch). These instructions set RF to the value specified by the memory image. For example, at the end of the breakpoint service routine, the IRET instruction can pop an EFLAG image having the RF bit set and resume the program's execution at the breakpoint address without generating another breakpoint fault on the same location.

NT (Nested Task, bit 14)

This flag applies to Protected Mode. NT is set to indicate that the execution of this task is nested within another task. If set, it indicates that the current nested task's Task State Segment (TSS) has a valid back link to the previous task's TSS. This bit is set or reset by control transfers to other tasks. The value of NT in EFLAGS is tested by the IRET instruction to determine whether to do an inter-task return or an intra-task return. A POPF or an IRET instruction **will** affect the setting of this bit according to the image popped, at any privilege level.

IOPL (Input/Output Privilege Level, bits 12-13)

This two-bit field applies to Protected Mode. IOPL indicates the numerically maximum CPL (current privilege level) value permitted to execute I/O instructions without generating an exception 13 fault or consulting the I/O Permission Bitmap. It also indicates the maximum CPL value allowing alteration of the IF (INTR Enable Flag) bit when new values are popped into the EFLAG register. POPF and IRET instruction can alter the IOPL field when executed at CPL = 0. Task switches can always alter the IOPL field, when the new flag image is loaded from the incoming task's TSS.

OF (Overflow Flag, bit 11)

OF is set if the operation resulted in a signed overflow. Signed overflow occurs when the operation resulted in carry/borrow **into** the sign bit (high-order bit) of the result but did not result in a carry/borrow **out of** the high-order bit, or vice-versa. For 8/16/32 bit operations, OF is set according to overflow at bit 7/15/31, respectively.

DF (Direction Flag, bit 10)

DF defines whether ESI and/or EDI registers postdecrement or postincrement during the string instructions. Postincrement occurs if DF is reset. Postdecrement occurs if DF is set.

IF (INTR Enable Flag, bit 9)

The IF flag, when set, allows recognition of external interrupts signalled on the INTR pin. When IF is reset, external interrupts signalled on the INTR are not recognized. IOPL indicates the maximum CPL value allowing alteration of the IF bit when new values are popped into EFLAGS or FLAGS.

TF (Trap Enable Flag, bit 8)

TF controls the generation of exception 1 trap when single-stepping through code. When TF is set, the 386 Microprocessor generates an exception 1 trap after the next instruction is executed. When TF is reset, exception 1 traps occur only as a function of the breakpoint addresses loaded into debug registers DR0–DR3.

SF (Sign Flag, bit 7)

SF is set if the high-order bit of the result is set, it is reset otherwise. For 8-, 16-, 32-bit operations, SF reflects the state of bit 7, 15, 31 respectively.

ZF (Zero Flag, bit 6)

ZF is set if all bits of the result are 0. Otherwise it is reset.

AF (Auxiliary Carry Flag, bit 4)

The Auxiliary Flag is used to simplify the addition and subtraction of packed BCD quantities. AF is set if the operation resulted in a carry out of bit 3 (addition) or a borrow into bit 3 (subtraction). Otherwise AF is reset. AF is affected by carry out of, or borrow into bit 3 only, regardless of overall operand length: 8, 16 or 32 bits.

PF (Parity Flags, bit 2)

PF is set if the low-order eight bits of the operation contains an even number of "1's" (even parity). PF is reset if the low-order eight bits have odd parity. PF is a function of only the low-order eight bits, regardless of operand size.

CF (Carry Flag, bit 0)

CF is set if the operation resulted in a carry out of (addition), or a borrow into (subtraction) the high-order bit. Otherwise CF is reset. For 8-, 16- or 32-bit operations, CF is set according to carry/borrow at bit 7, 15 or 31, respectively.

Note in these descriptions, "set" means "set to 1," and "reset" means "reset to 0."

2.3.4 Segment Registers

Six 16-bit segment registers hold segment selector values identifying the currently addressable memory segments. Segment registers are shown in Figure 2-4. In Protected Mode, each segment may range in size from one byte up to the entire linear and physi-

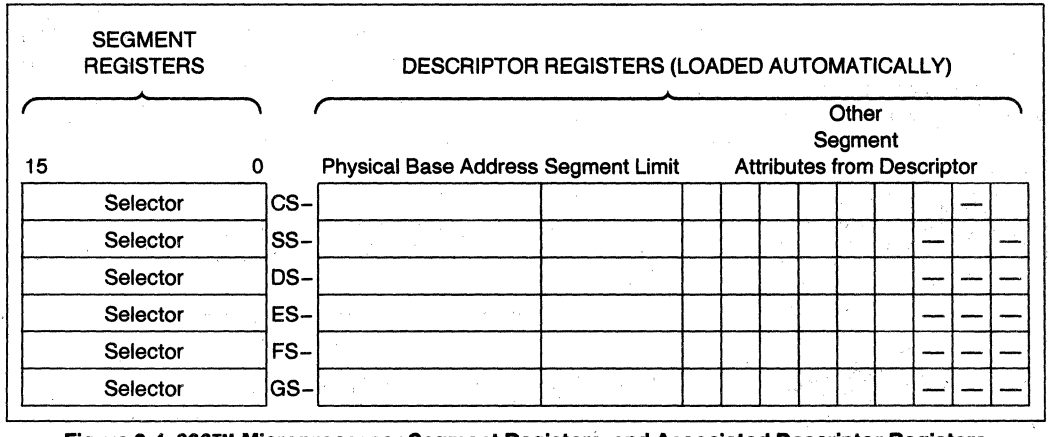


Figure 2-4. 386™ Microprocessor Segment Registers, and Associated Descriptor Registers

cal space of the machine, 4 Gbytes (2^{32} bytes). In Real Address Mode, the maximum segment size is fixed at 64 Kbytes (2^{16} bytes).

The six segments addressable at any given moment are defined by the segment registers CS, SS, DS, ES, FS and GS. The selector in CS indicates the current code segment; the selector in SS indicates the current stack segment; the selectors in DS, ES, FS and GS indicate the current data segments.

2.3.5 Segment Descriptor Registers

The segment descriptor registers are not programmer visible, yet it is very useful to understand their content. Inside the 386 Microprocessor, a descriptor register (programmer invisible) is associated with each programmer-visible segment register, as shown by Figure 2-4. Each descriptor register holds a 32-bit segment base address, a 32-bit segment limit, and the other necessary segment attributes.

When a selector value is loaded into a segment register, the associated descriptor register is automatically updated with the correct information. In Real Address Mode, only the base address is updated directly (by shifting the selector value four bits to the left), since the segment maximum limit and attributes are fixed in Real Mode. In Protected Mode, the base address, the limit, and the attributes are all updated per the contents of the segment descriptor indexed by the selector.

Whenever a memory reference occurs, the segment descriptor register associated with the segment being used is automatically involved with the memory reference. The 32-bit segment base address becomes a component of the linear address calculation, the 32-bit limit is used for the limit-check operation, and the attributes are checked against the type of memory reference requested.

2.3.6 Control Registers

The 386 Microprocessor has three control registers of 32 bits, CR0, CR2 and CR3, to hold machine state of a global nature (not specific to an individual task). These registers, along with System Address Registers described in the next section, hold machine state that affects all tasks in the system. To access the Control Registers, load and store instructions are defined.

CR0: Machine Control Register (includes 80286 Machine Status Word)

CR0, shown in Figure 2-5, contains 6 defined bits for control and status purposes. The low-order 16 bits of CR0 are also known as the Machine Status Word, MSW, for compatibility with 80286 Protected Mode. LMSW and SMSW instructions are taken as special aliases of the load and store CR0 operations, where only the low-order 16 bits of CR0 are involved. For compatibility with 80286 operating systems the 386 Microprocessor's LMSW instructions work in an identical fashion to the LMSW instruction on the 80286. (i.e. It only operates on the low-order 16-bits of CR0 and it ignores the new bits in CR0.) New 386 Microprocessor operating systems should use the MOV CR0, Reg instruction.

The defined CR0 bits are described below.

PG (Paging Enable, bit 31)

the PG bit is set to enable the on-chip paging unit. It is reset to disable the on-chip paging unit.

ET (Processor Extension Type, bit 4)

ET indicates the processor extension type (either 80287 or 80387) as detected by the level of the ERROR# input following 386 Microprocessor reset. The ET bit may also be set or reset by loading CR0 under program control if desired. If ET is set, the 80387-compatible 32-bit protocol is used. If ET is reset, 80287-compatible 16-bit protocol is used.

Note that for strict 80286 compatibility, ET is not affected by the LMSW instruction. When the MSW or CR0 is stored, bit 4 accurately reflects the current state of the ET bit.

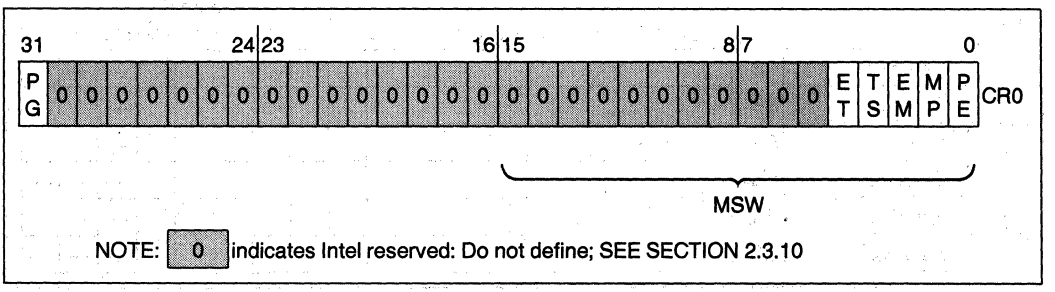


Figure 2-5. Control Register 0

TS (Task Switched, bit 3)

TS is automatically set whenever a task switch operation is performed. If TS is set, a coprocessor ESCape opcode will cause a Coprocessor Not Available trap (exception 7). The trap handler typically saves the 80287/80387 context belonging to a previous task, loads the 80287/80387 state belonging to the current task, and clears the TS bit before returning to the faulting coprocessor opcode.

EM (Emulate Coprocessor, bit 2)

The EMulate coprocessor bit is set to cause all coprocessor opcodes to generate a Coprocessor Not Available fault (exception 7). It is reset to allow coprocessor opcodes to be executed on an actual 80287 or 80387 coprocessor (this the default case after reset). Note that the WAIT opcode is not affected by the EM bit setting.

MP (Monitor Coprocessor, bit 1)

The MP bit is used in conjunction with the TS bit to determine if the WAIT opcode will generate a Coprocessor Not Available fault (exception 7) when TS = 1. When both MP = 1 and TS = 1, the WAIT opcode generates a trap. Otherwise, the WAIT opcode does not generate a trap. Note that TS is automatically set whenever a task switch operation is performed.

PE (Protection Enable, bit 0)

The PE bit is set to enable the Protected Mode. If PE is reset, the processor operates again in Real Mode. PE may be set by loading MSW or CR0. PE can be reset only by a load into CR0. Resetting the PE bit is typically part of a longer instruction sequence needed for proper transition from Protected Mode to Real Mode. Note that for strict 80286 compatibility, PE cannot be reset by the LMSW instruction.

CR1: reserved

CR1 is reserved for use in future Intel processors.

CR2: Page Fault Linear Address

CR2, shown in Figure 2-6, holds the 32-bit linear address that caused the last page fault detected. The error code pushed onto the page fault handler's stack when it is invoked provides additional status information on this page fault.

CR3: Page Directory Base Address

CR3, shown in Figure 2-6, contains the physical base address of the page directory table. The 386 Microprocessor page directory table is always page-aligned (4 Kbyte-aligned). Therefore the lowest twelve bits of CR3 are ignored when written and they store as undefined.

A task switch through a TSS which **changes** the value in CR3, or an explicit load into CR3 with any value, will invalidate all cached page table entries in the paging unit cache. Note that if the value in CR3 does not change during the task switch, the cached page table entries are not flushed.

2.3.7 System Address Registers

Four special registers are defined to reference the tables or segments supported by the 80286 CPU and 386 Microprocessor protection model. These tables or segments are:

GDT (Global Descriptor Table),

IDT (Interrupt Descriptor Table),

LDT (Local Descriptor Table),

TSS (Task State Segment).

The addresses of these tables and segments are stored in special registers, the System Address and System Segment Registers illustrated in Figure 2-7. These registers are named GDTR, IDTR, LDTR and TR, respectively. Section 4 **Protected Mode Architecture** describes the use of these registers.

GDTR and IDTR

These registers hold the 32-bit linear base address and 16-bit limit of the GDT and IDT, respectively.

The GDT and IDT segments, since they are global to all tasks in the system, are defined by 32-bit linear addresses (subject to page translation if paging is enabled) and 16-bit limit values.

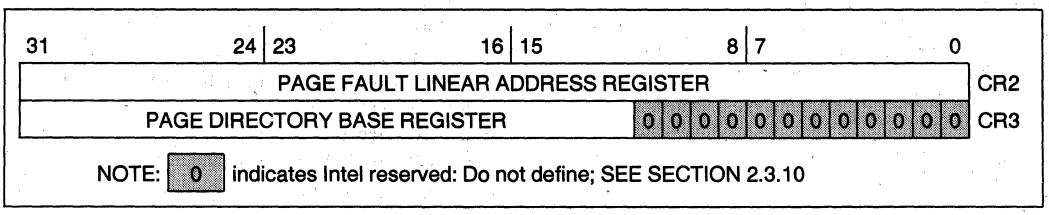


Figure 2-6. Control Registers 2 and 3

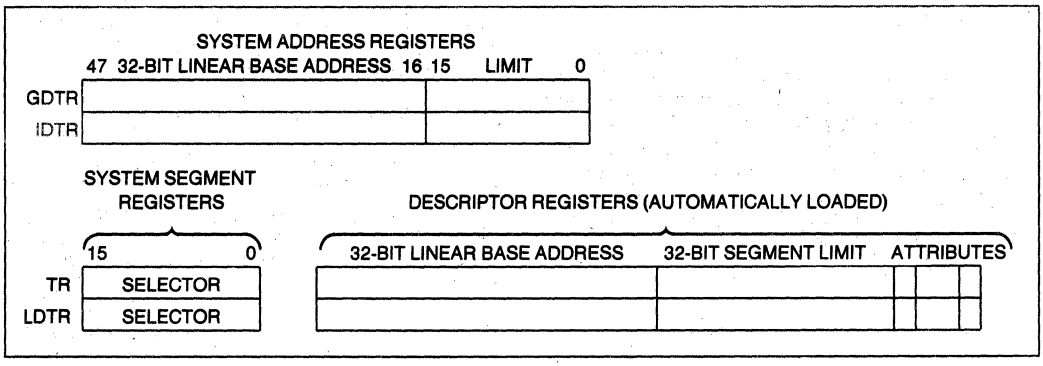


Figure 2-7. System Address and System Segment Registers

LDTR and TR

These registers hold the 16-bit selector for the LDT descriptor and the TSS descriptor, respectively.

The LDT and TSS segments, since they are task-specific segments, are defined by selector values stored in the system segment registers. Note that a segment descriptor register (programmer-invisible) is associated with each system segment register.

2.3.8 Debug and Test Registers

Debug Registers: The six programmer accessible debug registers provide on-chip support for debugging. Debug Registers DR0–3 specify the four linear breakpoints. The Debug Control Register DR7 is used to set the breakpoints and the Debug Status Register DR6, displays the current state of the breakpoints. The use of the debug registers is described in section 2.12 **Debugging support.**

DEBUG REGISTERS

31 0	
LINEAR BREAKPOINT ADDRESS 0	DR0
LINEAR BREAKPOINT ADDRESS 1	DR1
LINEAR BREAKPOINT ADDRESS 2	DR2
LINEAR BREAKPOINT ADDRESS 3	DR3
Intel reserved. Do not define.	DR4
Intel reserved. Do not define.	DR5
BREAKPOINT STATUS	DR6
BREAKPOINT CONTROL	DR7

TEST REGISTERS (FOR PAGE CACHE)

31 0	
TEST CONTROL	TR6
TEST STATUS	TR7

Figure 2-8. Debug and Test Registers

Test Registers: Two registers are used to control the testing of the RAM/CAM (Content Addressable Memories) in the Translation Lookaside Buffer portion of the 386 Microprocessor. TR6 is the command test register, and TR7 is the data register which contains the data of the Translation Lookaside buffer test. Their use is discussed in section 2.11 **Testability.**

Figure 2-8 shows the Debug and Test registers.

2.3.9 Register Accessibility

There are a few differences regarding the accessibility of the registers in Real and Protected Mode. Table 2-1 summarizes these differences. See Section 4 **Protected Mode Architecture** for further details.

2.3.10 Compatibility

VERY IMPORTANT NOTE:
COMPATIBILITY WITH FUTURE PROCESSORS

In the preceding register descriptions, note certain 386 Microprocessor register bits are Intel reserved. When reserved bits are called out, treat them as fully undefined. This is essential for your software compatibility with future processors! Follow the guidelines below:

1) **Do not depend on the states of any undefined bits when testing the values of defined register bits. Mask them out when testing.**
2) **Do not depend on the states of any undefined bits when storing them to memory or another register.**
3) **Do not depend on the ability to retain information written into any undefined bits.**
4) **When loading registers always load the undefined bits as zeros.**

Table 2-1. Register Usage

Register	Use in Real Mode		Use in Protected Mode		Use in Virtual 8086 Mode	
	Load	Store	Load	Store	Load	Store
General Registers	Yes	Yes	Yes	Yes	Yes	Yes
Segment Registers	Yes	Yes	Yes	Yes	Yes	Yes
Flag Register	Yes	Yes	Yes	Yes	IOPL	IOPL*
Control Registers	Yes	Yes	PL = 0	PL = 0	No	Yes
GDTR	Yes	Yes	PL = 0	Yes	No	Yes
IDTR	Yes	Yes	PL = 0	Yes	No	Yes
LDTR	No	No	PL = 0	Yes	No	No
TR	No	No	PL = 0	Yes	No	No
Debug Control	Yes	Yes	PL = 0	PL = 0	No	No
Test Registers	Yes	Yes	PL = 0	PL = 0	No	No

NOTES:
PL = 0: The registers can be accessed only when the current privilege level is zero.
*IOPL: The PUSHF and POPF instructions are made I/O Privilege Level sensitive in Virtual 8086 Mode.

5) However, registers which have been previously stored may be reloaded without masking.

Depending upon the values of undefined register bits will make your software dependent upon the unspecified 386 Microprocessor handling of these bits. Depending on undefined values risks making your software incompatible with future processors that define usages for the 386 Microprocessor-undefined bits. AVOID ANY SOFTWARE DEPENDENCE UPON THE STATE OF UNDEFINED 386 MICROPROCESSOR REGISTER BITS.

2.4 INSTRUCTION SET

2.4.1 Instruction Set Overview

The instruction set is divided into nine categories of operations:

Data Transfer
Arithmetic
Shift/Rotate
String Manipulation
Bit Manipulation
Control Transfer
High Level Language Support
Operating System Support
Processor Control

These 386 Microprocessor instructions are listed in Table 2-2.

All 386 Microprocessor instructions operate on either 0, 1, 2, or 3 operands; where an operand resides in a register, in the instruction itself, or in memory. Most zero operand instructions (e.g. CLI, STI) take only one byte. One operand instructions generally are two bytes long. The average instruction is 3.2 bytes long. Since the 386 Microprocessor has a 16-byte instruction queue, an average of 5 instructions will be prefetched. The use of two operands permits the following types of common instructions:

Register to Register
Memory to Register
Immediate to Register
Register to Memory
Immediate to Memory.

The operands can be either 8, 16, or 32 bits long. As a general rule, when executing code written for the 386 Microprocessor (32-bit code), operands are 8 or 32 bits; when executing existing 80286 or 8086 code (16-bit code), operands are 8 or 16 bits. Prefixes can be added to all instructions which override the default length of the operands, (i.e. use 32-bit operands for 16-bit code, or 16-bit operands for 32-bit code).

2.4.2 386™ Microprocessor Instructions

Table 2-2a. Data Transfer

GENERAL PURPOSE	
MOV	Move operand
PUSH	Push operand onto stack
POP	Pop operand off stack
PUSHA	Push all registers on stack
POPA	Pop all registers off stack
XCHG	Exchange Operand, Register
XLAT	Translate
CONVERSION	
MOVZX	Move byte or Word, Dword, with zero extension
MOVSX	Move byte or Word, Dword, sign extended
CBW	Convert byte to Word, or Word to Dword
CWD	Convert Word to DWORD
CWDE	Convert Word to DWORD extended
CDQ	Convert DWORD to QWORD
INPUT/OUTPUT	
IN	Input operand from I/O space
OUT	Output operand to I/O space
ADDRESS OBJECT	
LEA	Load effective address
LDS	Load pointer into D segment register
LES	Load pointer into E segment register
LFS	Load pointer into F segment register
LGS	Load pointer into G segment register
LSS	Load pointer into S (Stack) segment register
FLAG MANIPULATION	
LAHF	Load A register from Flags
SAHF	Store A register in Flags
PUSHF	Push flags onto stack
POPF	Pop flags off stack
PUSHFD	Push EFlags onto stack
POPFD	Pop EFlags off stack
CLC	Clear Carry Flag
CLD	Clear Direction Flag
CMC	Complement Carry Flag
STC	Set Carry Flag
STD	Set Direction Flag

Table 2-2b. Arithmetic Instructions

ADDITION	
ADD	Add operands
ADC	Add with carry
INC	Increment operand by 1
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition
SUBTRACTION	
SUB	Subtract operands
SBB	Subtract with borrow
DEC	Decrement operand by 1
NEG	Negate operand
CMP	Compare operands
DAS	Decimal adjust for subtraction
AAS	ASCII Adjust for subtraction
MULTIPLICATION	
MUL	Multiply Double/Single Precision
IMUL	Integer multiply
AAM	ASCII adjust after multiply
DIVISION	
DIV	Divide unsigned
IDIV	Integer Divide
AAD	ASCII adjust before division

Table 2-2c. String Instructions

MOVS	Move byte or Word, Dword string
INS	Input string from I/O space
OUTS	Output string to I/O space
CMPS	Compare byte or Word, Dword string
SCAS	Scan Byte or Word, Dword string
LODS	Load byte or Word, Dword string
STOS	Store byte or Word, Dword string
REP	Repeat
REPE/ REPZ	Repeat while equal/zero
RENE/ REPNZ	Repeat while not equal/not zero

Table 2-2d. Logical Instructions

LOGICALS	
NOT	"NOT" operands
AND	"AND" operands
OR	"Inclusive OR" operands
XOR	"Exclusive OR" operands
TEST	"Test" operands

Table 2-2d. Logical Instructions (Continued)

SHIFTS	
SHL/SHR	Shift logical left or right
SAL/SAR	Shift arithmetic left or right
SHLD/ SHRD	Double shift left or right
ROTATES	
ROL/ROR	Rotate left/right
RCL/RCR	Rotate through carry left/right

Table 2-2e. Bit Manipulation Instructions

SINGLE BIT INSTRUCTIONS	
BT	Bit Test
BTS	Bit Test and Set
BTR	Bit Test and Reset
BTC	Bit Test and Complement
BSF	Bit Scan Forward
BSR	Bit Scan Reverse

Table 2-2f. Program Control Instructions

CONDITIONAL TRANSFERS	
SETCC	Set byte equal to condition code
JA/JNBE	Jump if above/not below nor equal
JAE/JNB	Jump if above or equal/not below
JB/JNAE	Jump if below/not above nor equal
JBE/JNA	Jump if below or equal/not above
JC	Jump if carry
JE/JZ	Jump if equal/zero
JG/JNLE	Jump if greater/not less nor equal
JGE/JNL	Jump if greater or equal/not less
JL/JNGE	Jump if less/not greater nor equal
JLE/JNG	Jump if less or equal/not greater
JNC	Jump if not carry
JNE/JNZ	Jump if not equal/not zero
JNO	Jump if not overflow
JNP/JPO	Jump if not parity/parity odd
JNS	Jump if not sign
JO	Jump if overflow
JP/JPE	Jump if parity/parity even
JS	Jump if Sign

Table 2-2f. Program Control Instructions (Continued)

UNCONDITIONAL TRANSFERS	
CALL	Call procedure/task
RET	Return from procedure
JMP	Jump
ITERATION CONTROLS	
LOOP	Loop
LOOPE/ LOOPZ	Loop if equal/zero
LOOPNE/ LOOPNZ	Loop if not equal/not zero
JCXZ	JUMP if register CX = 0
INTERRUPTS	
INT	Interrupt
INTO	Interrupt if overflow
IRET	Return from Interrupt/Task
CLI	Clear interrupt Enable
STI	Set Interrupt Enable

Table 2-2g. High Level Language Instructions

BOUND	Check Array Bounds
ENTER	Setup Parameter Block for Entering Procedure
LEAVE	Leave Procedure

Table 2-2h. Protection Model

SGDT	Store Global Descriptor Table
SIDT	Store Interrupt Descriptor Table
STR	Store Task Register
SLDT	Store Local Descriptor Table
LGDT	Load Global Descriptor Table
LIDT	Load Interrupt Descriptor Table
LTR	Load Task Register
LLDT	Load Local Descriptor Table
ARPL	Adjust Requested Privilege Level
LAR	Load Access Rights
LSL	Load Segment Limit
VERR/ VERW	Verify Segment for Reading or Writing
LMSW	Load Machine Status Word (lower 16 bits of CR0)
SMSW	Store Machine Status Word

Table 2-2i. Processor Control Instructions

HLT	Halt
WAIT	Wait until BUSY# negated
ESC	Escape
LOCK	Lock Bus

2.5 ADDRESSING MODES

2.5.1 Addressing Modes Overview

The 386 Microprocessor provides a total of 11 addressing modes for instructions to specify operands. The addressing modes are optimized to allow the efficient execution of high level languages such as C and FORTRAN, and they cover the vast majority of data references needed by high-level languages.

2.5.2 Register and Immediate Modes

Two of the addressing modes provide for instructions that operate on register or immediate operands:

Register Operand Mode: The operand is located in one of the 8-, 16- or 32-bit general registers.

Immediate Operand Mode: The operand is included in the instruction as part of the opcode.

2.5.3 32-Bit Memory Addressing Modes

The remaining 9 modes provide a mechanism for specifying the effective address of an operand. The linear address consists of two components: the segment base address and an effective address. The effective address is calculated by using combinations of the following four address elements:

DISPLACEMENT: An 8-, or 32-bit immediate value, following the instruction.

BASE: The contents of any general purpose register. The base registers are generally used by compilers to point to the start of the local variable area.

INDEX: The contents of any general purpose register except for ESP. The index registers are used to access the elements of an array, or a string of characters.

SCALE: The index register's value can be multiplied by a scale factor, either 1, 2, 4 or 8. Scaled index mode is especially useful for accessing arrays or structures.

Combinations of these 4 components make up the 9 additional addressing modes. There is no performance penalty for using any of these addressing combinations, since the effective address calculation is pipelined with the execution of other instructions. The one exception is the simultaneous use of Base and Index components which requires one additional clock.

As shown in Figure 2-9, the effective address (EA) of an operand is calculated according to the following formula.

EA = Base Reg + (Index Reg * Scaling) + Displacement

Direct Mode: The operand's offset is contained as part of the instruction as an 8-, 16- or 32-bit displacement.
EXAMPLE: INC Word PTR [500]

Register Indirect Mode: A BASE register contains the address of the operand.
EXAMPLE: MOV [ECX], EDX

Based Mode: A BASE register's contents is added to a DISPLACEMENT to form the operands offset.
EXAMPLE: MOV ECX, [EAX + 24]

Index Mode: An INDEX register's contents is added to a DISPLACEMENT to form the operands offset.
EXAMPLE: ADD EAX, TABLE[ESI]

Scaled Index Mode: An INDEX register's contents is multiplied by a scaling factor which is added to a DISPLACEMENT to form the operands offset.
EXAMPLE: IMUL EBX, TABLE[ESI*4],7

Based Index Mode: The contents of a BASE register is added to the contents of an INDEX register to form the effective address of an operand.
EXAMPLE: MOV EAX, [ESI] [EBX]

Based Scaled Index Mode: The contents of an INDEX register is multiplied by a SCALING factor and the result is added to the contents of a BASE register to obtain the operands offset.
EXAMPLE: MOV ECX, [EDX*8] [EAX]

Based Index Mode with Displacement: The contents of an INDEX Register and a BASE register's contents and a DISPLACEMENT are all summed together to form the operand offset.
EXAMPLE: ADD EDX, [ESI] [EBP + 00FFFFF0H]

Based Scaled Index Mode with Displacement: The contents of an INDEX register are multiplied by a SCALING factor, the result is added to the contents of a BASE register and a DISPLACEMENT to form the operand's offset.
EXAMPLE: MOV EAX, LOCALTABLE[EDI*4] [EBP + 80]

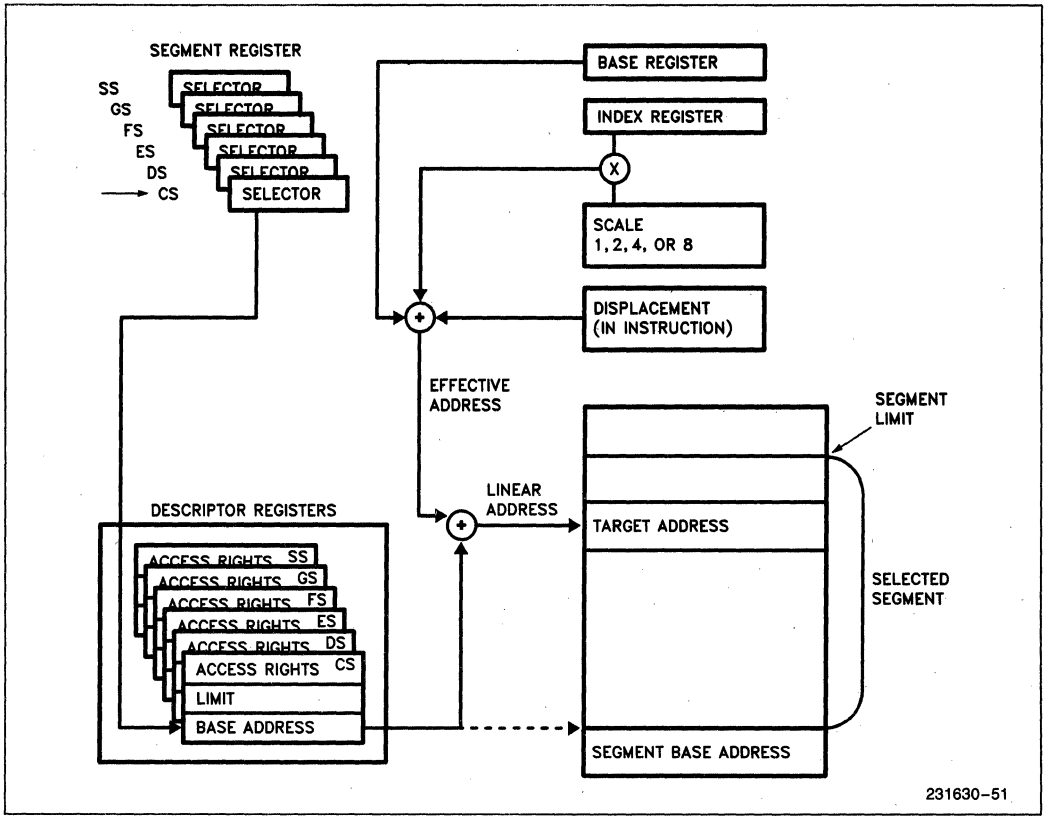


Figure 2-9. Addressing Mode Calculations

2.5.4 Differences Between 16 and 32 Bit Addresses

In order to provide software compatibility with the 80286 and the 8086, the 386 Microprocessor can execute 16-bit instructions in Real and Protected Modes. The processor determines the size of the instructions it is executing by examining the D bit in the CS segment Descriptor. If the D bit is 0 then all operand lengths and effective addresses are assumed to be 16 bits long. If the D bit is 1 then the default length for operands and addresses is 32 bits. In Real Mode the default size for operands and addresses is 16-bits.

Regardless of the default precision of the operands or addresses, the 386 Microprocessor is able to execute either 16 or 32-bit instructions. This is specified via the use of override prefixes. Two prefixes, the **Operand Size Prefix** and the **Address Length Prefix**, override the value of the D bit on an individual instruction basis. These prefixes are automatically added by Intel assemblers.

Example: The processor is executing in Real Mode and the programmer needs to access the EAX registers. The assembler code for this might be MOV EAX, 32-bit MEMORYOP, ASM386 Macro Assembler automatically determines that an Operand Size Prefix is needed and generates it.

Example: The D bit is 0, and the programmer wishes to use Scaled Index addressing mode to access an array. The Address Length Prefix allows the use of MOV DX, TABLE[ESI*2]. The assembler uses an Address Length Prefix since, with D=0, the default addressing mode is 16-bits.

Example: The D bit is 1, and the program wants to store a 16-bit quantity. The Operand Length Prefix is used to specify only a 16-bit value; MOV MEM16, DX.

Table 2-3. BASE and INDEX Registers for 16- and 32-Bit Addresses

	16-Bit Addressing	32-Bit Addressing
BASE REGISTER	BX,BP	Any 32-bit GP Register
INDEX REGISTER	SI,DI	Any 32-bit GP Register Except ESP
SCALE FACTOR	none	1, 2, 4, 8
DISPLACEMENT	0, 8, 16 bits	0, 8, 32 bits

The OPERAND LENGTH and Address Length Prefixes can be applied separately or in combination to any instruction. The Address Length Prefix does not allow addresses over 64K bytes to be accessed in Real Mode. A memory address which exceeds FFFFH will result in a General Protection Fault. An Address Length Prefix only allows the use of the additional 386 Microprocessor addressing modes.

When executing 32-bit code, the 386 Microprocessor uses either 8-, or 32-bit displacements, and any register can be used as base or index registers. When executing 16-bit code, the displacements are either 8, or 16 bits, and the base and index register conform to the 80286 model. Table 2-3 illustrates the differences.

2.6 DATA TYPES

The 386 Microprocessor supports all of the data types commonly used in high level languages:

Bit: A single bit quantity.

Bit Field: A group of up to 32 contiguous bits, which spans a maximum of four bytes.

Bit String: A set of contiguous bits, on the 386 Microprocessor bit strings can be up to 4 gigabits long.

Byte: A signed 8-bit quantity.

Unsigned Byte: An unsigned 8-bit quantity.

Integer (Word): A signed 16-bit quantity.

Long Integer (Double Word): A signed 32-bit quantity. All operations assume a 2's complement representation.

Unsigned Integer (Word): An unsigned 16-bit quantity.

Unsigned Long Integer (Double Word): An unsigned 32-bit quantity.

Signed Quad Word: A signed 64-bit quantity.

Unsigned Quad Word: An unsigned 64-bit quantity.

Offset: A 16- or 32-bit offset only quantity which indirectly references another memory location.

Pointer: A full pointer which consists of a 16-bit segment selector and either a 16- or 32-bit offset.

Char: A byte representation of an ASCII Alphanumeric or control character.

String: A contiguous sequence of bytes, words or dwords. A string may contain between 1 byte and 4 Gbytes.

BCD: A byte (unpacked) representation of decimal digits 0–9.

Packed BCD: A byte (packed) representation of two decimal digits 0–9 storing one digit in each nibble.

When the 386 Microprocessor is coupled with a 80387 Numerics Coprocessor then the following common Floating Point types are supported.

Floating Point: A signed 32-, 64-, or 80-bit real number representation. Floating point numbers are supported by the 80387 numerics coprocessor.

Figure 2-10 illustrates the data types supported by the 386 Microprocessor and the 80387.

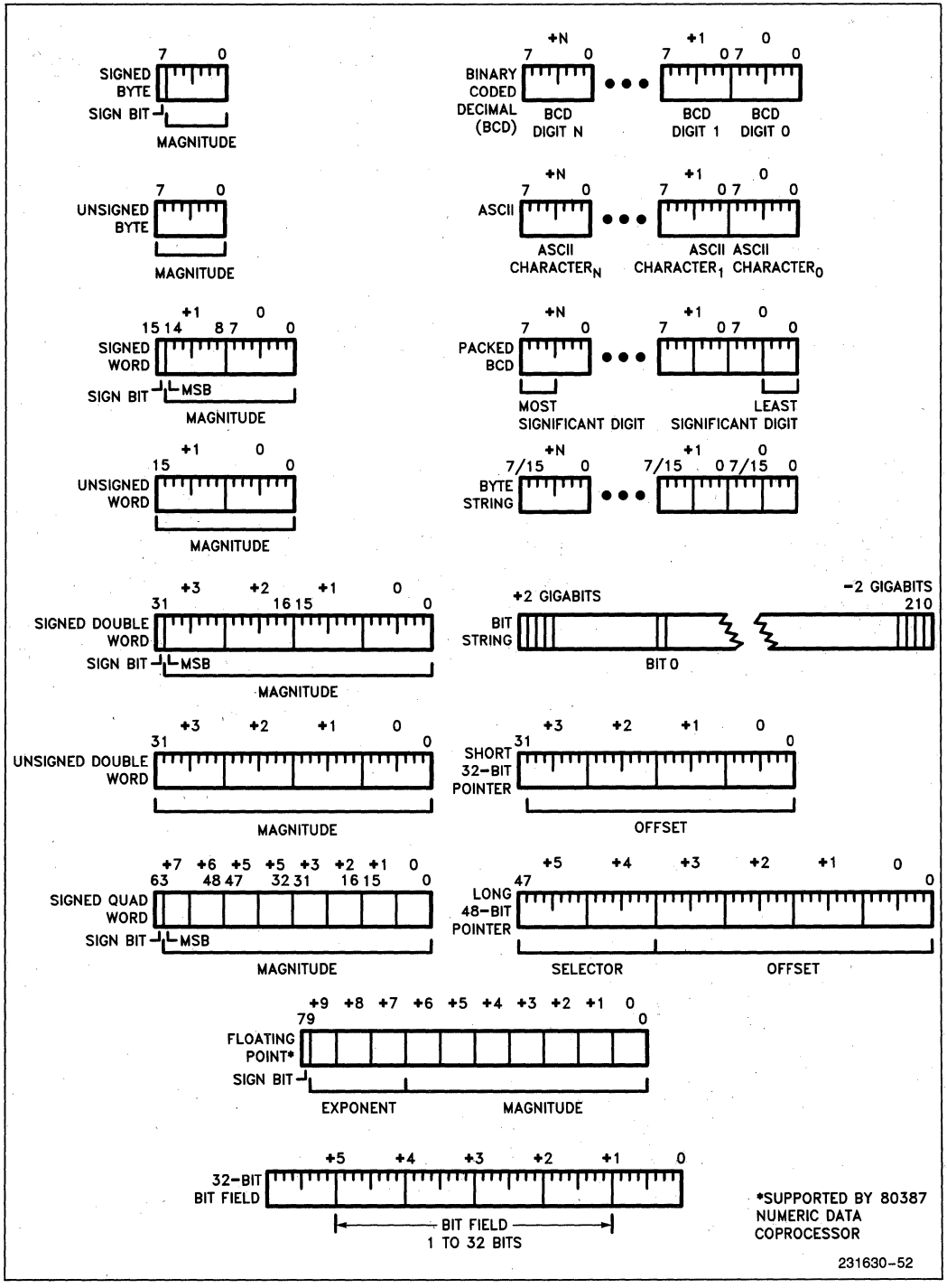


Figure 2-10. 386™ Microprocessor Supported Data Types

2.7 MEMORY ORGANIZATION

2.7.1 Introduction

Memory on the 386 Microprocessor is divided up into 8-bit quantities (bytes), 16-bit quantities (words), and 32-bit quantities (dwords). Words are stored in two consecutive bytes in memory with the low-order byte at the lowest address, the high order byte at the high address. Dwords are stored in four consecutive bytes in memory with the low-order byte at the lowest address, the high-order byte at the highest address. The address of a word or dword is the byte address of the low-order byte.

In addition to these basic data types, the 386 Microprocessor supports two larger units of memory: pages and segments. Memory can be divided up into one or more variable length segments, which can be swapped to disk or shared between programs. Memory can also be organized into one or more 4K byte pages. Finally, both segmentation and paging can be combined, gaining the advantages of both systems. The 386 Microprocessor supports both pages and segments in order to provide maximum flexibility to the system designer. Segmentation and paging are complementary. Segmentation is useful for organizing memory in logical modules, and as such is a tool for the application programmer, while pages are useful for the system programmer for managing the physical memory of a system.

2.7.2 Address Spaces

The 386 Microprocessor has three distinct address spaces: **logical, linear,** and **physical**. A **logical** address (also known as a **virtual** address) consists of a selector and an offset. A selector is the contents of a segment register. An offset is formed by summing all of the addressing components (BASE, INDEX, DISPLACEMENT) discussed in section 2.5.3 **Memory Addressing Modes** into an effective address. Since each task on 386 Microprocessor has a maximum of 16K ($2^{14} - 1$) selectors, and offsets can be 4 gigabytes, (2^{32} bits) this gives a total of 2^{46} bits or 64 terabytes of **logical** address space per task. The programmer sees this virtual address space.

The segmentation unit translates the **logical** address space into a 32-bit **linear** address space. If the paging unit is not enabled then the 32-bit **linear** address corresponds to the **physical** address. The paging unit translates the **linear** address space into the **physical** address space. The **physical address** is what appears on the address pins.

The primary difference between Real Mode and Protected Mode is how the segmentation unit performs the translation of the **logical** address into the **linear** address. In Real Mode, the segmentation unit shifts the selector left four bits and adds the result to the offset to form the **linear** address. While in Protected Mode every selector has a **linear** base address associated with it. The **linear base** address is stored in one of two operating system tables (i.e. the Local Descriptor Table or Global Descriptor Table). The selector's **linear base** address is added to the offset to form the final **linear** address.

Figure 2-11 shows the relationship between the various address spaces.

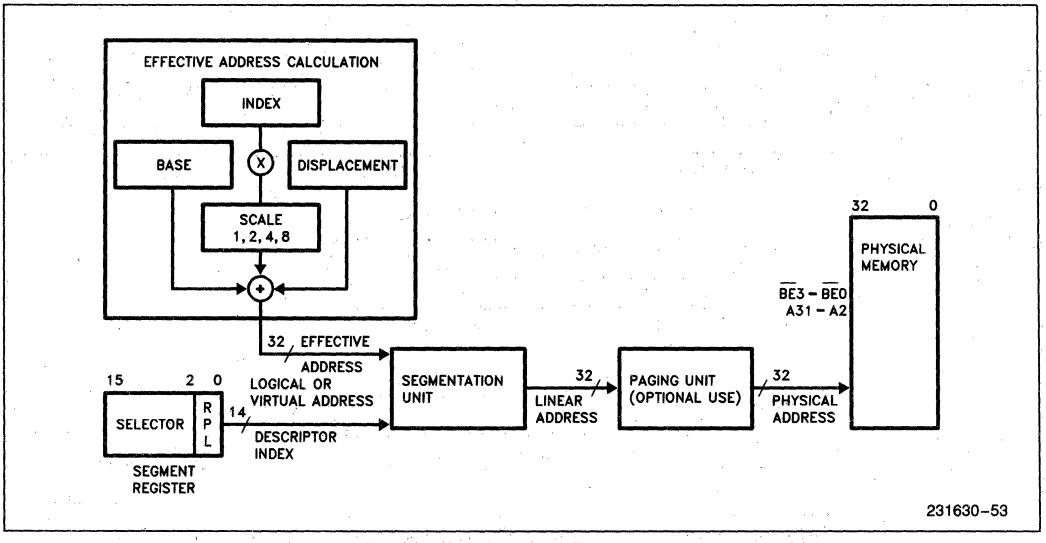


Figure 2-11. Address Translation

2.7.3 Segment Register Usage

The main data structure used to organize memory is the segment. On the 386 Microprocessor, segments are variable sized blocks of linear addresses which have certain attributes associated with them. There are two main types of segments: code and data, the segments are of variable size and can be as small as 1 byte or as large as 4 gigabytes (2^{32} bytes).

In order to provide compact instruction encoding, and increase processor performance, instructions do not need to explicitly specify which segment register is used. A default segment register is automatically chosen according to the rules of Table 2-4 (Segment Register Selection Rules). In general, data references use the selector contained in the DS register; Stack references use the SS register and Instruction fetches use the CS register. The contents of the Instruction Pointer provides the offset. Special segment override prefixes allow the explicit use of a given segment register, and override the implicit rules listed in Table 2-4. The override prefixes also allow the use of the ES, FS and GS segment registers.

There are no restrictions regarding the overlapping of the base addresses of any segments. Thus, all 6 segments could have the base address set to zero and create a system with a four gigabyte linear address space. This creates a system where the virtual address space is the same as the linear address space. Further details of segmentation are discussed in section 4.1.

2.8 I/O SPACE

The 386 Microprocessor has two distinct physical address spaces: Memory and I/O. Generally, peripherals are placed in I/O space although the 386 Microprocessor also supports memory-mapped peripherals. The I/O space consists of 64K bytes, it can be divided into 64K 8-bit ports, 32K 16-bit ports, or 16K 32-bit ports, or any combination of ports which add up to less than 64K bytes. The 64K I/O address space refers to physical memory rather than linear address since I/O instructions do not go through the segmentation or paging hardware. The M/IO# pin acts as an additional address line thus allowing the system designer to easily determine which address space the processor is accessing.

Table 2-4. Segment Register Selection Rules

Type of Memory Reference	Implied (Default) Segment Use	Segment Override Prefixes Possible
Code Fetch	CS	None
Destination of PUSH, PUSHF, INT, CALL, PUSHA Instructions	SS	None
Source of POP, POPA, POPF, IRET, RET instructions	SS	None
Destination of STOS, MOVS, REP STOS, REP MOVS Instructions (DI is Base Register)	ES	None
Other Data References, with Effective Address Using Base Register of:		
[EAX]	DS	CS,SS,ES,FS,GS
[EBX]	DS	CS,SS,ES,FS,GS
[ECX]	DS	CS,SS,ES,FS,GS
[EDX]	DS	CS,SS,ES,FS,GS
[ESI]	DS	CS,SS,ES,FS,GS
[EDI]	DS	CS,SS,ES,FS,GS
[EBP]	SS	CS,SS,ES,FS,GS
[ESP]	SS	CS,SS,ES,FS,GS

The I/O ports are accessed via the IN and OUT I/O instructions, with the port address supplied as an immediate 8-bit constant in the instruction or in the DX register. All 8- and 16-bit port addresses are zero extended on the upper address lines. The I/O instructions cause the M/IO# pin to be driven low.

I/O port addresses 00F8H through 00FFH are reserved for use by Intel.

2.9 INTERRUPTS

2.9.1 Interrupts and Exceptions

Interrupts and exceptions alter the normal program flow, in order to handle external events, to report errors or exceptional conditions. The difference between interrupts and exceptions is that interrupts are used to handle asynchronous external events while exceptions handle instruction faults. Although a program can generate a software interrupt via an INT N instruction, the processor treats software interrupts as exceptions.

Hardware interrupts occur as the result of an external event and are classified into two types: maskable or non-maskable. Interrupts are serviced after the execution of the current instruction. After the interrupt handler is finished servicing the interrupt, execution proceeds with the instruction immediately **after** the interrupted instruction. Sections 2.9.3 and 2.9.4 discuss the differences between Maskable and Non-Maskable interrupts.

Exceptions are classified as faults, traps, or aborts depending on the way they are reported, and whether or not restart of the instruction causing the exception is supported. **Faults** are exceptions that are detected and serviced **before** the execution of the faulting instruction. A fault would occur in a virtual memory system, when the processor referenced a page or a segment which was not present. The operating system would fetch the page or segment from disk, and then the 386 Microprocessor would restart the instruction. **Traps** are exceptions that are reported immediately **after** the execution of the instruction which caused the problem. User defined interrupts are examples of traps. **Aborts** are exceptions which do not permit the precise location of the instruction causing the exception to be determined. Aborts are used to report severe errors, such as a hardware error, or illegal values in system tables.

Thus, when an interrupt service routine has been completed, execution proceeds from the instruction immediately following the interrupted instruction. On the other hand, the return address from an exception fault routine will always point at the instruction causing the exception and include any leading instruction prefixes. Table 2-5 summarizes the possible interrupts for the 386 Microprocessor and shows where the return address points.

The 386 Microprocessor has the ability to handle up to 256 different interrupts/exceptions. In order to service the interrupts, a table with up to 256 interrupt vectors must be defined. The interrupt vectors are simply pointers to the appropriate interrupt service routine. In Real Mode (see section 3.1), the vectors are 4 byte quantities, a Code Segment plus a 16-bit offset; in Protected Mode, the interrupt vectors are 8 byte quantities, which are put in an Interrupt Descriptor Table (see section 4.1). Of the 256 possible interrupts, 32 are reserved for use by Intel, the remaining 224 are free to be used by the system designer.

2.9.2 Interrupt Processing

When an interrupt occurs the following actions happen. First, the current program address and the Flags are saved on the stack to allow resumption of the interrupted program. Next, an 8-bit vector is supplied to the 386 Microprocessor which identifies the appropriate entry in the interrupt table. The table contains the starting address of the interrupt service routine. Then, the user supplied interrupt service routine is executed. Finally, when an IRET instruction is executed the old processor state is restored and program execution resumes at the appropriate instruction.

The 8-bit interrupt vector is supplied to the 386 Microprocessor in several different ways: exceptions supply the interrupt vector internally; software INT instructions contain or imply the vector; maskable hardware interrupts supply the 8-bit vector via the interrupt acknowledge bus sequence. Non-Maskable hardware interrupts are assigned to interrupt vector 2.

2.9.3 Maskable Interrupt

Maskable interrupts are the most common way used by the 386 Microprocessor to respond to asynchronous external hardware events. A hardware interrupt occurs when the INTR is pulled high and the Interrupt Flag bit (IF) is enabled. The processor only responds to interrupts between instructions, (REPeat String instructions, have an "interrupt window", between memory moves, which allows interrupts

Table 2-5. Interrupt Vector Assignments

Function	Interrupt Number	Instruction Which Can Cause Exception	Return Address Points to Faulting Instruction	Type
Divide Error	0	DIV, IDIV	YES	FAULT
Debug Exception	1	any instruction	YES	TRAP*
NMI Interrupt	2	INT 2 or NMI	NO	NMI
One Byte Interrupt	3	INT	NO	TRAP
Interrupt on Overflow	4	INTO	NO	TRAP
Array Bounds Check	5	BOUND	YES	FAULT
Invalid OP-Code	6	Any Illegal Instruction	YES	FAULT
Device Not Available	7	ESC, WAIT	YES	FAULT
Double Fault	8	Any Instruction That Can Generate an Exception		ABORT
Coprocessor Segment Overrun	9	ESC	NO	ABORT
Invalid TSS	10	JMP, CALL, IRET, INT	YES	FAULT
Segment Not Present	11	Segment Register Instructions	YES	FAULT
Stack Fault	12	Stack References	YES	FAULT
General Protection Fault	13	Any Memory Reference	YES	FAULT
Page Fault	14	Any Memory Access or Code Fetch	YES	FAULT
Coprocessor Error	16	ESC, WAIT	YES	FAULT
Intel Reserved	17–32			
Two Byte Interrupt	0–255	INT n	NO	TRAP

* Some debug exceptions may report both traps on the previous instruction, and faults on the next instruction.

during long string moves). When an interrupt occurs the processor reads an 8-bit vector supplied by the hardware which identifies the source of the interrupt, (one of 224 user defined interrupts). The exact nature of the interrupt sequence is discussed in section 5.

The IF bit in the EFLAG registers is reset when an interrupt is being serviced. This effectively disables servicing additional interrupts during an interrupt service routine. However, the IF may be set explicitly by the interrupt handler, to allow the nesting of interrupts. When an IRET instruction is executed the original state of the IF is restored.

2.9.4 Non-Maskable Interrupt

Non-maskable interrupts provide a method of servicing very high priority interrupts. A common example of the use of a non-maskable interrupt (NMI) would be to activate a power failure routine. When the NMI input is pulled high it causes an interrupt with an internally supplied vector value of 2. Unlike a normal hardware interrupt, no interrupt acknowledgment sequence is performed for an NMI.

While executing the NMI servicing procedure, the 386 Microprocessor will not service further NMI requests, until an interrupt return (IRET) instruction is executed or the processor is reset. If NMI occurs while currently servicing an NMI, its presence will be saved for servicing after executing the first IRET instruction. The IF bit is cleared at the beginning of an NMI interrupt to inhibit further INTR interrupts.

2.9.5 Software Interrupts

A third type of interrupt/exception for the 386 Microprocessor is the software interrupt. An INT n instruction causes the processor to execute the inter-

rupt service routine pointed to by the nth vector in the interrupt table.

A special case of the two byte software interrupt INT n is the one byte INT 3, or breakpoint interrupt. By inserting this one byte instruction in a program, the user can set breakpoints in his program as a debugging tool.

A final type of software interrupt, is the single step interrupt. It is discussed in section 2.12.

2.9.6 Interrupt and Exception Priorities

Interrupts are externally-generated events. Maskable Interrupts (on the INTR input) and Non-Maskable Interrupts (on the NMI input) are recognized at instruction boundaries. When NMI and maskable INTR are **both** recognized at the **same** instruction boundary, the 386 Microprocessor invokes the NMI service routine first. If, after the NMI service routine has been invoked, maskable interrupts are still enabled, then the 386 Microprocessor will invoke the appropriate interrupt service routine.

Table 2-6a. 386™ Microprocessor Priority for Invoking Service Routines in Case of Simultaneous External Interrupts

1. NMI
2. INTR

Exceptions are internally-generated events. Exceptions are detected by the 386 Microprocessor if, in the course of executing an instruction, the 386 Microprocessor detects a problematic condition. The 386 Microprocessor then immediately invokes the appropriate exception service routine. The state of the 386 Microprocessor is such that the instruction causing the exception can be restarted. If the exception service routine has taken care of the problematic condition, the instruction will execute without causing the same exception.

It is possible for a single instruction to generate several exceptions (for example, transferring a single operand could generate two page faults if the operand location spans two "not present" pages). However, only one exception is generated upon each attempt to execute the instruction. Each exception service routine should correct its corresponding exception, and restart the instruction. In this manner, exceptions are serviced until the instruction executes successfully.

As the 386 Microprocessor executes instructions, it follows a consistent cycle in checking for exceptions, as shown in Table 2-6b. This cycle is repeated as each instruction is executed, and occurs in parallel with instruction decoding and execution.

Table 2-6b. Sequence of Exception Checking

Consider the case of the 386 Microprocessor having just completed an instruction. It then performs the following checks before reaching the point where the next instruction is completed:

1. Check for Exception 1 Traps from the instruction just completed (single-step via Trap Flag, or Data Breakpoints set in the Debug Registers).
2. Check for Exception 1 Faults in the next instruction (Instruction Execution Breakpoint set in the Debug Registers for the next instruction).
3. Check for external NMI and INTR.
4. Check for Segmentation Faults that prevented fetching the entire next instruction (exceptions 11 or 13).
5. Check for Page Faults that prevented fetching the entire next instruction (exception 14).
6. Check for Faults decoding the next instruction (exception 6 if illegal opcode; exception 6 if in Real Mode or in Virtual 8086 Mode and attempting to execute an instruction for Protected Mode only (see 4.6.4); or exception 13 if instruction is longer than 15 bytes, or privilege violation in Protected Mode (i.e. not at IOPL or at CPL=0).
7. If WAIT opcode, check if TS=1 and MP=1 (exception 7 if both are 1).
8. If ESCAPE opcode for numeric coprocessor, check if EM=1 or TS=1 (exception 7 if either are 1).
9. If WAIT opcode or ESCAPE opcode for numeric coprocessor, check ERROR# input signal (exception 16 if ERROR# input is asserted).
10. Check in the following order for each memory reference required by the instruction:
    a. Check for Segmentation Faults that prevent transferring the entire memory quantity (exceptions 11, 12, 13).
    b. Check for Page Faults that prevent transferring the entire memory quantity (exception 14).

Note that the order stated supports the concept of the paging mechanism being "underneath" the segmentation mechanism. Therefore, for any given code or data reference in memory, segmentation exceptions are generated before paging exceptions are generated.

2.9.7 Instruction Restart

The 386 Microprocessor fully supports restarting all instructions after faults. If an exception is detected in the instruction to be executed (exception categories 4 through 10 in Table 2-6c), the 386 Microprocessor invokes the appropriate exception service routine. The 386 Microprocessor is in a state that permits restart of the instruction, for all cases but those in Table 2-6c. Note that all such cases are easily avoided by proper design of the operating system.

Table 2-6c. Conditions Preventing Instruction Restart

A. An instruction causes a task switch to a task whose Task State Segment is **partially** "not present". (An entirely "not present" TSS is restartable.) Partially present TSS's can be avoided either by keeping the TSS's of such tasks present in memory, or by aligning TSS segments to reside entirely within a single 4K page (for TSS segments of 4K bytes or less).

B. A coprocessor operand wraps around the top of a 64K-byte segment or a 4G-byte segment, and spans three pages, and the page holding the middle portion of the operand is "not present." This condition can be avoided by starting **at a page boundary** any segments containing coprocessor operands if the segments are approximately 64K-200 bytes or larger (i.e. large enough for wraparound of the coprocessor operand to possibly occur).

Note that these conditions are avoided by using the operating system designs mentioned in this table.

2.9.8 Double Fault

A Double Fault (exception 8) results when the processor attempts to invoke an exception service routine for the segment exceptions (10, 11, 12 or 13), but in the process of doing so, detects an exception **other than** a Page Fault (exception 14).

A Double Fault (exception 8) will also be generated when the processor attempts to invoke the Page Fault (exception 14) service routine, and detects an exception other than a second Page Fault. In any functional system, the entire Page Fault service routine must remain "present" in memory.

When a Double Fault occurs, the 386 Microprocessor invokes the exception service routine for exception 8.

2.10 RESET AND INITIALIZATION

When the processor is initialized or Reset the registers have the values shown in Table 2-7. The 386 Microprocessor will then start executing instructions near the top of physical memory, at location FFFFFFF0H. When the first InterSegment Jump or Call is executed, address lines A20-31 will drop low for CS-relative memory cycles, and the 386 Microprocessor will only execute instructions in the lower one megabyte of physical memory. This allows the system designer to use a ROM at the top of physical memory to initialize the system and take care of Resets.

RESET forces the 386 Microprocessor to terminate all execution and local bus activity. No instruction execution or bus activity will occur as long as Reset is active. Between 350 and 450 CLK2 periods after Reset becomes inactive the 386 Microprocessor will start executing instructions at the top of physical memory.

Table 2-7. Register Values after Reset

Flag Word	UUUU0002H	Note 1
Machine Status Word (CR0)	UUUUUUU0H	Note 2
Instruction Pointer	0000FFF0H	
Code Segment	F000H	Note 3
Data Segment	0000H	
Stack Segment	0000H	
Extra Segment (ES)	0000H	
Extra Segment (FS)	0000H	
Extra Segment (GS)	0000H	
DX register	component and stepping ID	Note 5
All other registers	undefined	Note 4

NOTES:
1. EFLAG Register. The upper 14 bits of the EFLAGS register are undefined, VM (Bit 17) and RF (BIT) 16 are 0 as are all other defined flag bits.
2. CR0: (Machine Status Word). All of the defined fields in the CR0 are 0 (PG Bit 31, TS Bit 3, EM Bit 2, MP Bit 1, and PE Bit 0) except for ET Bit 4 (processor extension type). The ET Bit is set during Reset according to the type of Coprocessor in the system. If the coprocessor is an 80387 then ET will be 1, if the coprocessor is an 80287 or no coprocessor is present then ET will be 0. All other bits are undefined.
3. The Code Segment Register (CS) will have its Base Address set to FFFF0000H and Limit set to 0FFFFH.
4. All undefined bits are Intel Reserved and should not be used.
5. DX register always holds component and stepping identifier (see 5.7). EAX register holds self-test signature if self-test was requested (see 5.6).

2.11 TESTABILITY

2.11.1 Self-Test

The 386 Microprocessor has the capability to perform a self-test. The self-test checks the function of all of the Control ROM and most of the non-random logic of the part. Approximately one-half of the 386 Microprocessor can be tested during self-test.

Self-Test is initiated on the 386 Microprocessor when the RESET pin transitions from HIGH to LOW, and the BUSY# pin is low. The self-test takes about 2**19 clocks, or approximately 33 milliseconds with a 16 MHz 386 Microprocessor. At the completion of self-test the processor performs reset and begins normal operation. The part has successfully passed self-test if the contents of the EAX register are zero (0). If the results of EAX are not zero then the self-test has detected a flaw in the part.

2.11.2 TLB Testing

The 386 Microprocessor provides a mechanism for testing the Translation Lookaside Buffer (TLB) if desired. This particular mechanism is unique to the 386 Microprocessor and may not be continued in the same way in future processors. When testing the TLB paging must be turned off (PG = 0 in CR0) to enable the TLB testing hardware and avoid interference with the test data being written to the TLB.

There are two TLB testing operations: 1) write entries into the TLB, and, 2) perform TLB lookups. Two Test Registers, shown in Figure 2-12, are provided for the purpose of testing. TR6 is the "test command register", and TR7 is the "test data register". The fields within these registers are defined below.

C: This is the command bit. For a write into TR6 to cause an immediate write into the TLB entry, write a 0 to this bit. For a write into TR6 to cause an immediate TLB lookup, write a 1 to this bit.

Linear Address: This is the tag field of the TLB. On a TLB write, a TLB entry is allocated to this linear address and the rest of that TLB entry is set per the value of TR7 and the value just written into TR6. On a TLB lookup, the TLB is interrogated per this value and if one and only one TLB entry matches, the rest of the fields of TR6 and TR7 are set from the matching TLB entry.

Physical Address: This is the data field of the TLB. On a write to the TLB, the TLB entry allocated to the linear address in TR6 is set to this value. On a TLB lookup, the data field (physical address) from the TLB is read out to here.

PL: On a TLB write, PL = 1 causes the REP field of TR7 to select which of four associative blocks of the TLB is to be written, but PL = 0 allows the internal pointer in the paging unit to select which TLB block is written. On a TLB lookup, the PL bit indicates whether the lookup was a hit (PL gets set to 1) or a miss (PL gets reset to 0).

V: The valid bit for this TLB entry. All valid bits can also be cleared by writing to CR3.

D, D#: The dirty bit for/from the TLB entry.

U, U#: The user bit for/from the TLB entry.

W, W#: The writable bit for/from the TLB entry.

For D, U and W, both the attribute and its complement are provided as tag bits, to permit the option of a "don't care" on TLB lookups. The meaning of these pairs of bits is given in the following table:

X	X#	Effect During TLB Lookup	Value of Bit X after TLB Write
0	0	Miss All	Bit X Becomes Undefined
0	1	Match if X = 0	Bit X Becomes 0
1	0	Match if X = 1	Bit X Becomes 1
1	1	Match all	Bit X Becomes Undefined

For writing a TLB entry:

1. Write TR7 for the desired physical address, PL and REP values.
2. Write TR6 with the appropriate linear address, etc. (be sure to write C = 0 for "write" command).

For looking up (reading) a TLB entry:

1. Write TR6 with the appropriate linear address (be sure to write C = 1 for "lookup" command).
2. Read TR7 and TR6. If the PL bit in TR7 indicates a hit, then the other values reveal the TLB contents. If PL indicates a miss, then the other values in TR7 and TR6 are indeterminate.

2.12 DEBUGGING SUPPORT

The 386 Microprocessor provides several features which simplify the debugging process. The three categories of on-chip debugging aids are:

1) the code execution breakpoint opcode (0CCH),

2) the single-step capability provided by the TF bit in the flag register, and

3) the code and data breakpoint capability provided by the Debug Registers DR0-3, DR6, and DR7.

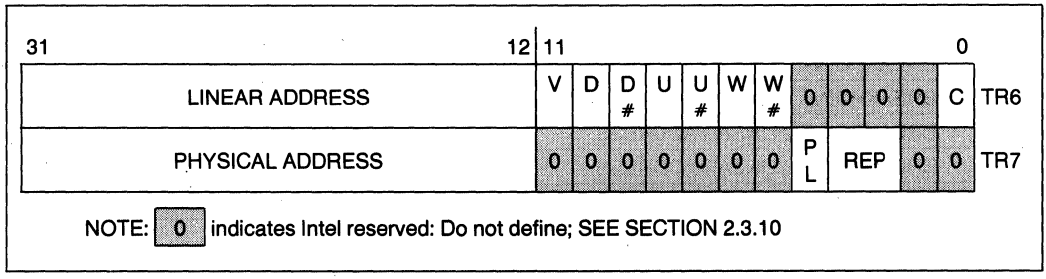


Figure 2-12. Test Registers

2.12.1 Breakpoint Instruction

A single-byte-opcode breakpoint instruction is available for use by software debuggers. The breakpoint opcode is 0CCh, and generates an exception 3 trap when executed. In typical use, a debugger program can "plant" the breakpoint instruction at all desired code execution breakpoints. The single-byte breakpoint opcode is an alias for the two-byte general software interrupt instruction, INT n, where n=3. The only difference between INT 3 (0CCh) and INT n is that INT 3 is never IOPL-sensitive but INT n is IOPL-sensitive in Protected Mode and Virtual 8086 Mode.

2.12.2 Single-Step Trap

If the single-step flag (TF, bit 8) in the EFLAG register is found to be set at the end of an instruction, a single-step exception occurs. The single-step exception is auto vectored to exception number 1. Precisely, exception 1 occurs as a trap after the instruction following the instruction which set TF. In typical practice, a debugger sets the TF bit of a flag register image on the debugger's stack. It then typically transfers control to the user program and loads the flag image with a signal instruction, the IRET instruction. The single-step trap occurs after executing one instruction of the user program.

Since the exception 1 occurs as a trap (that is, it occurs after the instruction has already executed), the CS:EIP pushed onto the debugger's stack points to the next unexecuted instruction of the program being debugged. An exception 1 handler, merely by ending with an IRET instruction, can therefore efficiently support single-stepping through a user program.

2.12.3 Debug Registers

The Debug Registers are an advanced debugging feature of the 386 Microprocessor. They allow data access breakpoints as well as code execution breakpoints. Since the breakpoints are indicated by on-chip registers, an instruction execution breakpoint can be placed in ROM code or in code shared by several tasks, neither of which can be supported by the INT3 breakpoint opcode.

The 386 Microprocessor contains six Debug Registers, providing the ability to specify up to four distinct breakpoints addresses, breakpoint control options, and read breakpoint status. Initially after reset, breakpoints are in the disabled state. Therefore, no breakpoints will occur unless the debug registers are programmed. Breakpoints set up in the Debug Registers are autovectored to exception number 1.

2.12.3.1 LINEAR ADDRESS BREAKPOINT REGISTERS (DR0–DR3)

Up to four breakpoint addresses can be specified by writing into Debug Registers DR0–DR3, shown in Figure 2-13. The breakpoint addresses specified are 32-bit linear addresses. 386 Microprocessor hardware continuously compares the linear breakpoint addresses in DR0–DR3 with the linear addresses generated by executing software (a linear address is the result of computing the effective address and adding the 32-bit segment base address). Note that if paging is not enabled the linear address equals the physical address. If paging is enabled, the linear address is translated to a physical 32-bit address by the on-chip paging unit. Regardless of whether paging is enabled or not, however, the breakpoint registers hold linear addresses.

2.12.3.2 DEBUG CONTROL REGISTER (DR7)

A Debug Control Register, DR7 shown in Figure 2-13, allows several debug control functions such as enabling the breakpoints and setting up other control options for the breakpoints. The fields within the Debug Control Register, DR7, are as follows:

LENi (breakpoint length specification bits)

A 2-bit LEN field exists for each of the four breakpoints. LEN specifies the length of the associated breakpoint field. The choices for data breakpoints are: 1 byte, 2 bytes, and 4 bytes. Instruction execu-

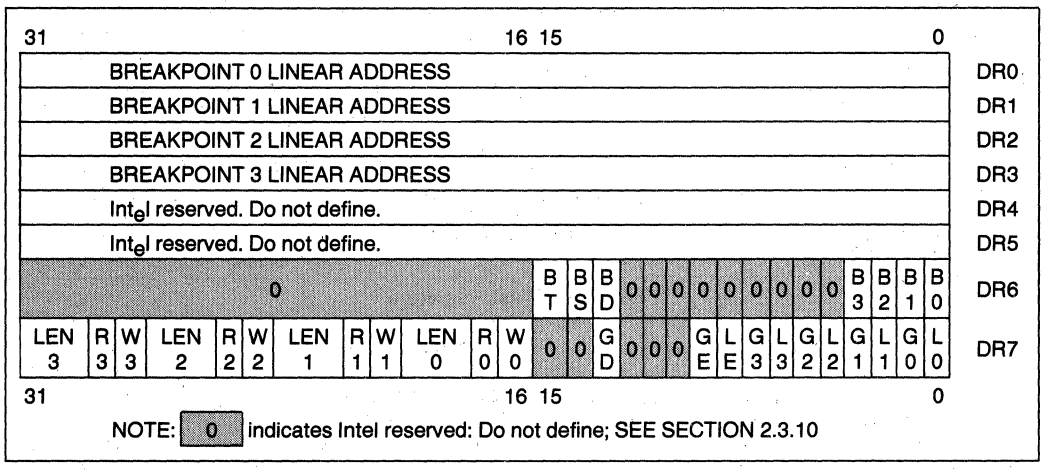


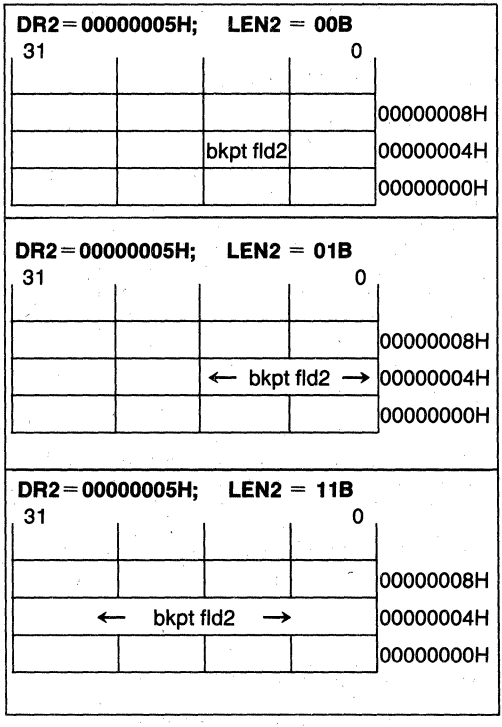
Figure 2-13. Debug Registers

tion breakpoints must have a length of 1 (LENi = 00). Encoding of the LENi field is as follows:

LENi Encoding	Breakpoint Field Width	Usage of Least Significant Bits in Breakpoint Address Register i, (i = 0 – 3)
00	1 byte	All 32-bits used to specify a single-byte breakpoint field.
01	2 bytes	A1–A31 used to specify a two-byte, word-aligned breakpoint field. A0 in Breakpoint Address Register is not used.
10	Undefined—do not use this encoding	
11	4 bytes	A2–A31 used to specify a four-byte, dword-aligned breakpoint field. A0 and A1 in Breakpoint Address Register are not used.

The LENi field controls the size of breakpoint field i by controlling whether all low-order linear address bits in the breakpoint address register are used to detect the breakpoint event. Therefore, all breakpoint fields are aligned; 2-byte breakpoint fields begin on Word boundaries, and 4-byte breakpoint fields begin on Dword boundaries.

The following is an example of various size breakpoint fields. Assume the breakpoint linear address in DR2 is 00000005H. In that situation, the following illustration indicates the region of the breakpoint field for lengths of 1, 2, or 4 bytes.



RWi (memory access qualifier bits)

A 2-bit RW field exists for each of the four breakpoints. The 2-bit RW field specifies the type of usage which must occur in order to activate the associated breakpoint.

RW Encoding	Usage Causing Breakpoint
00	Instruction execution only
01	Data writes only
10	Undefined—do not use this encoding
11	Data reads and writes only

RW encoding 00 is used to set up an instruction execution breakpoint. RW encodings 01 or 11 are used to set up write-only or read/write data breakpoints.

Note that **instruction execution breakpoints are taken as faults** (i.e. before the instruction executes), but **data breakpoints are taken as traps** (i.e. after the data transfer takes place).

Using LENi and RWi to Set Data Breakpoint i

A data breakpoint can be set up by writing the linear address into DRi (i = 0–3). For data breakpoints, RWi can = 01 (write-only) or 11 (write/read). LEN can = 00, 01, or 11.

If a data access entirely or partly falls within the data breakpoint field, the data breakpoint condition has occurred, and if the breakpoint is enabled, an exception 1 trap will occur.

Using LENi and RWi to Set Instruction Execution Breakpoint i

An instruction execution breakpoint can be set up by writing address of the beginning of the instruction (including prefixes if any) into DRi (i = 0–3). RWi must = 00 and LEN must = 00 for instruction execution breakpoints.

If the instruction beginning at the breakpoint address is about to be executed, the instruction execution breakpoint condition has occurred, and if the breakpoint is enabled, an exception 1 fault will occur before the instruction is executed.

Note that an instruction execution breakpoint address must be equal to the **beginning** byte address of an instruction (including prefixes) in order for the instruction execution breakpoint to occur.

GD (Global Debug Register access detect)

The Debug Registers can only be accessed in Real Mode or at privilege level 0 in Protected Mode. The GD bit, when set, provides extra protection against **any** Debug Register access even in Real Mode or at privilege level 0 in Protected Mode. This additional protection feature is provided to guarantee that a software debugger (or ICE™-386) can have full control over the Debug Register resources when required. The GD bit, when set, causes an exception 1 fault if an instruction attempts to read or write any Debug Register. The GD bit is then automatically cleared when the exception 1 handler is invoked, allowing the exception 1 handler free access to the debug registers.

GE and LE (Exact data breakpoint match, global and local)

If either GE or LE is set, any data breakpoint trap will be reported exactly after completion of the instruction that caused the operand transfer. Exact reporting is provided by forcing the 386 Microprocessor execution unit to wait for completion of data operand transfers before beginning execution of the next instruction.

If exact data breakpoint match is not selected, data breakpoints may not be reported until several instructions later or may not be reported at all. When enabling a data breakpoint, it is therefore recommended to enable the exact data breakpoint match.

When the 386 Microprocessor performs a task switch, the LE bit is cleared. Thus, the LE bit supports fast task switching out of tasks, that have enabled the exact data breakpoint match for their task-local breakpoints. The LE bit is cleared by the processor during a task switch, to avoid having exact data breakpoint match enabled in the new task. Note that exact data breakpoint match must be re-enabled under software control.

The 386 Microprocessor GE bit is unaffected during a task switch. The GE bit supports exact data breakpoint match that is to remain enabled during all tasks executing in the system.

Note that **instruction execution** breakpoints are always reported exactly, whether or not exact data breakpoint match is selected.

Gi and Li (breakpoint enable, global and local)

If either Gi or Li is set then the associated breakpoint (as defined by the linear address in DRi, the length in LENi and the usage criteria in RWi) is enabled. If either Gi or Li is set, and the 386 Microprocessor detects the ith breakpoint condition, then the exception 1 handler is invoked.

When the 386 Microprocessor performs a task switch to a new Task State Segment (TSS), all Li bits are cleared. Thus, the Li bits support fast task switching out of tasks that use some task-local

breakpoint registers. The Li bits are cleared by the processor during a task switch, to avoid spurious exceptions in the new task. Note that the breakpoints must be re-enabled under software control.

All 386 Microprocessor Gi bits are unaffected during a task switch. The Gi bits support breakpoints that are active in all tasks executing in the system.

2.12.3.3 DEBUG STATUS REGISTER (DR6)

A Debug Status Register, DR6 shown in Figure 2-13, allows the exception 1 handler to easily determine why it was invoked. Note the exception 1 handler can be invoked as a result of one of several events:

1) DR0 Breakpoint fault/trap.
2) DR1 Breakpoint fault/trap.
3) DR2 Breakpoint fault/trap.
4) DR3 Breakpoint fault/trap.
5) Single-step (TF) trap.
6) Task switch trap.
7) Fault due to attempted debug register access when GD = 1.

The Debug Status Register contains single-bit flags for each of the possible events invoking exception 1. Note below that some of these events are faults (exception taken before the instruction is executed), while other events are traps (exception taken after the debug events occurred).

The flags in DR6 are set by the hardware but never cleared by hardware. Exception 1 handler software should clear DR6 before returning to the user program to avoid future confusion in identifying the source of exception 1.

The fields within the Debug Status Register, DR6, are as follows:

Bi (debug fault/trap due to breakpoint 0–3)

Four breakpoint indicator flags, B0–B3, correspond one-to-one with the breakpoint registers in DR0–DR3. A flag Bi is set when the condition described by DRi, LENi, and RWi occurs.

If Gi or Li is set, and if the ith breakpoint is detected, the processor will invoke the exception 1 handler. The exception is handled as a fault if an instruction execution breakpoint occurred, or as a trap if a data breakpoint occurred.

IMPORTANT NOTE: A flag Bi is set whenever the hardware detects a match condition on **enabled** breakpoint i. Whenever a match is detected on at least one **enabled** breakpoint i, the hardware immediately sets all Bi bits corresponding to breakpoint conditions matching at that instant, whether enabled **or not.** Therefore, the exception 1 handler may see that multiple Bi bits are set, but only set Bi bits corresponding to **enabled** breakpoints (Li or Gi set) are **true** indications of why the exception 1 handler was invoked.

BD (debug fault due to attempted register access when GD bit set)

This bit is set if the exception 1 handler was invoked due to an instruction attempting to read or write to the debug registers when GD bit was set. If such an event occurs, then the GD bit is automatically cleared when the exception 1 handler is invoked, allowing handler access to the debug registers.

BS (debug trap due to single-step)

This bit is set if the exception 1 handler was invoked due to the TF bit in the flag register being set (for single-stepping). See section 2.12.2.

BT (debug trap due to task switch)

This bit is set if the exception 1 handler was invoked due to a task switch occurring to a task having a 386 Microprocessor TSS with the T bit set. (See Figure 4-15a). Note the task switch into the new task occurs normally, but before the first instruction of the task is executed, the exception 1 handler is invoked. With respect to the task switch operation, the operation is considered to be a trap.

2.12.3.4 USE OF RESUME FLAG (RF) IN FLAG REGISTER

The Resume Flag (RF) in the flag word can suppress an instruction execution breakpoint when the exception 1 handler returns to a user program at a user address which is also an instruction execution breakpoint. See section 2.3.3.

3. REAL MODE ARCHITECTURE

3.1 REAL MODE INTRODUCTION

When the processor is reset or powered up it is initialized in Real Mode. Real Mode has the same base architecture as the 8086, but allows access to the 32-bit register set of the 386 Microprocessor. The addressing mechanism, memory size, interrupt handling, are all identical to the Real Mode on the 80286.

All of the 386 Microprocessor instructions are available in Real Mode (except those instructions

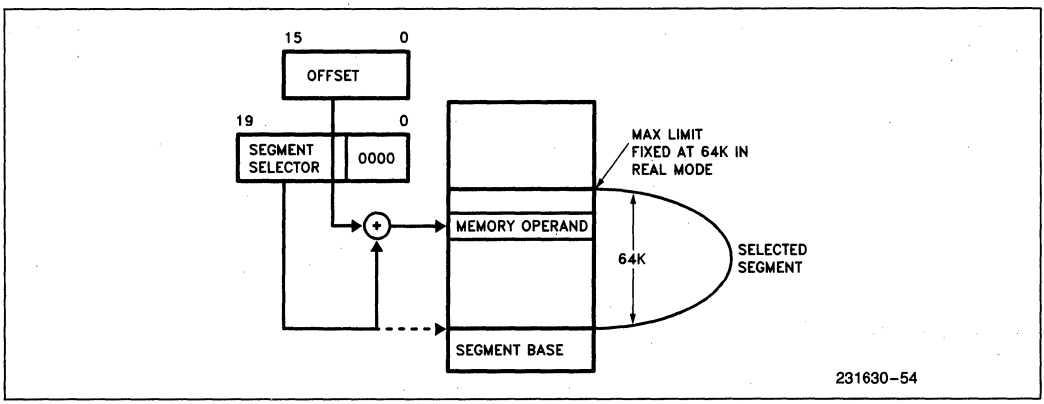


Figure 3-1. Real Address Mode Addressing

listed in 4.6.4). The default operand size in Real Mode is 16-bits, just like the 8086. In order to use the 32-bit registers and addressing modes, override prefixes must be used. In addition, the segment size on the 386 Microprocessor in Real Mode is 64K bytes so 32-bit effective addresses must have a value less the 0000FFFFH. The primary purpose of Real Mode is to set up the processor for Protected Mode Operation.

The LOCK prefix on the 386 Microprocessor, even in Real Mode, is more restrictive than on the 80286. This is due to the addition of paging on the 386 Microprocessor in Protected Mode and Virtual 8086 Mode. Paging makes it impossible to guarantee that repeated string instructions can be LOCKed. The 386 Microprocessor can't require that all pages holding the string be physically present in memory. Hence, a Page Fault (exception 14) might have to be taken during the repeated string instruction. Therefore the LOCK prefix can't be supported during repeated string instructions.

These are the only instruction forms where the LOCK prefix is legal on the 386 Microprocessor:

Opcode	Operands (Dest, Source)
BIT Test and SET/RESET/COMPLEMENT	Mem, Reg/immed
XCHG	Reg, Mem
XCHG	Mem, Reg
ADD, OR, ADC, SBB, AND, SUB, XOR	Mem, Reg/immed
NOT, NEG, INC, DEC	Mem

An exception 6 will be generated if a LOCK prefix is placed before any instruction form or opcode not listed above. The LOCK prefix allows indivisible read/modify/write operations on memory operands using the instructions above. For example, even the ADD Reg, Mem is not LOCKable, because the Mem operand is not the destination (and therefore no memory read/modify/operation is being performed).

Since, on the 386 Microprocessor, repeated string instructions are not LOCKable, it is not possible to LOCK the bus for a long period of time. Therefore, the LOCK prefix is not IOPL-sensitive on the 386 Microprocessor. The LOCK prefix can be used at any privilege level, but only on the instruction forms listed above.

3.2 MEMORY ADDRESSING

In Real Mode the maximum memory size is limited to 1 megabyte. Thus, only address lines A2–A19 are active. (Exception, the high address lines A20–A31 are high during CS-relative memory cycles until an intersegment jump or call is executed (see section 2.10)).

Since paging is not allowed in Real Mode the linear addresses are the same as physical addresses. Physical addresses are formed in Real Mode by adding the contents of the appropriate segment register which is shifted left by four bits to an effective address. This addition results in a physical address from 00000000H to 0010FFEFH. This is compatible with 80286 Real Mode. Since segment registers are shifted left by 4 bits this implies that Real Mode segments always start on 16 byte boundaries.

All segments in Real Mode are exactly 64K bytes long, and may be read, written, or executed. The 386 Microprocessor will generate an exception 13 if a data operand or instruction fetch occurs past the end of a segment. (i.e. if an operand has an offset greater than FFFFH, for example a word with a low byte at FFFFH and the high byte at 0000H.)

Segments may be overlapped in Real Mode. Thus, if a particular segment does not use all 64K bytes another segment can be overlayed on top of the unused portion of the previous segment. This allows the programmer to minimize the amount of physical memory needed for a program.

3.3 RESERVED LOCATIONS

There are two fixed areas in memory which are reserved in Real address mode: system initialization area and the interrupt table area. Locations 00000H through 003FFH are reserved for interrupt vectors. Each one of the 256 possible interrupts has a 4-byte jump vector reserved for it. Locations FFFFFFF0H through FFFFFFFFH are reserved for system initialization.

3.4 INTERRUPTS

Many of the exceptions shown in Table 2-5 and discussed in section 2.9 are not applicable to Real Mode operation, in particular exceptions 10, 11, 14, will not happen in Real Mode. Other exceptions have slightly different meanings in Real Mode; Table 3-1 identifies these exceptions.

3.5 SHUTDOWN AND HALT

The HLT instruction stops program execution and prevents the processor from using the local bus until restarted. Either NMI, INTR with interrupts enabled (IF = 1), or RESET will force the 386 Microprocessor out of halt. If interrupted, the saved CS:IP will point to the next instruction after the HLT.

Shutdown will occur when a severe error is detected that prevents further processing. In Real Mode, shutdown can occur under two conditions:

An interrupt or an exception occur (Exceptions 8 or 13) and the interrupt vector is larger than the Interrupt Descriptor Table (i.e. There is not an interrupt handler for the interrupt).

A CALL, INT or PUSH instruction attempts to wrap around the stack segment when SP is not even. (e.g. pushing a value on the stack when SP = 0001 resulting in a stack segment greater than FFFFH)

An NMI input can bring the processor out of shutdown if the Interrupt Descriptor Table limit is large enough to contain the NMI interrupt vector (at least 0017H) and the stack has enough room to contain the vector and flag information (i.e. SP is greater than 0005H). Otherwise shutdown can only be exited via the RESET input.

4. PROTECTED MODE ARCHITECTURE

4.1 INTRODUCTION

The complete capabilities of the 386 Microprocessor are unlocked when the processor operates in Protected Virtual Address Mode (Protected Mode). Protected Mode vastly increases the linear address space to four gigabytes (2^{32} bytes) and allows the running of virtual memory programs of almost unlimited size (64 terabytes or 2^{46} bytes). In addition Protected Mode allows the 386 Microprocessor to run all of the existing 8086 and 80286 software, while providing a sophisticated memory management and a hardware-assisted protection mechanism. Protected Mode allows the use of additional instructions especially optimized for supporting multitasking operating systems. The base architecture of the 386 Microprocessor remains the same, the registers, instructions, and addressing modes described in the previous sections are retained. The main difference between Protected Mode, and Real Mode from a programmer's view is the increased address space, and a different addressing mechanism.

Table 3-1

Function	Interrupt Number	Related Instructions	Return Address Location
Interrupt table limit too small	8	INT Vector is not within table limit	Before Instruction
CS, DS, ES, FS, GS Segment overrun exception	13	Word memory reference beyond offset = FFFFH. An attempt to execute past the end of CS segment.	Before Instruction
SS Segment overrun exception	12	Stack Reference beyond offset = FFFFH	Before Instruction

4.2 ADDRESSING MECHANISM

Like Real Mode, Protected Mode uses two components to form the logical address, a 16-bit selector is used to determine the linear base address of a segment, the base address is added to a 32-bit effective address to form a 32-bit linear address. The linear address is then either used as the 32-bit physical address, or if paging is enabled the paging mechanism maps the 32-bit linear address into a 32-bit physical address.

The difference between the two modes lies in calculating the base address. In Protected Mode the selector is used to specify an index into an operating system defined table (see Figure 4-1). The table contains the 32-bit base address of a given segment. The physical address is formed by adding the base address obtained from the table to the offset.

Paging provides an additional memory management mechanism which operates only in Protected Mode. Paging provides a means of managing the very large segments of the 386 Microprocessor. As such, paging operates beneath segmentation. The paging mechanism translates the protected linear address which comes from the segmentation unit into a physical address. Figure 4-2 shows the complete 386 Microprocessor addressing mechanism with paging enabled.

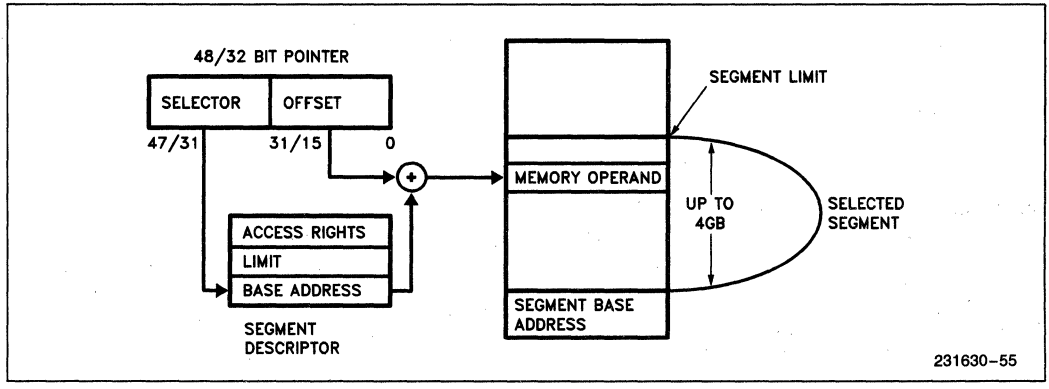


Figure 4-1. Protected Mode Addressing

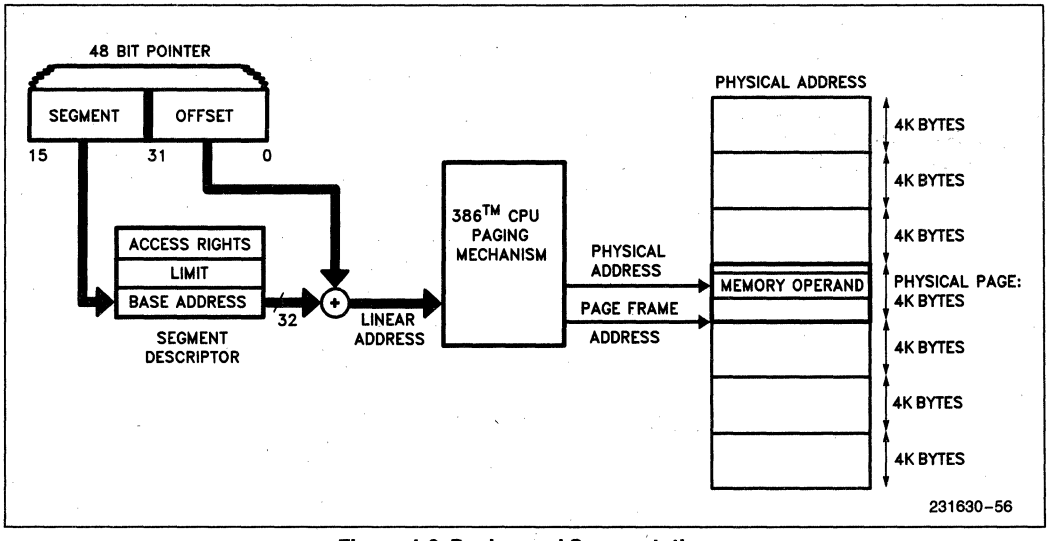


Figure 4-2. Paging and Segmentation

4.3 SEGMENTATION

4.3.1 Segmentation Introduction

Segmentation is one method of memory management. Segmentation provides the basis for protection. Segments are used to encapsulate regions of memory which have common attributes. For example, all of the code of a given program could be contained in a segment, or an operating system table may reside in a segment. All information about a segment is stored in an 8 byte data structure called a descriptor. All of the descriptors in a system are contained in tables recognized by hardware.

4.3.2 Terminology

The following terms are used throughout the discussion of descriptors, privilege levels and protection:

PL: Privilege Level—One of the four hierarchical privilege levels. Level 0 is the most privileged level and level 3 is the least privileged. More privileged levels are numerically smaller than less privileged levels.

RPL: Requestor Privilege Level—The privilege level of the original supplier of the selector. RPL is determined by the **least two** significant bits of a selector.

DPL: Descriptor Privilege Level—This is the least privileged level at which a task may access that descriptor (and the segment associated with that descriptor). Descriptor Privilege Level is determined by bits 6:5 in the Access Right Byte of a descriptor.

CPL: Current Privilege Level—The privilege level at which a task is currently executing, which equals the privilege level of the code segment being executed. CPL can also be determined by examining the lowest 2 bits of the CS register, except for conforming code segments.

EPL: Effective Privilege Level—The effective privilege level is the least privileged of the RPL and DPL. Since smaller privilege level **values** indicate greater privilege, EPL is the numerical maximum of RPL and DPL.

Task: One instance of the execution of a program. Tasks are also referred to as processes.

4.3.3 Descriptor Tables

4.3.3.1 DESCRIPTOR TABLES INTRODUCTION

The descriptor tables define all of the segments which are used in an 386 Microprocessor system. There are three types of tables on the 386 Microprocessor which hold descriptors: the Global Descriptor Table, Local Descriptor Table, and the Interrupt Descriptor Table. All of the tables are variable length memory arrays. They can range in size between 8 bytes and 64K bytes. Each table can hold up to 8192 8 byte descriptors. The upper 13 bits of a selector are used as an index into the descriptor table. The tables have registers associated with them which hold the 32-bit linear base address, and the 16-bit limit of each table.

Each of the tables has a register associated with it the GDTR, LDTR, and the IDTR (see Figure 4-3). The LGDT, LLDT, and LIDT instructions, load the base and limit of the Global, Local, and Interrupt Descriptor Tables, respectively, into the appropriate register. The SGDT, SLDT, and SIDT store the base and limit values. These tables are manipulated by the operating system. Therefore, the load descriptor table instructions are privileged instructions.

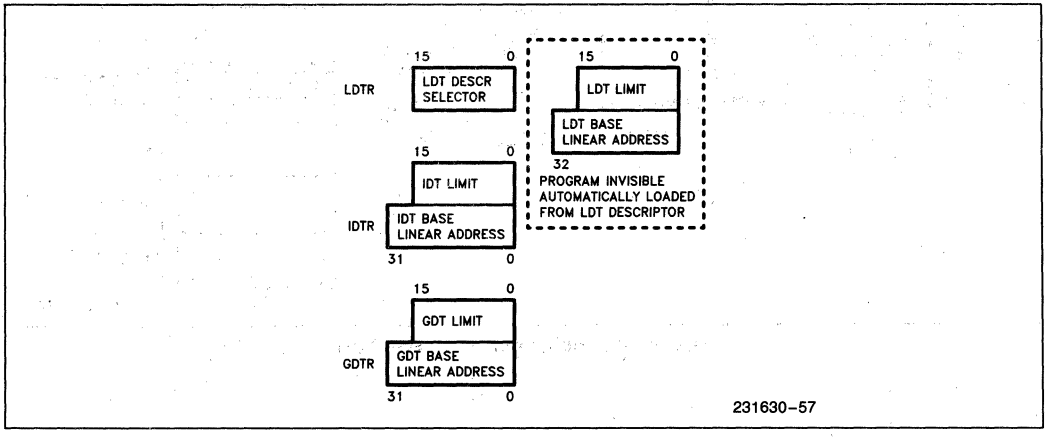


Figure 4-3. Descriptor Table Registers

4.3.3.2 GLOBAL DESCRIPTOR TABLE

The Global Descriptor Table (GDT) contains descriptors which are possibly available to all of the tasks in a system. The GDT can contain any type of segment descriptor except for descriptors which are used for servicing interrupts (i.e. interrupt and trap descriptors). Every 386 Microprocessor system contains a GDT. Generally the GDT contains code and data segments used by the operating systems and task state segments, and descriptors for the LDTs in a system.

The first slot of the Global Descriptor Table corresponds to the null selector and is not used. The null selector defines a null pointer value.

4.3.3.3 LOCAL DESCRIPTOR TABLE

LDTs contain descriptors which are associated with a given task. Generally, operating systems are designed so that each task has a separate LDT. The LDT may contain only code, data, stack, task gate, and call gate descriptors. LDTs provide a mechanism for isolating a given task's code and data segments from the rest of the operating system, while the GDT contains descriptors for segments which are common to all tasks. A segment cannot be accessed by a task if its segment descriptor does not exist in either the current LDT or the GDT. This provides both isolation and protection for a task's segments, while still allowing global data to be shared among tasks.

Unlike the 6 byte GDT or IDT registers which contain a base address and limit, the visible portion of the LDT register contains only a 16-bit selector. This selector refers to a Local Descriptor Table descriptor in the GDT.

4.3.3.4 INTERRUPT DESCRIPTOR TABLE

The third table needed for 386 Microprocessor systems is the Interrupt Descriptor Table. (See Figure 4-4.) The IDT contains the descriptors which point to the location of up to 256 interrupt service routines. The IDT may contain only task gates, interrupt gates, and trap gates. The IDT should be at least 256 bytes in size in order to hold the descriptors for the 32 Intel Reserved Interrupts. Every interrupt used by a system must have an entry in the IDT. The IDT entries are referenced via INT instructions, external interrupt vectors, and exceptions. (See 2.9 **Interrupts).**

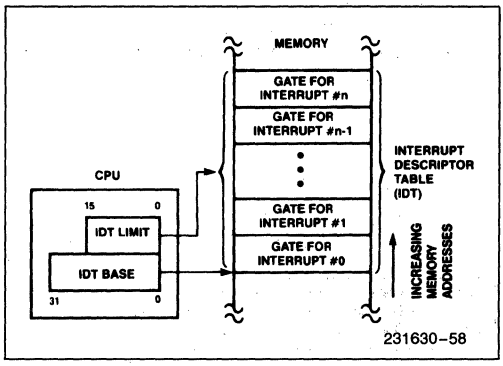
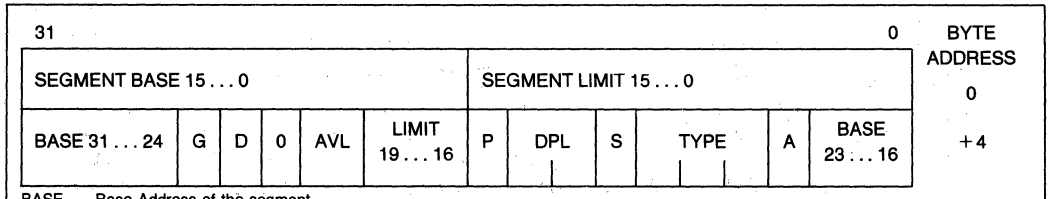


Figure 4-4. Interrupt Descriptor Table Register Use

4.3.4 Descriptors

4.3.4.1 DESCRIPTOR ATTRIBUTE BITS

The object to which the segment selector points to is called a descriptor. Descriptors are eight byte



BASE	Base Address of the segment
LIMIT	The length of the segment
P	Present Bit 1 = Present 0 = Not Present
DPL	Descriptor Privilege Level 0-3
S	Segment Descriptor 0 = System Descriptor 1 = Code or Data Segment Descriptor
TYPE	Type of Segment
A	Accessed Bit
G	Granularity Bit 1 = Segment length is page granular 0 = Segment length is byte granular
D	Default Operation Size (recognized in code segment descriptors only) 1 = 32-bit segment 0 = 16-bit segment
0	Bit must be zero (0) for compatibility with future processors
AVL	Available field for user or OS

NOTE:
In a maximum-size segment (ie. a segment with G = 1 and segment limit 19...0 = FFFFFH), the lowest 12 bits of the segment base should be zero (ie. segment base 11...000 = 000H).

Figure 4-5. Segment Descriptors

quantities which contain attributes about a given region of linear address space (i.e. a segment). These attributes include the 32-bit base linear address of the segment, the 20-bit length and granularity of the segment, the protection level, read, write or execute privileges, the default size of the operands (16-bit or 32-bit), and the type of segment. All of the attribute information about a segment is contained in 12 bits in the segment descriptor. Figure 4-5 shows the general format of a descriptor. All segments on the 386 Microprocessor have three attribute fields in common: the **P** bit, the **DPL** bit, and the **S** bit. The Present **P** bit is 1 if the segment is loaded in physical memory, if P=0 then any attempt to access this segment causes a not present exception (exception 11). The Descriptor Privilege Level **DPL** is a two-bit field which specifies the protection level 0–3 associated with a segment.

The 386 Microprocessor has two main categories of segments system segments and non-system segments (for code and data). The segment **S** bit in the segment descriptor determines if a given segment is a system segment or a code or data segment. If the S bit is 1 then the segment is either a code or data segment, if it is 0 then the segment is a system segment.

4.3.4.2 386™ CPU CODE, DATA DESCRIPTORS (S=1)

Figure 4-6 shows the general format of a code and data descriptor and Table 4-1 illustrates how the bits in the Access Rights Byte are interpreted.

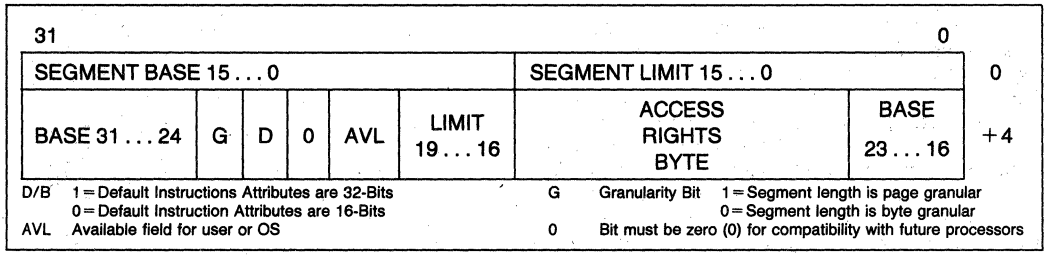


Figure 4-6. Segment Descriptors

Table 4-1. Access Rights Byte Definition for Code and Data Descriptions

	Bit Position	Name	Function	
	7	Present (P)	P = 1 Segment is mapped into physical memory. P = 0 No mapping to physical memory exits, base and limit are not used.	
	6–5	Descriptor Privilege Level (DPL)	Segment privilege attribute used in privilege tests.	
	4	Segment Descriptor (S)	S = 1 Code or Data (includes stacks) segment descriptor S = 0 System Segment Descriptor or Gate Descriptor	
Type Field Definition	3	Executable (E)	E = 0 Descriptor type is data segment:	If Data Segment (S = 1, E = 0)
	2	Expansion Direction (ED)	ED = 0 Expand up segment, offsets must be ≤ limit. ED = 1 Expand down segment, offsets must be > limit.	
	1	Writeable (W)	W = 0 Data segment may not be written into. W = 1 Data segment may be written into.	
	3	Executable (E)	E = 1 Descriptor type is code segment:	If Code Segment (S = 1, E = 1)
	2	Conforming (C)	C = 1 Code segment may only be executed when CPL ≥ DPL and CPL remains unchanged.	
	1	Readable (R)	R = 0 Code segment may not be read. R = 1 Code segment may be read.	
	0	Accessed (A)	A = 0 Segment has not been accessed. A = 1 Segment selector has been loaded into segment register or used by selector test instructions.	

Code and data segments have several descriptor fields in common. The accessed **A** bit is set whenever the processor accesses a descriptor. The **A** bit is used by operating systems to keep usage statistics on a given segment. The **G** bit, or granularity bit, specifies if a segment length is byte-granular or page-granular. 386 Microprocessor segments can be one megabyte long with byte granularity (G=0) or four gigabytes with page granularity (G=1), (i.e., 2^{20} pages each page is 4K bytes in length). The granularity is totally unrelated to paging. A 386 Microprocessor system can consist of segments with byte granularity, and page granularity, whether or not paging is enabled.

The executable **E** bit tells if a segment is a code or data segment. A code segment (E=1, S=1) may be execute-only or execute/read as determined by the Read **R** bit. Code segments are execute only if R=0, and execute/read if R=1. Code segments may never be written into.

NOTE:
Code segments may be modified via aliases. Aliases are writeable data segments which occupy the same range of linear address space as the code segment.

The **D** bit indicates the default length for operands and effective addresses. If D=1 then 32-bit operands and 32-bit addressing modes are assumed. If D=0 then 16-bit operands and 16-bit addressing modes are assumed. Therefore all existing 80286 code segments will execute on the 386 Microprocessor assuming the D bit is set 0.

Another attribute of code segments is determined by the conforming **C** bit. Conforming segments, C=1, can be executed and shared by programs at different privilege levels. (See section 4.4 **Protection.**)

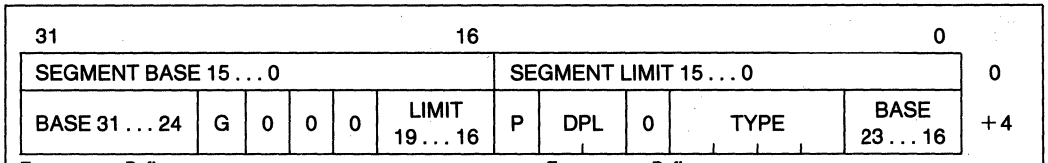
Segments identified as data segments (E=0, S=1) are used for two types of 386 Microprocessor segments: stack and data segments. The expansion direction **(ED)** bit specifies if a segment expands downward (stack) or upward (data). If a segment is a stack segment all offsets must be greater than the segment limit. On a data segment all offsets must be less than or equal to the limit. In other words, stack segments start at the base linear address plus the maximum segment limit and grow down to the base linear address plus the limit. On the other hand, data segments start at the base linear address and expand to the base linear address plus limit.

The write **W** bit controls the ability to write into a segment. Data segments are read-only if W=0. The stack segment must have W=1.

The **B** bit controls the size of the stack pointer register. If B=1, then PUSHes, POPs, and CALLs all use the 32-bit ESP register for stack references and assume an upper limit of FFFFFFFFH. If B=0, stack instructions all use the 16-bit SP register and assume an upper limit of FFFFH.

4.3.4.3 SYSTEM DESCRIPTOR FORMATS

System segments describe information about operating system tables, tasks, and gates. Figure 4-7 shows the general format of system segment descriptors, and the various types of system segments. 386 Microprocessor system descriptors contain a 32-bit base linear address and a 20-bit segment limit. 80286 system descriptors have a 24-bit base address and a 16-bit segment limit. 80286 system descriptors are identified by the upper 16 bits being all zero.



Type	Defines	Type	Defines
0	Invalid	8	Invalid
1	Available 80286 TSS	9	Available 386™ CPU TSS
2	LDT	A	Undefined (Intel Reserved)
3	Busy 80286 TSS	B	Busy 386™ CPU TSS
4	80286 Call Gate	C	386™ CPU Call Gate
5	Task Gate (for 80286 or 386™ CPU Task)	D	Undefined (Intel Reserved)
6	80286 Interrupt Gate	E	386™ CPU Interrupt Gate
7	80286 Trap Gate	F	386™ CPU Trap Gate

Figure 4-7. System Segments Descriptors

4.3.4.4 LDT DESCRIPTORS (S=0, TYPE=2)

LDT descriptors (S=0 TYPE=2) contain information about Local Descriptor Tables. LDTs contain a table of segment descriptors, unique to a particular task. Since the instruction to load the LDTR is only available at privilege level 0, the DPL field is ignored. LDT descriptors are only allowed in the Global Descriptor Table (GDT).

4.3.4.5 TSS DESCRIPTORS (S=0, TYPE=1, 3, 9, B)

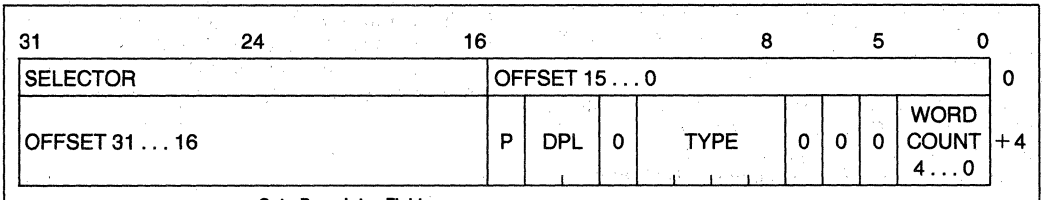
A Task State Segment (TSS) descriptor contains information about the location, size, and privilege level of a Task State Segment (TSS). A TSS in turn is a special fixed format segment which contains all the state information for a task and a linkage field to permit nesting tasks. The TYPE field is used to indicate whether the task is currently BUSY (i.e. on a chain of active tasks) or the TSS is available. The TYPE field also indicates if the segment contains a 80286 or a 386 Microprocessor TSS. The Task Register (TR) contains the selector which points to the current Task State Segment.

4.3.4.6 GATE DESCRIPTORS (S=0, TYPE=4-7, C, F)

Gates are used to control access to entry points within the target code segment. The various types of gate descriptors are **call** gates, **task** gates, **interrupt** gates, and **trap** gates. Gates provide a level of indirection between the source and destination of the control transfer. This indirection allows the processor to automatically perform protection checks. It also allows system designers to control entry points to the operating system. Call gates are used to change privilege levels (see section 4.4 **Protection**), task gates are used to perform a task switch, and interrupt and trap gates are used to specify interrupt service routines.

Figure 4-8 shows the format of the four types of gate descriptors. Call gates are primarily used to transfer program control to a more privileged level. The call gate descriptor consists of three fields: the access byte, a long pointer (selector and offset) which points to the start of a routine and a word count which specifies how many parameters are to be copied from the caller's stack to the stack of the called routine. The word count field is only used by call gates when there is a change in the privilege level, other types of gates ignore the word count field.

Interrupt and trap gates use the destination selector and destination offset fields of the gate descriptor as a pointer to the start of the interrupt or trap handler routines. The difference between interrupt gates and trap gates is that the interrupt gate disables interrupts (resets the IF bit) while the trap gate does not.



Gate Descriptor Fields

Name	Value	Description
Type	4	80286 call gate
	5	Task gate (for 80286 or 386™ CPU task)
	6	80286 interrupt gate
	7	80286 trap gate
	C	386™ CPU call gate
	E	386™ CPU interrupt gate
	F	386™ CPU trap gate
P	0	Descriptor contents are not valid
	1	Descriptor contents are valid

DPL—least privileged level at which a task may access the gate. WORD COUNT 0-31—the number of parameters to copy from caller's stack to the called procedure's stack. The parameters are 32-bit quantities for 386™ CPU gates, and 16-bit quantities for 80286 gates.

DESTINATION SELECTOR	16-bit selector	Selector to the target code segment or Selector to the target task state segment for task gate
DESTINATION OFFSET	offset 16-bit 80286 32-bit 386™ CPU	Entry point within the target code segment

Figure 4-8. Gate Descriptor Formats

Task gates are used to switch tasks. Task gates may only refer to a task state segment (see section 4.4.6 **Task Switching**) therefore only the destination selector portion of a task gate descriptor is used, and the destination offset is ignored.

Exception 13 is generated when a destination selector does not refer to a correct descriptor type, i.e., a code segment for an interrupt, trap or call gate, a TSS for a task gate.

The access byte format is the same for all gate descriptors. P=1 indicates that the gate contents are valid. P=0 indicates the contents are not valid and causes exception 11 if referenced. DPL is the descriptor privilege level and specifies when this descriptor may be used by a task (see section 4.4 **Protection**). The S field, bit 4 of the access rights byte, must be 0 to indicate a system control descriptor. The type field specifies the descriptor type as indicated in Figure 4-8.

4.3.4.7 DIFFERENCES BETWEEN 386™ MICROPROCESSOR AND 80286 DESCRIPTORS

In order to provide operating system compatibility between the 80286 and 386 Microprocessor, the 386 Microprocessor supports all of the 80286 segment descriptors. Figure 4-9 shows the general format of an 80286 system segment descriptor. The only differences between 80286 and 386 Microprocessor descriptor formats are that the values of the type fields, and the limit and base address fields have been expanded for the 386 Microprocessor. The 80286 system segment descriptors contained a 24-bit base address and 16-bit limit, while the 386 Microprocessor system segment descriptors have a 32-bit base address, a 20-bit limit field, and a granularity bit.

By supporting 80286 system segments the 386 Microprocessor is able to execute 80286 application programs on a 386 Microprocessor operating system. This is possible because the processor automatically understands which descriptors are 80286-style descriptors and which descriptors are 386 Microprocessor-style descriptors. In particular, if the upper word of a descriptor is zero, then that descriptor is a 80286-style descriptor.

The only other differences between 80286-style descriptors and 386 Microprocessor descriptors is the interpretation of the word count field of call gates and the B bit. The word count field specifies the number of 16-bit quantities to copy for 80286 call gates and 32-bit quantities for 386 Microprocessor call gates. The B bit controls the size of PUSHes when using a call gate; if B=0 PUSHes are 16 bits, if B=1 PUSHes are 32 bits.

4.3.4.8 SELECTOR FIELDS

A selector in Protected Mode has three fields: Local or Global Descriptor Table Indicator (TI), Descriptor Entry Index (Index), and Requestor (the selector's) Privilege Level (RPL) as shown in Figure 4-10. The TI bits select one of two memory-based tables of descriptors (the Global Descriptor Table or the Local Descriptor Table). The Index selects one of 8K descriptors in the appropriate descriptor table. The RPL bits allow high speed testing of the selector's privilege attributes.

4.3.4.9 SEGMENT DESCRIPTOR CACHE

In addition to the selector value, every segment register has a segment descriptor cache register associated with it. Whenever a segment register's contents are changed, the 8-byte descriptor associated with that selector is automatically loaded (cached) on the chip. Once loaded, all references to that segment use the cached descriptor information instead of reaccessing the descriptor. The contents of the descriptor cache are not visible to the programmer. Since descriptor caches only change when a segment register is changed, programs which modify the descriptor tables must reload the appropriate segment registers after changing a descriptor's value.

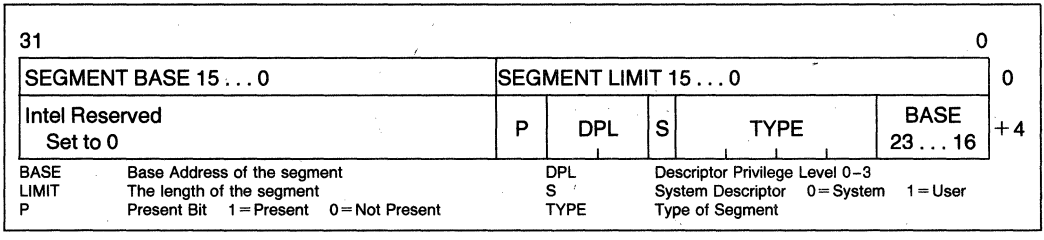


Figure 4-9. 80286 Code and Data Segment Descriptors

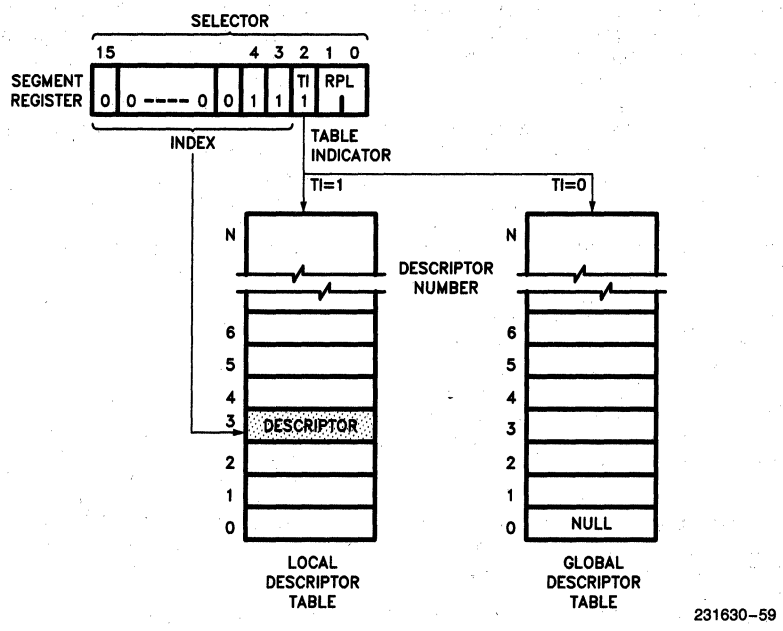
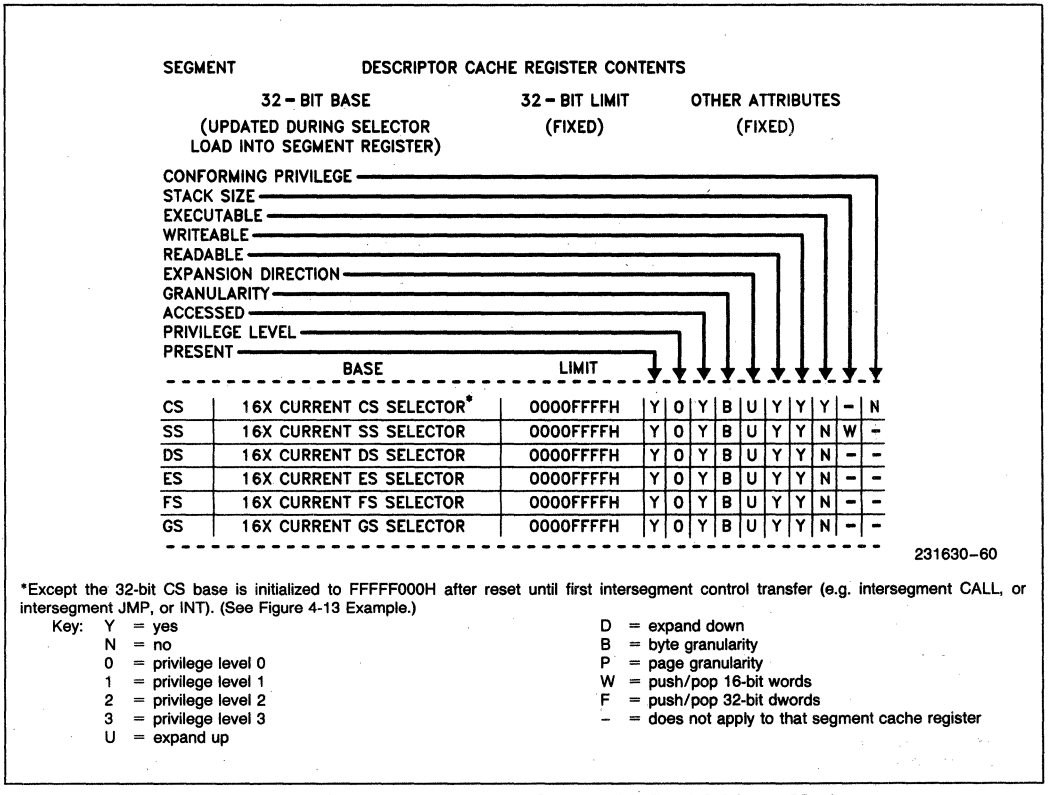


Figure 4-10. Example Descriptor Selection

4.3.4.10 SEGMENT DESCRIPTOR REGISTER SETTINGS

The contents of the segment descriptor cache vary depending on the mode the 386 Microprocessor is operating in. When operating in Real Address Mode, the segment base, limit, and other attributes within the segment cache registers are defined as shown in Figure 4-11. For compatiblity with the 8086 architecture, the base is set to sixteen times the current selector value, the limit is fixed at 0000FFFFH, and the attributes are fixed so as to indicate the segment is present and fully usable. In Real Address Mode, the internal "privilege level" is always fixed to the highest level, level 0, so I/O and other privileged opcodes may be executed.

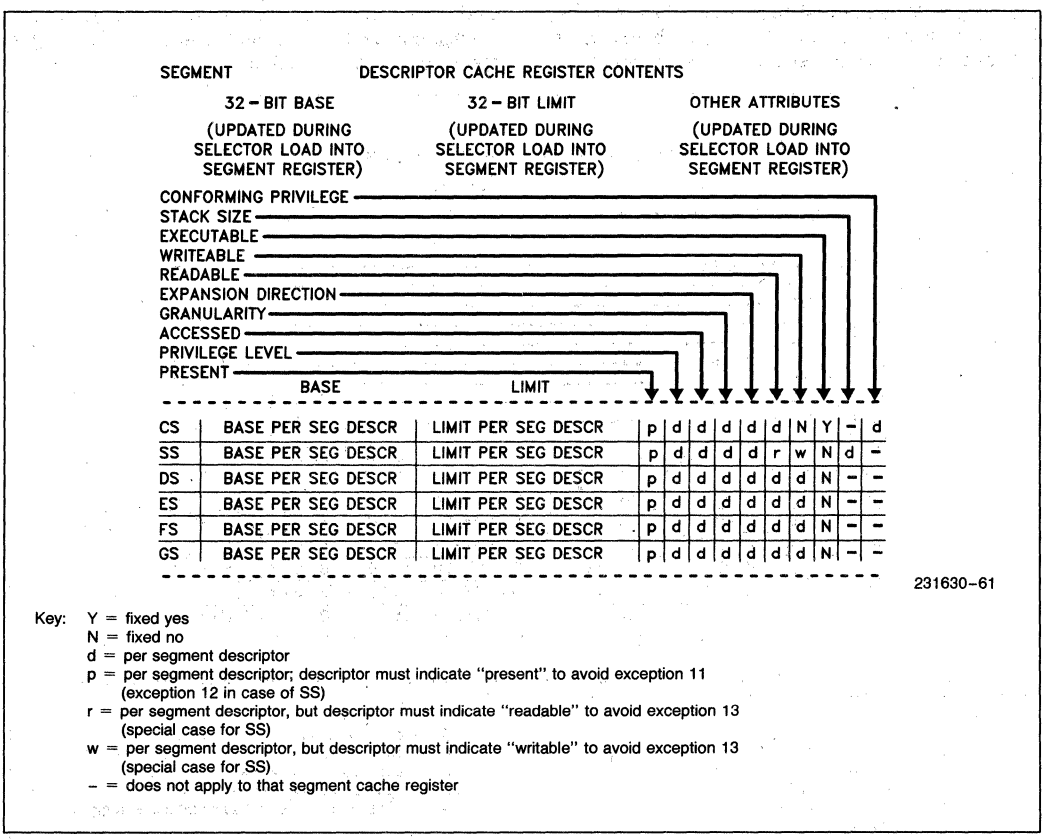


*Except the 32-bit CS base is initialized to FFFFF000H after reset until first intersegment control transfer (e.g. intersegment CALL, or intersegment JMP, or INT). (See Figure 4-13 Example.)

Key:
- Y = yes
- N = no
- 0 = privilege level 0
- 1 = privilege level 1
- 2 = privilege level 2
- 3 = privilege level 3
- U = expand up
- D = expand down
- B = byte granularity
- P = page granularity
- W = push/pop 16-bit words
- F = push/pop 32-bit dwords
- – = does not apply to that segment cache register

Figure 4-11. Segment Descriptor Caches for Real Address Mode (Segment Limit and Attributes are Fixed)

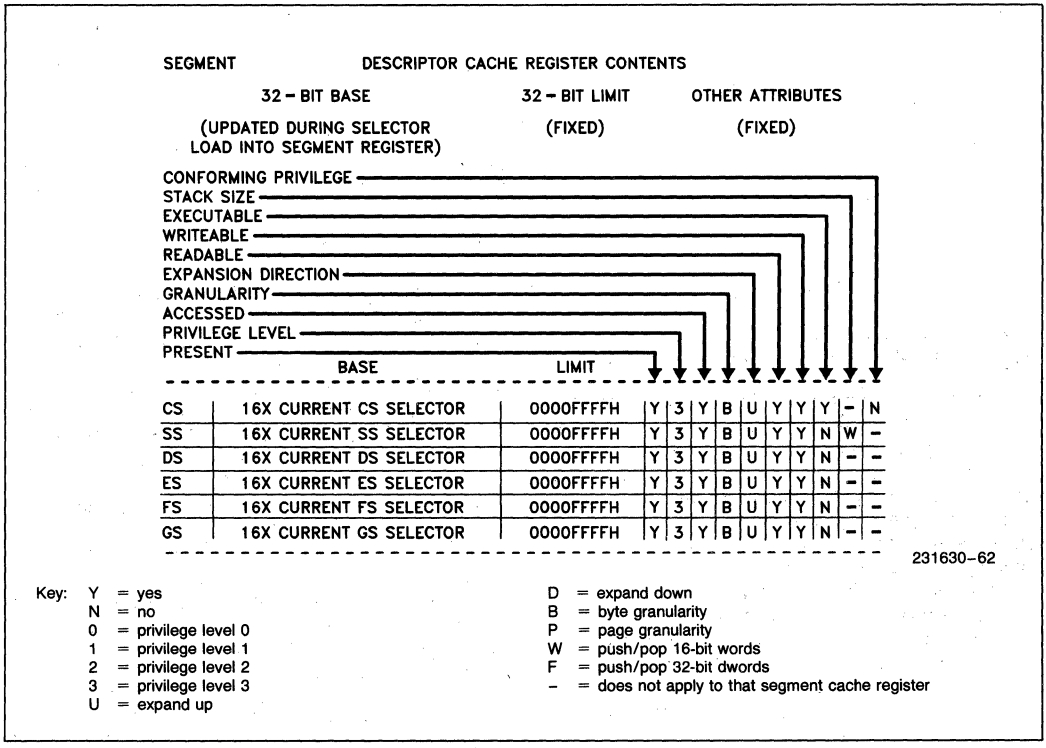
When operating in Protected Mode, the segment base, limit, and other attributes within the segment cache registers are defined as shown in Figure 4-12. In Protected Mode, each of these fields are defined according to the contents of the segment descriptor indexed by the selector value loaded into the segment register.



Key: Y = fixed yes
N = fixed no
d = per segment descriptor
p = per segment descriptor; descriptor must indicate "present" to avoid exception 11 (exception 12 in case of SS)
r = per segment descriptor, but descriptor must indicate "readable" to avoid exception 13 (special case for SS)
w = per segment descriptor, but descriptor must indicate "writable" to avoid exception 13 (special case for SS)
– = does not apply to that segment cache register

Figure 4-12. Segment Descriptor Caches for Protected Mode (Loaded per Descriptor)

When operating in a Virtual 8086 Mode within the Protected Mode, the segment base, limit, and other attributes within the segment cache registers are defined as shown in Figure 4-13. For compatibility with the 8086 architecture, the base is set to sixteen times the current selector value, the limit is fixed at 0000FFFFH, and the attributes are fixed so as to indicate the segment is present and fully usable. The virtual program executes at lowest privilege level, level 3, to allow trapping of all IOPL-sensitive instructions and level-0-only instructions.



Key:
- Y = yes
- N = no
- 0 = privilege level 0
- 1 = privilege level 1
- 2 = privilege level 2
- 3 = privilege level 3
- U = expand up
- D = expand down
- B = byte granularity
- P = page granularity
- W = push/pop 16-bit words
- F = push/pop 32-bit dwords
- – = does not apply to that segment cache register

Figure 4-13. Segment Descriptor Caches for Virtual 8086 Mode within Protected Mode (Segment Limit and Attributes are Fixed)

4.4 PROTECTION

4.4.1 Protection Concepts

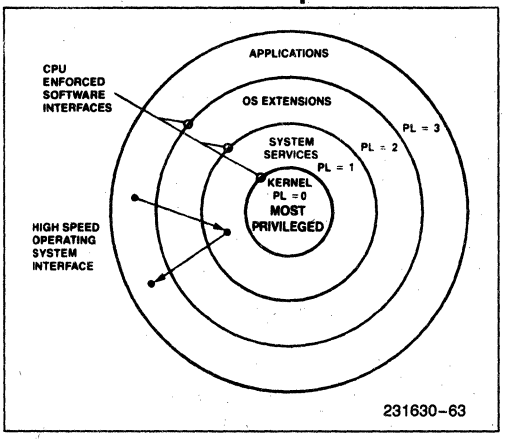


Figure 4-14. Four-Level Hierachical Protection

The 386 Microprocessor has four levels of protection which are optimized to support the needs of a multi-tasking operating system to isolate and protect user programs from each other and the operating system. The privilege levels control the use of privileged instructions, I/O instructions, and access to segments and segment descriptors. Unlike traditional microprocessor-based systems where this protection is achieved only through the use of complex external hardware and software the 386 Microprocessor provides the protection as part of its integrated Memory Management Unit. The 386 Microprocessor offers an additional type of protection on a page basis, when paging is enabled (See section 4.5.3 **Page Level Protection**).

The four-level hierarchical privilege system is illustrated in Figure 4-14. It is an extension of the user/supervisor privilege mode commonly used by minicomputers and, in fact, the user/supervisor mode is fully supported by the 386 Microprocessor paging mechanism. The privilege levels (PL) are numbered 0 through 3. Level 0 is the most privileged or trusted level.

4.4.2 Rules of Privilege

The 386 Microprocessor controls access to both data and procedures between levels of a task, according to the following rules.

- Data stored in a segment with privilege level **p** can be accessed only by code executing at a privilege level at least as privileged as **p**.
- A code segment/procedure with privilege level **p** can only be called by a task executing at the same or a lesser privilege level than **p**.

4.4.3 Privilege Levels

4.4.3.1 TASK PRIVILEGE

At any point in time, a task on the 386 Microprocessor always executes at one of the four privilege levels. The Current Privilege Level (CPL) specifies the task's privilege level. A task's CPL may only be changed by control transfers through gate descriptors to a code segment with a different privilege level. (See section 4.4.4 **Privilege Level Transfers**) Thus, an application program running at PL = 3 may call an operating system routine at PL = 1 (via a gate) which would cause the task's CPL to be set to 1 until the operating system routine was finished.

4.4.3.2 SELECTOR PRIVILEGE (RPL)

The privilege level of a selector is specified by the RPL field. The RPL is the two least significant bits of the selector. The selector's RPL is only used to establish a less trusted privilege level than the current privilege level for the use of a segment. This level is called the task's effective privilege level (EPL). The EPL is defined as being the least privileged (i.e. numerically larger) level of a task's CPL and a selector's RPL. Thus, if selector's RPL = 0 then the CPL always specifies the privilege level for making an access using the selector. On the other hand if RPL = 3 then a selector can only access segments at level 3 regardless of the task's CPL. The RPL is most commonly used to verify that pointers passed to an operating system procedure do not access data that is of higher privilege than the procedure that originated the pointer. Since the originator of a selector can specify any RPL value, the Adjust RPL (ARPL) instruction is provided to force the RPL bits to the originator's CPL.

4.4.3.3 I/O PRIVILEGE AND I/O PERMISSION BITMAP

The I/O privilege level (IOPL, a 2-bit field in the EFLAG register) defines the least privileged level at which I/O instructions can be unconditionally performed. I/O instructions can be unconditionally performed when CPL ≤ IOPL. (The I/O instructions are IN, OUT, INS, OUTS, REP INS, and REP OUTS.) When CPL > IOPL, and the current task is associated with a 286 TSS, attempted I/O instructions cause an exception 13 fault. When CPL > IOPL, and the current task is associated with a 386 Microprocessor TSS, the I/O Permission Bitmap (part of a 386 Microprocessor TSS) is consulted on whether I/O to the port is allowed, or an exception 13 fault is to be generated instead. For diagrams of the I/O Permission Bitmap, refer to Figures 4-15a and 4-15b. For further information on how the I/O Permission Bitmap is used in Protected Mode or in Virtual 8086

Mode, refer to section 4.6.4 Protection and I/O Permission Bitmap.

The I/O privilege level (IOPL) also affects whether several other instructions can be executed or cause an exception 13 fault instead. These instructions are called "IOPL-sensitive" instructions and they are CLI and STI. (Note that the LOCK prefix is *not* IOPL-sensitive on the 386 Microprocessor.)

The IOPL also affects whether the IF (interrupts enable flag) bit can be changed by loading a value into the EFLAGS register. When CPL ≤ IOPL, then the IF bit can be changed by loading a new value into the EFLAGS register. When CPL > IOPL, the IF bit cannot be changed by a new value POP'ed into (or otherwise loaded into) the EFLAGS register; the IF bit merely remains unchanged and no exception is generated.

Table 4-2. Pointer Test Instructions

Instruction	Operands	Function
ARPL	Selector, Register	Adjust Requested Privilege Level: adjusts the RPL of the selector to the numeric maximum of current selector RPL value and the RPL value in the register. Set zero flag if selector RPL was changed.
VERR	Selector	VERify for Read: sets the zero flag if the segment referred to by the selector can be read.
VERW	Selector	VERify for Write: sets the zero flag if the segment referred to by the selector can be written.
LSL	Register, Selector	Load Segment Limit: reads the segment limit into the register if privilege rules and descriptor type allow. Set zero flag if successful.
LAR	Register, Selector	Load Access Rights: reads the descriptor access rights byte into the register if privilege rules allow. Set zero flag if successful.

4.4.3.4 PRIVILEGE VALIDATION

The 386 Microprocessor provides several instructions to speed pointer testing and help maintain system integrity by verifying that the selector value refers to an appropriate segment. Table 4-2 summarizes the selector validation procedures available for the 386 Microprocessor.

This pointer verification prevents the common problem of an application at PL = 3 calling a operating systems routine at PL = 0 and passing the operating system routine a "bad" pointer which corrupts a data structure belonging to the operating system. If the operating system routine uses the ARPL instruction to ensure that the RPL of the selector has no greater privilege than that of the caller, then this problem can be avoided.

4.4.3.5 DESCRIPTOR ACCESS

There are basically two types of segment accesses: those involving code segments such as control transfers, and those involving data accesses. Determining the ability of a task to access a segment involves the type of segment to be accessed, the instruction used, the type of descriptor used and CPL, RPL, and DPL as described above.

Any time an instruction loads data segment registers (DS, ES, FS, GS) the 386 Microprocessor makes protection validation checks. Selectors loaded in the DS, ES, FS, GS registers must refer only to data segments or readable code segments. The data access rules are specified in section 4.2.2 **Rules of Privilege**. The only exception to those rules is readable conforming code segments which can be accessed at any privilege level.

Finally the privilege validation checks are performed. The CPL is compared to the EPL and if the EPL is more privileged than the CPL an exception 13 (general protection fault) is generated.

The rules regarding the stack segment are slightly different than those involving data segments. Instructions that load selectors into SS must refer to data segment descriptors for writeable data segments. The DPL and RPL must equal the CPL. All other descriptor types or a privilege level violation will cause exception 13. A stack not present fault causes exception 12. Note that an exception 11 is used for a not-present code or data segment.

4.4.4 Privilege Level Transfers

Inter-segment control transfers occur when a selector is loaded in the CS register. For a typical system most of these transfers are simply the result of a call

Table 4-3. Descriptor Types Used for Control Transfer

Control Transfer Types	Operation Types	Descriptor Referenced	Descriptor Table
Intersegment within the same privilege level	JMP, CALL, RET, IRET*	Code Segment	GDT/LDT
Intersegment to the same or higher privilege level Interrupt within task may change CPL	CALL	Call Gate	GDT/LDT
	Interrupt Instruction, Exception, External Interrupt	Trap or Interrupt Gate	IDT
Intersegment to a lower privilege level (changes task CPL)	RET, IRET*	Code Segment	GDT/LDT
	CALL, JMP	Task State Segment	GDT
Task Switch	CALL, JMP	Task Gate	GDT/LDT
	IRET** Interrupt Instruction, Exception, External Interrupt	Task Gate	IDT

*NT (Nested Task bit of flag register) = 0
**NT (Nested Task bit of flag register) = 1

or a jump to another routine. There are five types of control transfers which are summarized in Table 4-3. Many of these transfers result in a privilege level transfer. Changing privilege levels is done only via control transfers, by using gates, task switches, and interrupt or trap gates.

Control transfers can only occur if the operation which loaded the selector references the correct descriptor type. Any violation of these descriptor usage rules will cause an exception 13 (e.g. JMP through a call gate, or IRET from a normal subroutine call).

In order to provide further system security, all control transfers are also subject to the privilege rules.

The privilege rules require that:

— Privilege level transitions can only occur via gates.

— JMPs can be made to a non-conforming code segment with the same privilege or to a conforming code segment with greater or equal privilege.

— CALLs can be made to a non-conforming code segment with the same privilege or via a gate to a more privileged level.

— Interrupts handled within the task obey the same privilege rules as CALLs.

— Conforming Code segments are accessible by privilege levels which are the same or less privileged than the conforming-code segment's DPL.

— Both the requested privilege level (RPL) in the selector pointing to the gate and the task's CPL must be of equal or greater privilege than the gate's DPL.

— The code segment selected in the gate must be the same or more privileged than the task's CPL.

— Return instructions that do not switch tasks can only return control to a code segment with same or less privilege.

— Task switches can be performed by a CALL, JMP, or INT which references either a task gate or task state segment who's DPL is less privileged or the same privilege as the old task's CPL.

Any control transfer that changes CPL within a task causes a change of stacks as a result of the privilege level change. The initial values of SS:ESP for privilege levels 0, 1, and 2 are retained in the task state segment (see section 4.4.6 **Task Switching**). During a JMP or CALL control transfer, the new stack pointer is loaded into the SS and ESP registers and the previous stack pointer is pushed onto the new stack.

When RETurning to the original privilege level, use of the lower-privileged stack is restored as part of the RET or IRET instruction operation. For subroutine calls that pass parameters on the stack and cross privilege levels, a fixed number of words (as specified in the gate's word count field) are copied from the previous stack to the current stack. The inter-segment RET instruction with a stack adjustment value will correctly restore the previous stack pointer upon return.

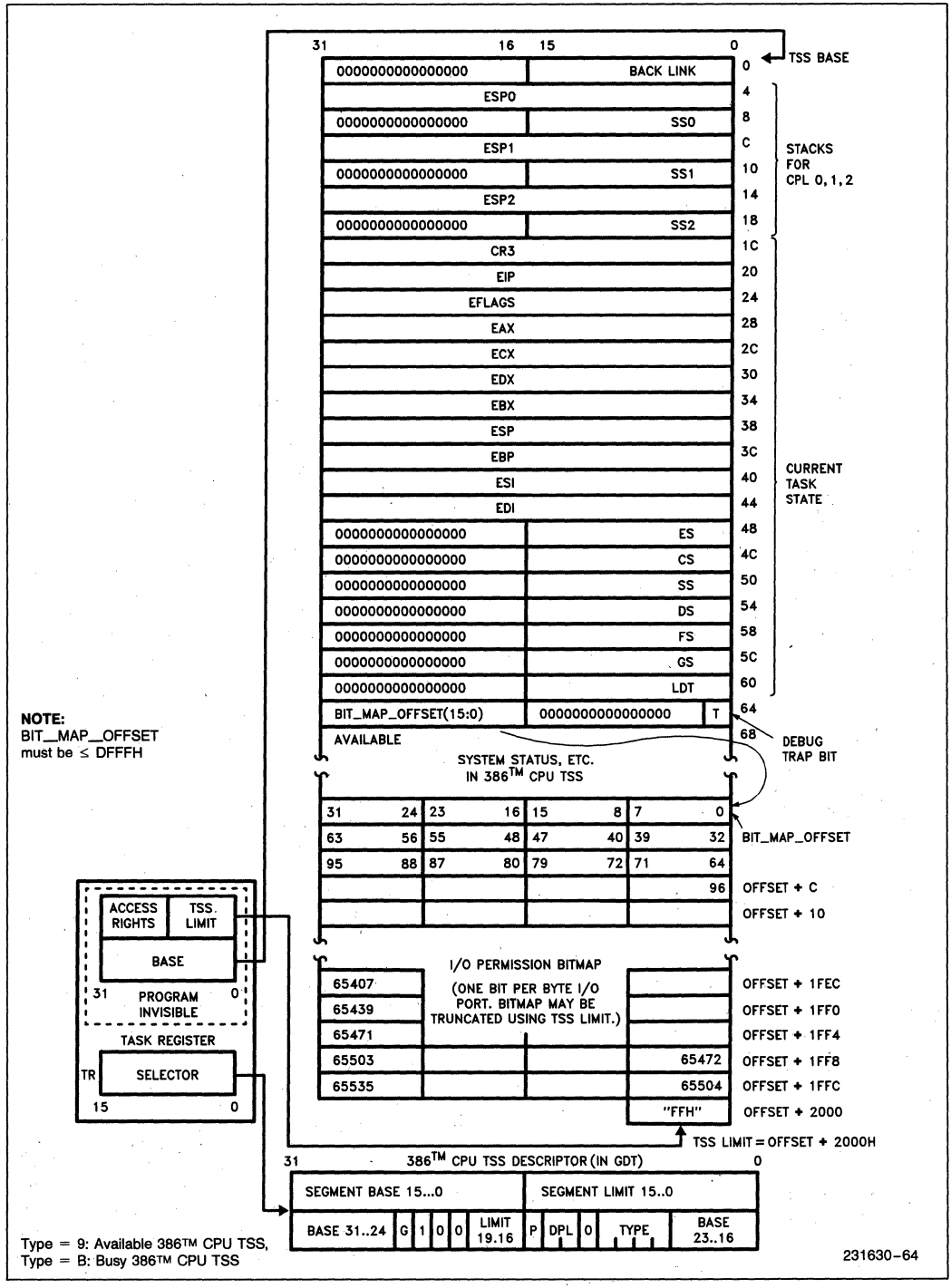


Figure 4-15a. 386™ Microprocessor TSS and TSS Registers

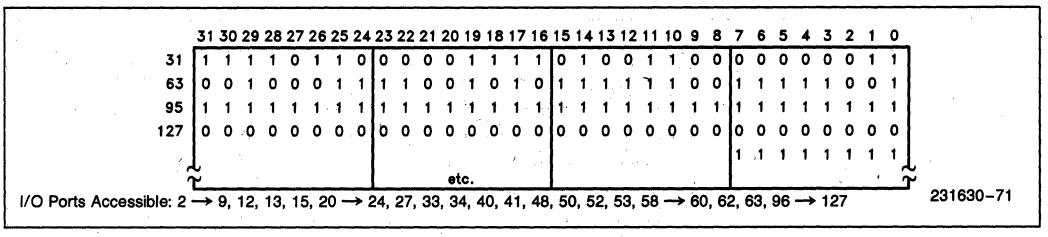


Figure 4-15b. Sample I/O Permission Bit Map

4.4.5 Call Gates

Gates provide protected, indirect CALLs. One of the major uses of gates is to provide a secure method of privilege transfers within a task. Since the operating system defines all of the gates in a system, it can ensure that all gates only allow entry into a few trusted procedures (such as those which allocate memory, or perform I/O).

Gate descriptors follow the data access rules of privilege; that is, gates can be accessed by a task if the EPL, is equal to or more privileged than the gate descriptor's DPL. Gates follow the control transfer rules of privilege and therefore may only transfer control to a more privileged level.

Call Gates are accessed via a CALL instruction and are syntactically identical to calling a normal subroutine. When an inter-level 386 Microprocessor call gate is activated, the following actions occur.

1. Load CS:EIP from gate check for validity
2. SS is pushed zero-extended to 32 bits
3. ESP is pushed
4. Copy Word Count 32-bit parameters from the old stack to the new stack
5. Push Return address on stack

The procedure is identical for 80286 Call gates, except that 16-bit parameters are copied and 16-bit registers are pushed.

Interrupt Gates and Trap gates work in a similar fashion as the call gates, except there is no copying of parameters. The only difference between Trap and Interrupt gates is that control transfers through an Interrupt gate disable further interrupts (i.e. the IF bit is set to 0), and Trap gates leave the interrupt status unchanged.

4.4.6 Task Switching

A very important attribute of any multi-tasking/multi-user operating systems is its ability to rapidly switch between tasks or processes. The 386 Microprocessor directly supports this operation by providing a task switch instruction in hardware. The 386 Microprocessor task switch operation saves the entire state of the machine (all of the registers, address space, and a link to the previous task), loads a new execution state, performs protection checks, and commences execution in the new task, in about 17 microseconds. Like transfer of control via gates, the task switch operation is invoked by executing an inter-segment JMP or CALL instruction which refers to a Task State Segment (TSS), or a task gate descriptor in the GDT or LDT. An INT n instruction, exception, trap, or external interrupt may also invoke the task switch operation if there is a task gate descriptor in the associated IDT descriptor slot.

The TSS descriptor points to a segment (see Figure 4-15) containing the entire 386 Microprocessor execution state while a task gate descriptor contains a TSS selector. The 386 Microprocessor supports both 80286 and 386 Microprocessor style TSSs. Figure 4-16 shows a 80286 TSS. The limit of a 386 Microprocessor TSS must be greater than 0064H (002BH for a 80286 TSS), and can be as large as 4 Gigabytes. In the additional TSS space, the operating system is free to store additional information such as the reason the task is inactive, time the task has spent running, and open files belong to the task.

Each task must have a TSS associated with it. The current TSS is identified by a special register in the 386 Microprocessor called the Task State Segment Register (TR). This register contains a selector referring to the task state segment descriptor that defines the current TSS. A hidden base and limit register associated with TR are loaded whenever TR is loaded with a new selector. Returning from a task is accomplished by the IRET instruction. When IRET is executed, control is returned to the task which was interrupted. The current executing task's state is saved in the TSS and the old task state is restored from its TSS.

Several bits in the flag register and machine status word (CR0) give information about the state of a task which are useful to the operating system. The Nested Task (NT) (bit 14 in EFLAGS) controls the function of the IRET instruction. If NT = 0, the IRET instruction performs the regular return; when NT = 1, IRET performs a task switch operation back to the previous task. The NT bit is set or reset in the following fashion:

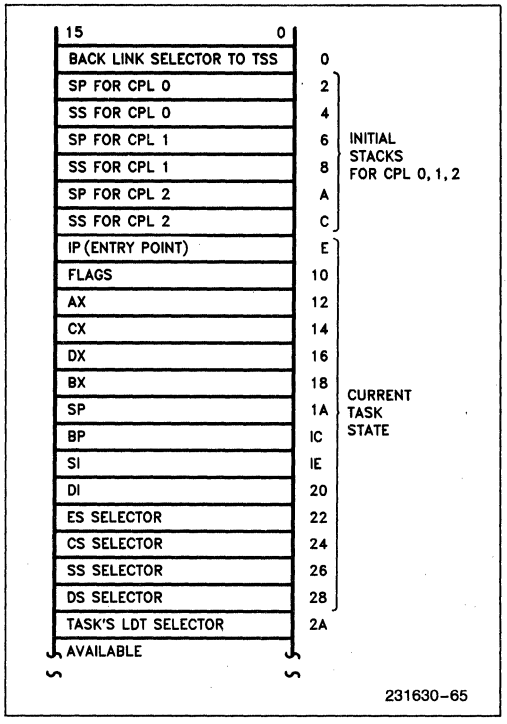


Figure 4-16. 80286 TSS

When a CALL or INT instruction initiates a task switch, the new TSS will be marked busy and the back link field of the new TSS set to the old TSS selector. The NT bit of the new task is set by CALL or INT initiated task switches. An interrupt that does not cause a task switch will clear NT. (The NT bit will be restored after execution of the interrupt handler) NT may also be set or cleared by POPF or IRET instructions.

The 386 Microprocessor task state segment is marked busy by changing the descriptor type field from TYPE 9H to TYPE BH. An 80286 TSS is marked busy by changing the descriptor type field from TYPE 1 to TYPE 3. Use of a selector that references a busy task state segment causes an exception 13.

The Virtual Mode (VM) bit 17 is used to indicate if a task, is a virtual 8086 task. If VM = 1, then the tasks will use the Real Mode addressing mechanism. The virtual 8086 environment is only entered and exited via a task switch (see section 4.6 **Virtual Mode**).

The coprocessor's state is not automatically saved when a task switch occurs, because the incoming task may not use the coprocessor. The Task Switched (TS) Bit (bit 3 in the CR0) helps deal with the coprocessor's state in a multi-tasking environment. Whenever the 386 Microprocessor switches tasks, it sets the TS bit. The 386 Microprocessor detects the first use of a processor extension instruction after a task switch and causes the processor extension not available exception 7. The exception handler for exception 7 may then decide whether to save the state of the coprocessor. A processor extension not present exception (7) will occur when attempting to execute an ESC or WAIT instruction if the Task Switched and Monitor coprocessor extension bits are both set (i.e. TS = 1 and MP = 1).

The **T** bit in the 386 Microprocessor TSS indicates that the processor should generate a debug exception when switching to a task. If T = 1 then upon entry to a new task a debug exception 1 will be generated.

4.4.7 Initialization and Transition to Protected Mode

Since the 386 Microprocessor begins executing in Real Mode immediately after RESET it is necessary to initialize the system tables and registers with the appropriate values.

The GDT and IDT registers must refer to a valid GDT and IDT. The IDT should be at least 256 bytes long, and GDT must contain descriptors for the initial code, and data segments. Figure 4-17 shows the tables and Figure 4-18 the descriptors needed for a simple Protected Mode 386 Microprocessor system. It has a single code and single data/stack segment each four gigabytes long and a single privilege level PL = 0.

The actual method of enabling Protected Mode is to load CR0 with the PE bit set, via the MOV CR0, R/M instruction. This puts the 386 Microprocessor in Protected Mode.

After enabling Protected Mode, the next instruction should execute an intersegment JMP to load the CS register and flush the instruction decode queue. The final step is to load all of the data segment registers with the initial selector values.

An alternate approach to entering Protected Mode which is especially appropriate for multi-tasking operating systems, is to use the built in task-switch to load all of the registers. In this case the GDT would contain two TSS descriptors in addition to the code and data descriptors needed for the first task. The first JMP instruction in Protected Mode would jump to the TSS causing a task switch and loading all of the registers with the values stored in the TSS. The Task State Segment Register should be initialized to point to a valid TSS descriptor since a task switch saves the state of the current task in a task state segment.

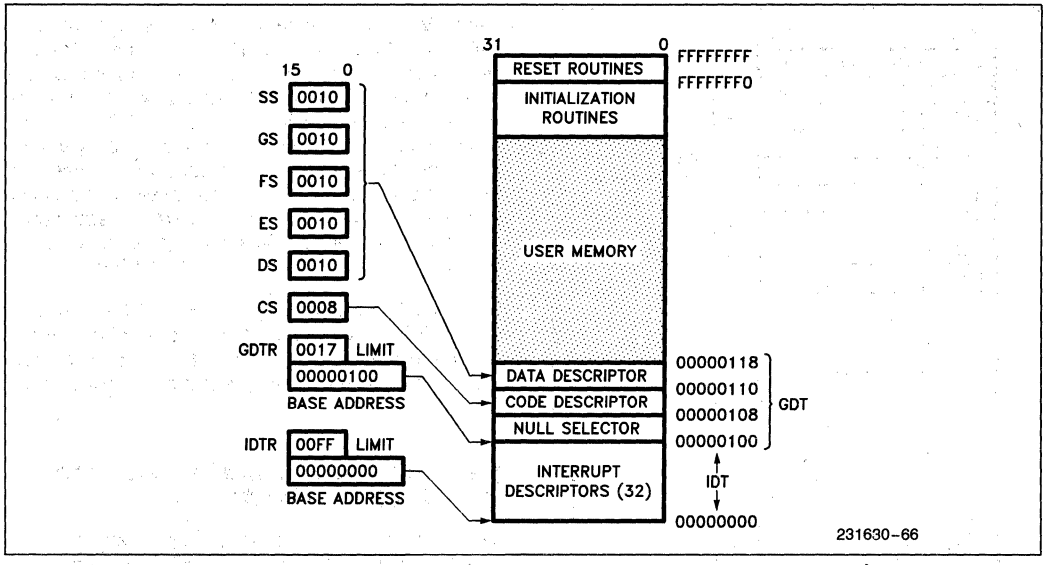


Figure 4-17. Simple Protected System

DATA DESCRIPTOR	SEGMENT BASE 15 . . . 0 0118 (H)								SEGMENT LIMIT 15 . . . 0 FFFF (H)								
2	BASE 31 . . . 24 00 (H)	G 1	D 1	0	0	LIMIT 19.16 F (H)			1	0	0	1	0	0	1	0	BASE 23 . . . 16 00 (H)
CODE DESCRIPTOR	SEGMENT BASE 15 . . . 0 0118 (H)								SEGMENT LIMIT 15 . . . 0 FFFF (H)								
1	BASE 31 . . . 24 00 (H)	G 1	D 1	0	0	LIMIT 19.16 F (H)			1	0	0	1	1	0	1	0	BASE 23 . . . 16 00 (H)
	NULL DESCRIPTOR																
0																	
	31 24 16								15 8 0								

Figure 4-18. GDT Descriptors for Simple System

4.4.8 Tools for Building Protected Systems

In order to simplify the design of a protected multitasking system, Intel provides a tool which allows the system designer an easy method of constructing the data structures needed for a Protected Mode 386 Microprocessor system. This tool is the builder BLD-386™. BLD-386 lets the operating system writer specify all of the segment descriptors discussed in the previous sections (LDTs, IDTs, GDTs, Gates, and TSSs) in a high-level language.

4.5 PAGING

4.5.1 Paging Concepts

Paging is another type of memory management useful for virtual memory multitasking operating systems. Unlike segmentation which modularizes programs and data into variable length segments, paging divides programs into multiple uniform size pages. Pages bear no direct relation to the logical

structure of a program. While segment selectors can be considered the logical "name" of a program module or data structure, a page most likely corresponds to only a portion of a module or data structure.

By taking advantage of the locality of reference displayed by most programs, only a small number of pages from each active task need be in memory at any one moment.

4.5.2 Paging Organization

4.5.2.1 PAGE MECHANISM

The 386 Microprocessor uses two levels of tables to translate the linear address (from the segmentation unit) into a physical address. There are three components to the paging mechanism of the 386 Microprocessor: the page directory, the page tables, and the page itself (page frame). All memory-resident elements of the 386 Microprocessor paging mechanism are the same size, namely, 4K bytes. A uniform size for all of the elements simplifies memory allocation and reallocation schemes, since there is no problem with memory fragmentation. Figure 4-19 shows how the paging mechanism works.

4.5.2.2 PAGE DESCRIPTOR BASE REGISTER

CR2 is the Page Fault Linear Address register. It holds the 32-bit linear address which caused the last page fault detected.

CR3 is the Page Directory Physical Base Address Register. It contains the physical starting address of the Page Directory. The lower 12 bits of CR3 are always zero to ensure that the Page Directory is always page aligned. Loading it via a MOV CR3, reg instruction causes the Page Table Entry cache to be flushed, as will a task switch through a TSS which **changes** the value of CR0. (See 4.5.4 **Translation Lookaside Buffer**).

4.5.2.3 PAGE DIRECTORY

The Page Directory is 4K bytes long and allows up to 1024 Page Directory Entries. Each Page Directory Entry contains the address of the next level of tables, the Page Tables and information about the page table. The contents of a Page Directory Entry are shown in Figure 4-20. The upper 10 bits of the linear address (A22–A31) are used as an index to select the correct Page Directory Entry.

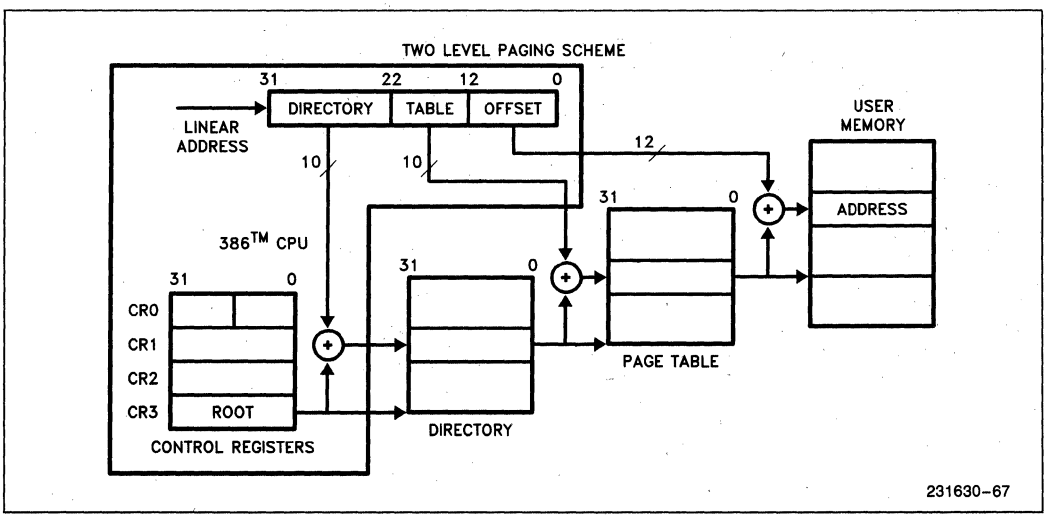


Figure 4-19. Paging Mechanism

31 12	11 10 9	8	7	6	5	4	3	2	1	0
PAGE TABLE ADDRESS 31..12	OS RESERVED	0	0	D	A	0	0	U/S	R/W	P

Figure 4-20. Page Directory Entry (Points to Page Table)

31 12	11 10	9	8	7	6	5	4	3	2	1	0
PAGE FRAME ADDRESS 31..12	OS RESERVED		0	0	D	A	0	0	U — S	R — W	P

Figure 4-21. Page Table Entry (Points to Page)

4.5.2.4 PAGE TABLES

Each Page Table is 4K bytes and holds up to 1024 Page Table Entries. Page Table Entries contain the starting address of the page frame and statistical information about the page (see Figure 4-21). Address bits A12–A21 are used as an index to select one of the 1024 Page Table Entries. The 20 upper-bit page frame address is concatenated with the lower 12 bits of the linear address to form the physical address. Page tables can be shared between tasks and swapped to disks.

4.5.2.5 PAGE DIRECTORY/TABLE ENTRIES

The lower 12 bits of the Page Table Entries and Page Directory Entries contain statistical information about pages and page tables respectively. The **P** (Present) bit 0 indicates if a Page Directory or Page Table entry can be used in address translation. If P = 1 the entry can be used for address translation if P = 0 the entry can not be used for translation, and all of the other bits are available for use by the software. For example the remaining 31 bits could be used to indicate where on the disk the page is stored.

The **A** (Accessed) bit 5, is set by the 386 Microprocessor for both types of entries before a read or write access occurs to an address covered by the entry. The **D** (Dirty) bit 6 is set to 1 before a write to an address covered by that page table entry occurs. The D bit is undefined for Page Directory Entries. When the P, A and D bits are updated by the 386 Microprocessor, the processor generates a Read-Modify-Write cycle which locks the bus and prevents conflicts with other processors or perpherials. Software which modifies these bits should use the LOCK prefix to ensure the integrity of the page tables in multi-master systems.

The 3 bits marked **OS Reserved** in Figure 4-20 and Figure 4-21 (bits 9–11) are software definable. OSs are free to use these bits for whatever purpose they wish. An example use of the **OS Reserved** bits would be to store information about page aging. By keeping track of how long a page has been in memory since being accessed, an operating system can implement a page replacement algorithm like Least Recently Used.

The (User/Supervisor) U/S bit 2 and the (Read/Write) R/W bit 1 are used to provide protection attributes for individual pages.

4.5.3 Page Level Protection (R/W, U/S Bits)

The 386 Microprocessor provides a set of protection attributes for paging systems. The paging mechanism distinguishes between two levels of protection: User which corresponds to level 3 of the segmentation based protection, and supervisor which encompasses all of the other protection levels (0, 1, 2). Programs executing at Level 0, 1 or 2 bypass the page protection, although segmentation based protection is still enforced by the hardware.

The U/S and R/W bits are used to provide User/Supervisor and Read/Write protection for individual pages or for all pages covered by a Page Table Directory Entry. The U/S and R/W bits in the first level Page Directory Table apply to all pages described by the page table pointed to by that directory entry. The U/S and R/W bits in the second level Page Table Entry apply only to the page described by that entry. The U/S and R/W bits for a given page are obtained by taking the most restrictive of the U/S and R/W from the Page Directory Table Entries and the Page Table Entries and using these bits to address the page.

Example: If the U/S and R/W bits for the Page Directory entry were 10 and the U/S and R/W bits for the Page Table Entry were 01, the access rights for the page would be 01, the numerically smaller of the two. Table 4-4 shows the affect of the U/S and R/W bits on accessing memory.

Table 4-4. Protection Provided by R/W and U/S

U/S	R/W	Permitted Level 3	Permitted Access Levels 0, 1, or 2
0	0	None	Read/Write
0	1	None	Read/Write
1	0	Read-Only	Read/Write
1	1	Read/Write	Read/Write

However a given segment can be easily made read-only for level 0, 1, or 2 via the use of segmented protection mechanisms. (Section 4.4 **Protection**).

4.5.4 Translation Lookaside Buffer

The 386 Microprocessor paging hardware is designed to support demand paged virtual memory systems. However, performance would degrade substantially if the processor was required to access two levels of tables for every memory reference. To solve this problem, the 386 Microprocessor keeps a cache of the most recently accessed pages, this cache is called the Translation Lookaside Buffer (TLB). The TLB is a four-way set associative 32-entry page table cache. It automatically keeps the most commonly used Page Table Entries in the processor. The 32-entry TLB coupled with a 4K page size, results in coverage of 128K bytes of memory addresses. For many common multi-tasking systems, the TLB will have a hit rate of about 98%. This means that the processor will only have to access the two-level page structure on 2% of all memory references. Figure 4-22 illustrates how the TLB complements the 386 Microprocessor's paging mechanism.

4.5.5 Paging Operation

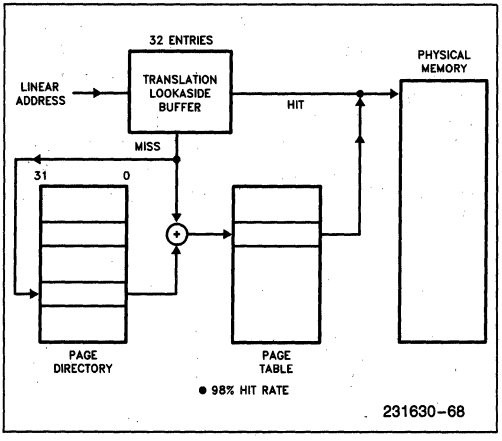


Figure 4-22. Translation Lookaside Buffer

The paging hardware operates in the following fashion. The paging unit hardware receives a 32-bit linear address from the segmentation unit. The upper 20 linear address bits are compared with all 32 entries in the TLB to determine if there is a match. If there is a match (i.e. a TLB hit), then the 32-bit physical address is calculated and will be placed on the address bus.

However, if the page table entry is not in the TLB, the 386 Microprocessor will read the appropriate Page Directory Entry. If P = 1 on the Page Directory Entry indicating that the page table is in memory, then the 386 Microprocessor will read the appropriate Page Table Entry and set the Access bit. If P = 1 on the Page Table Entry indicating that the page is in memory, the 386 Microprocessor will update the Access and Dirty bits as needed and fetch the operand. The upper 20 bits of the linear address, read from the page table, will be stored in the TLB for future accesses. However, if P = 0 for either the Page Directory Entry or the Page Table Entry, then the processor will generate a page fault, an Exception 14.

The processor will also generate an exception 14, page fault, if the memory reference violated the page protection attributes (i.e. U/S or R/W) (e.g. trying to write to a read-only page). CR2 will hold the linear address which caused the page fault. If a second page fault occurs, while the processor is attempting to enter the service routine for the first, then the processor will invoke the page fault (exception 14) handler a second time, rather than the double fault (exception 8) handler. Since Exception 14 is classified as a fault, CS: EIP will point to the instruction causing the page fault. The 16-bit error code pushed as part of the page fault handler will contain status bits which indicate the cause of the page fault.

The 16-bit error code is used by the operating system to determine how to handle the page fault Figure 4-23A shows the format of the page-fault error code and the interpretation of the bits.

NOTE:
Even though the bits in the error code (U/S, W/R, and P) have similar names as the bits in the Page Directory/Table Entries, the interpretation of the error code bits is different. Figure 4-23B indicates what type of access caused the page fault.

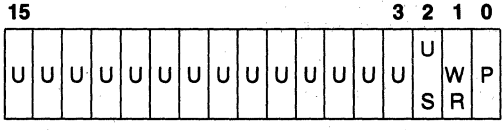


Figure 4-23A. Page Fault Error Code Format

U/S: The U/S bit indicates whether the access causing the fault occurred when the processor was executing in User Mode (U/S = 1) or in Supervisor mode (U/S = 0)

W/R: The W/R bit indicates whether the access causing the fault was a Read (W/R = 0) or a Write (W/R = 1)

P: The P bit indicates whether a page fault was caused by a not-present page (P = 0), or by a page level protection violation (P = 1)

U: UNDEFINED

U/S	W/R	Access Type
0	0	Supervisor* Read
0	1	Supervisor Write
1	0	User Read
1	1	User Write

*Descriptor table access will fault with U/S = 0, even if the program is executing at level 3.

Figure 4-23B. Type of Access Causing Page Fault

4.5.6 Operating System Responsibilities

The 386 Microprocessor takes care of the page address translation process, relieving the burden from an operating system in a demand-paged system. The operating system is responsible for setting up the initial page tables, and handling any page faults. The operating system also is required to invalidate (i.e. flush) the TLB when any changes are made to any of the page table entries. The operating system must reload CR3 to cause the TLB to be flushed.

Setting up the tables is simply a matter of loading CR3 with the address of the Page Directory, and allocating space for the Page Directory and the Page Tables. The primary responsibility of the operating system is to implement a swapping policy and handle all of the page faults.

A final concern of the operating system is to ensure that the TLB cache matches the information in the paging tables. In particular, any time the operating system sets the P present bit of page table entry to zero, the TLB must be flushed. Operating systems may want to take advantage of the fact that CR3 is stored as part of a TSS, to give every task or group of tasks its own set of page tables.

4.6 VIRTUAL 8086 ENVIRONMENT

4.6.1 Executing 8086 Programs

The 386 Microprocessor allows the execution of 8086 application programs in both Real Mode and in the Virtual 8086 Mode (Virtual Mode). Of the two methods, Virtual 8086 Mode offers the system designer the most flexibility. The Virtual 8086 Mode allows the execution of 8086 applications, while still allowing the system designer to take full advantage of the 386 Microprocessor protection mechanism. In particular, the 386 Microprocessor allows the simultaneous execution of 8086 operating systems and its applications, and a 386 Microprocessor operating system and both 80286 and 386 Microprocessor applications. Thus, in a multi-user 386 Microprocessor computer, one person could be running an MS-DOS spreadsheet, another person using MS-DOS, and a third person could be running multiple Unix utilities and applications. Each person in this scenario would believe that he had the computer completely to himself. Figure 4-24 illustrates this concept.

4.6.2 Virtual 8086 Mode Addressing Mechanism

One of the major differences between 386 Microprocessor Real and Protected modes is how the segment selectors are interpreted. When the processor is executing in Virtual 8086 Mode the segment registers are used in an identical fashion to Real Mode. The contents of the segment register is shifted left 4 bits and added to the offset to form the segment base linear address.

The 386 Microprocessor allows the operating system to specify which programs use the 8086 style address mechanism, and which programs use Protected Mode addressing, on a per task basis. Through the use of paging, the one megabyte address space of the Virtual Mode task can be mapped to anywhere in the 4 gigabyte linear address space of the 386 Microprocessor. Like Real Mode, Virtual Mode effective addresses (i.e., segment offsets) that exceed 64K byte will cause an exception 13. However, these restrictions should not prove to be important, because most tasks running in Virtual 8086 Mode will simply be existing 8086 application programs.

4.6.3 Paging In Virtual Mode

The paging hardware allows the concurrent running of multiple Virtual Mode tasks, and provides protection and operating system isolation. Although it is not strictly necessary to have the paging hardware enabled to run Virtual Mode tasks, it is needed in order to run multiple Virtual Mode tasks or to relocate the address space of a Virtual Mode task to physical address space greater than one megabyte.

The paging hardware allows the 20-bit linear address produced by a Virtual Mode program to be divided into up to 256 pages. Each one of the pages can be located anywhere within the maximum 4 gigabyte physical address space of the 386 Microprocessor. In addition, since CR3 (the Page Directory Base Register) is loaded by a task switch, each Virtual Mode task can use a different mapping scheme to map pages to different physical locations. Finally, the paging hardware allows the sharing of the 8086 operating system code between multiple 8086 ap-

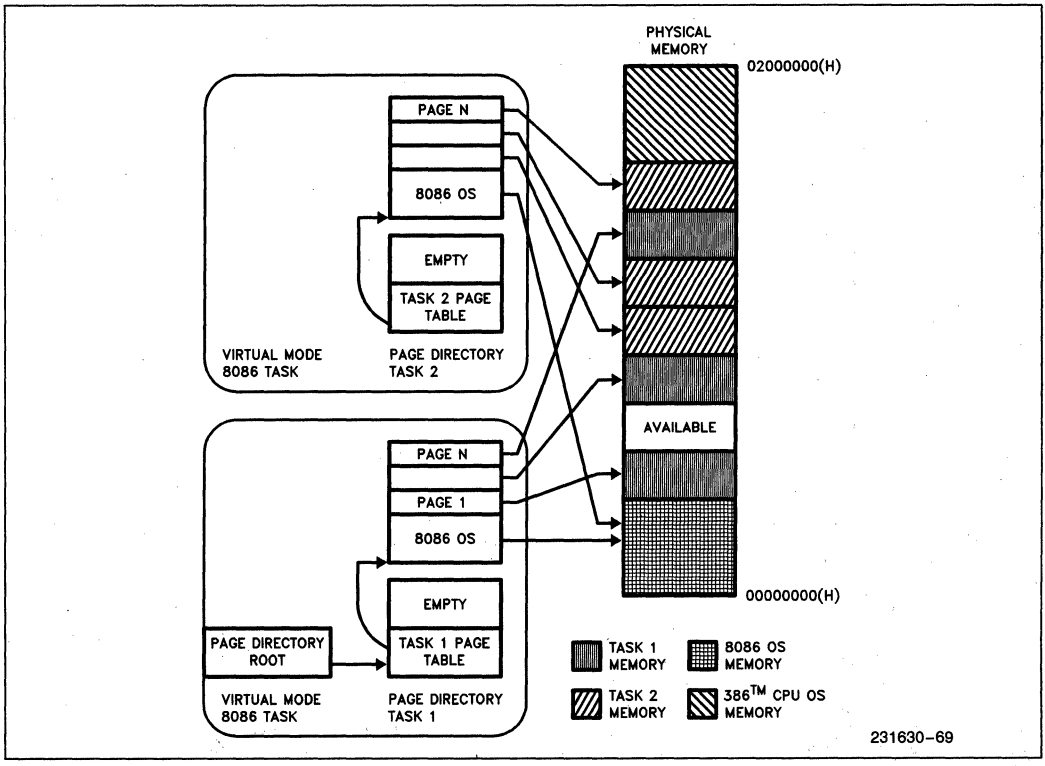


Figure 4-24. Virtual 8086 Environment Memory Management

plications. Figure 4-24 shows how the 386 Microprocessor paging hardware enables multiple 8086 programs to run under a virtual memory demand paged system.

4.6.4 Protection and I/O Permission Bitmap

All Virtual 8086 Mode programs execute at privilege level 3, the level of least privilege. As such, Virtual 8086 Mode programs are subject to all of the protection checks defined in Protected Mode. (This is different from Real Mode which implicitly is executing at privilege level 0, the level of greatest privilege.) Thus, an attempt to execute a privileged instruction when in Virtual 8086 Mode will cause an exception 13 fault.

The following are privileged instructions, which may be executed only at Privilege Level 0. Therefore, attempting to execute these instructions in Virtual 8086 Mode (or anytime CPL > 0) causes an exception 13 fault:

```
LIDT;    MOV DRn,reg;    MOV reg,DRn;
LGDT;    MOV TRn,reg;    MOV reg,TRn;
LMSW;    MOV CRn,reg;    MOV reg,CRn.
CLTS;
HLT;
```

Several instructions, particularly those applying to the multitasking model and protection model, are available only in Protected Mode. Therefore, attempting to execute the following instructions in Real Mode or in Virtual 8086 Mode generates an exception 6 fault:

```
LTR;     STR;
LLDT;    SLDT;
LAR;     VERR;
LSL;     VERW;
ARPL.
```

The instructions which are IOPL-sensitive in Protected Mode are:

```
IN;          STI;
OUT;         CLI
INS;
OUTS;
REP INS;
REP OUTS;
```

In Virtual 8086 Mode, a slightly different set of instructions are made IOPL-sensitive. The following instructions are IOPL-sensitive in Virtual 8086 Mode:

```
INT n;    STI;
PUSHF;    CLI;
POPF;     IRET
```

The PUSHF, POPF, and IRET instructions are IOPL-sensitive in Virtual 8086 Mode only. This provision allows the IF flag (interrupt enable flag) to be virtualized to the Virtual 8086 Mode program. The INT n software interrupt instruction is also IOPL-sensitive in Virtual 8086 Mode. Note, however, that the INT 3 (opcode 0CCH), INTO, and BOUND instructions are not IOPL-sensitive in Virtual 8086 mode (they aren't IOPL sensitive in Protected Mode either).

Note that the I/O instructions (IN, OUT, INS, OUTS, REP INS, and REP OUTS) are **not** IOPL-sensitive in Virtual 8086 mode. Rather, the I/O instructions become automatically sensitive to the **I/O Permission Bitmap** contained in the **386 Microprocessor Task State Segment**. The I/O Permission Bitmap, automatically used by the 386 Microprocessor in Virtual 8086 Mode, is illustrated by Figures 4.15a and 4-15b.

The I/O Permission Bitmap can be viewed as a 0–64 Kbit bit string, which begins in memory at offset Bit_Map_Offset in the current TSS. Bit_Map_Offset must be ≤ DFFFH so the entire bit map and the byte FFH which follows the bit map are all at offsets ≤ FFFFH from the TSS base. The 16-bit pointer Bit_Map_Offset (15:0) is found in the word beginning at offset 66H (102 decimal) from the TSS base, as shown in Figure 4-15a.

Each bit in the I/O Permission Bitmap corresponds to a single byte-wide I/O port, as illustrated in Figure 4-15a. If a bit is 0, I/O to the corresponding byte-wide port can occur without generating an exception. Otherwise the I/O instruction causes an exception 13 fault. Since every byte-wide I/O port must be protectable, all bits corresponding to a word-wide or dword-wide port must be 0 for the word-wide or dword-wide I/O to be permitted. If all the referenced bits are 0, the I/O will be allowed. If any referenced bits are 1, the attempted I/O will cause an exception 13 fault.

Due to the use of a pointer to the base of the I/O Permission Bitmap, the bitmap may be located anywhere within the TSS, or may be ignored completely by pointing the Bit_Map_Offset (15:0) beyond the limit of the TSS segment. In the same manner, only a small portion of the 64K I/O space need have an associated map bit, by adjusting the TSS limit to truncate the bitmap. This eliminates the commitment of 8K of memory when a complete bitmap is not required, while allowing the fully general case if desired.

EXAMPLE OF BITMAP FOR I/O PORTS 0–255: Setting the TSS limit to {bit_Map_Offset + 31 +1**} [** see note below] will allow a 32-byte bitmap for the I/O ports #0–255, plus a terminator byte of all 1's [** see note below]. This allows the I/O bitmap to control I/O Permission to I/O port 0–255 while causing an exception 13 fault on attempted I/O to any I/O port 80256 through 65,565.

****IMPORTANT IMPLEMENTATION NOTE:** Beyond the last byte of I/O mapping information in the I/O Permission Bitmap **must** be a byte containing all 1's. The byte of all 1's must be within the limit of the 386 Microprocessor TSS segment (see Figure 4-15a).

4.6.5 Interrupt Handling

In order to fully support the emulation of an 8086 machine, interrupts in Virtual 8086 Mode are handled in a unique fashion. When running in Virtual Mode all interrupts and exceptions involve a privilege change back to the host 386 Microprocessor operating system. The 386 Microprocessor operating system determines if the interrupt comes from a Protected Mode application or from a Virtual Mode program by examining the VM bit in the EFLAGS image stored on the stack.

When a Virtual Mode program is interrupted and execution passes to the interrupt routine at level 0, the VM bit is cleared. However, the VM bit is still set in the EFLAG image on the stack.

The 386 Microprocessor operating system in turn handles the exception or interrupt and then returns control to the 8086 program. The 386 Microprocessor operating system may choose to let the 8086 operating system handle the interrupt or it may emulate the function of the interrupt handler. For example, many 8086 operating system calls are accessed by PUSHing parameters on the stack, and then executing an INT n instruction. If the IOPL is set to 0 then all INT n instructions will be intercepted by the 386 Microprocessor operating system. The 386 Microprocessor operating system could emulate the 8086 operating system's call. Figure 4-25 shows how the 386 Microprocessor operating system could intercept an 8086 operating system's call to "Open a File".

A 386 Microprocessor operating system can provide a Virtual 8086 Environment which is totally transparent to the application software via intercepting and then emulating 8086 operating system's calls, and intercepting IN and OUT instructions.

4.6.6 Entering and Leaving Virtual 8086 Mode

Virtual 8086 mode is entered by executing an IRET instruction (at CPL=0), or Task Switch (at any CPL) to a 386 Microprocessor task whose 386 Microprocessor TSS has a FLAGS image containing a 1 in the VM bit position while the processor is executing in Protected Mode. That is, one way to enter Virtual 8086 mode is to switch to a task with a 386 Microprocessor TSS that has a 1 in the VM bit in the EFLAGS image. The other way is to execute a 32-bit IRET instruction at privilege level 0, where the stack has a 1 in the VM bit in the EFLAGS image. POPF does not affect the VM bit, even if the processor is in Protected Mode or level 0, and so cannot be used to enter Virtual 8086 Mode. PUSHF always pushes a 0 in the VM bit, even if the processor is in Virtual 8086 Mode, so that a program cannot tell if it is executing in REAL mode, or in Virtual 8086 mode.

The VM bit can be set by executing an IRET instruction only at privilege level 0, or by any instruction or Interrupt which causes a task switch in Protected Mode (with VM=1 in the new FLAGS image), and can be cleared only by an interrupt or exception in Virtual 8086 Mode. IRET and POPF instructions executed in REAL mode or Virtual 8086 mode will not change the value in the VM bit.

The transition out of virtual 8086 mode to 386 Microprocessor protected mode occurs only on receipt of an interrupt or exception (such as due to a sensitive instruction). In Virtual 8086 mode, all interrupts and exceptions vector through the protected mode IDT, and enter an interrupt handler in protected 386 Microprocessor mode. That is, as part of interrupt processing, the VM bit is cleared.

Because the matching IRET must occur from level 0, if an Interrupt or Trap Gate is used to field an interrupt or exception out of Virtual 8086 mode, the Gate must perform an inter-level interrupt only to level 0. Interrupt or Trap Gates through conforming segments, or through segments with DPL>0, will raise a GP fault with the CS selector as the error code.

4.6.6.1 TASK SWITCHES TO/FROM VIRTUAL 8086 MODE

Tasks which can execute in virtual 8086 mode must be described by a TSS with the new 386 Microprocessor format (TYPE 9 or 11 descriptor).

A task switch out of virtual 8086 mode will operate exactly the same as any other task switch out of a task with a 386 Microprocessor TSS. All of the programmer visible state, including the FLAGS register with the VM bit set to 1, is stored in the TSS. The segment registers in the TSS will contain 8086 segment base values rather than selectors.

A task switch into a task described by a 386 Microprocessor TSS will have an additional check to determine if the incoming task should be resumed in virtual 8086 mode. Tasks described by 80286 format TSSs cannot be resumed in virtual 8086 mode, so no check is required there (the FLAGS image in 80286 format TSS has only the low order 16 FLAGS bits). Before loading the segment register images from a 386 Microprocessor TSS, the FLAGS image is loaded, so that the segment registers are loaded from the TSS image as 8086 segment base values. The task is now ready to resume in virtual 8086 execution mode.

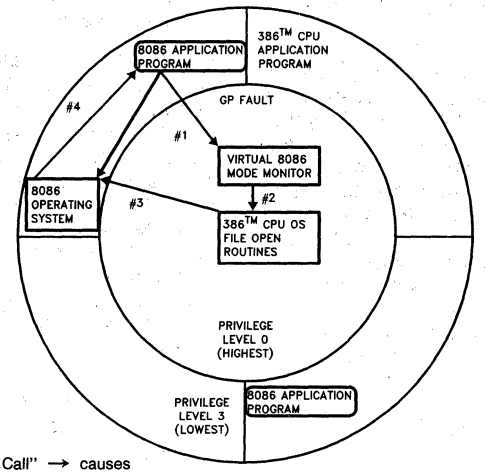
4.6.6.2 TRANSITIONS THROUGH TRAP AND INTERRUPT GATES, AND IRET

A task switch is one way to enter or exit virtual 8086 mode. The other method is to exit through a Trap or Interrupt gate, as part of handling an interrupt, and to enter as part of executing an IRET instruction. The transition out must use a 386 Microprocessor Trap Gate (Type 14), or 386 Microprocessor Interrupt Gate (Type 15), which must point to a non-conforming level 0 segment (DPL=0) in order to permit the trap handler to IRET back to the Virtual 8086 program. The Gate must point to a non-conforming level 0 segment to perform a level switch to level 0 so that the matching IRET can change the VM bit. 386 Microprocessor gates must be used, since 80286 gates save only the low 16 bits of the FLAGS register, so that the VM bit will not be saved on transitions through the 80286 gates. Also, the 16-bit IRET (presumably) used to terminate the 80286 interrupt handler will pop only the lower 16 bits from FLAGS, and will not affect the VM bit. The action taken for a 386 Microprocessor Trap or Interrupt gate if an interrupt occurs while the task is executing in virtual 8086 mode is given by the following sequence.

(1) Save the FLAGS register in a temp to push later. Turn off the VM and TF bits, and if the interrupt is serviced by an Interrupt Gate, turn off IF also.

(2) Interrupt and Trap gates must perform a level switch from 3 (where the VM86 program executes) to level 0 (so IRET can return). This process involves a stack switch to the stack given in the TSS for privilege level 0. Save the Virtual 8086 Mode SS and ESP registers to push in a later step. The segment register load of SS will be done as a Protected Mode segment load, since the VM bit was turned off above.

(3) Push the 8086 segment register values onto the new stack, in the order: GS, FS, DS, ES. These are pushed as 32-bit quantities, with undefined values in the upper 16 bits. Then load these 4 registers with null selectors (0).



8086 Application makes "Open File Call" → causes General Protection Fault (Arrow #1)
Virtual 8086 Monitor intercepts call. Calls 386™ CPU OS (Arrow #2)
386™ CPU OS opens file returns control to 8086 OS (Arrow #3)
8086 OS returns control to application. (Arrow #4)
Transparent to Application

231630-70

Figure 4-25. Virtual 8086 Environment Interrupt and Call Handling

(4) Push the old 8086 stack pointer onto the new stack by pushing the SS register (as 32-bits, high bits undefined), then pushing the 32-bit ESP register saved above.

(5) Push the 32-bit FLAGS register saved in step 1.

(6) Push the old 8086 instruction pointer onto the new stack by pushing the CS register (as 32-bits, high bits undefined), then pushing the 32-bit EIP register.

(7) Load up the new CS:EIP value from the interrupt gate, and begin execution of the interrupt routine in protected 386 Microprocessor mode.

The transition out of virtual 8086 mode performs a level change and stack switch, in addition to changing back to protected mode. In addition, all of the 8086 segment register images are stored on the stack (behind the SS:ESP image), and then loaded with null (0) selectors before entering the interrupt handler. This will permit the handler to safely save and restore the DS, ES, FS, and GS registers as 80286 selectors. This is needed so that interrupt handlers which don't care about the mode of the interrupted program can use the same prolog and epilog code for state saving (i.e. push all registers in prolog, pop all in epilog) regardless of whether or not a "native" mode or Virtual 8086 mode program was interrupted. Restoring null selectors to these registers before executing the IRET will not cause a trap in the interrupt handler. Interrupt routines which expect values in the segment registers, or return values in segment registers will have to obtain/return values from the 8086 register images pushed onto the new stack. They will need to know the mode of the interrupted program in order to know where to find/return segment registers, and also to know how to interpret segment register values.

The IRET instruction will perform the inverse of the above sequence. Only the extended 386 Microprocessors IRET instruction (operand size = 32) can be used, and must be executed at level 0 to change the VM bit to 1.

(1) If the NT bit in the FLAGs register is on, an inter-task return is performed. The current state is stored in the current TSS, and the link field in the current TSS is used to locate the TSS for the interrupted task which is to be resumed.

Otherwise, continue with the following sequence.

(2) Read the FLAGS image from SS:8[ESP] into the FLAGS register. This will set VM to the value active in the interrupted routine.

(3) Pop off the instruction pointer CS:EIP. EIP is popped first, then a 32-bit word is popped which contains the CS value in the lower 16 bits. If VM = 0, this CS load is done as a protected mode segment load. If VM = 1, this will be done as an 8086 segment load.

(4) Increment the ESP register by 4 to bypass the FLAGS image which was "popped" in step 1.

(5) If VM = 1, load segment registers ES, DS, FS, and GS from memory locations SS:[ESP + 8], SS:[ESP + 12], SS:[ESP + 16], and SS:[ESP + 20], respectively, where the new val-

ue of ESP stored in step 4 is used. Since VM = 1, these are done as 8086 segment register loads.

Else if VM = 0, check that the selectors in ES, DS, FS, and GS are valid in the interrupted routine. Null out invalid selectors to trap if an attempt is made to access through them.

(6) If (RPL(CS) > CPL), pop the stack pointer SS:ESP from the stack. The ESP register is popped first, followed by 32-bits containing SS in the lower 16 bits. If VM = 0, SS is loaded as a protected mode segment register load. If VM = 1, an 8086 segment register load is used.

(7) Resume execution of the interrupted routine. The VM bit in the FLAGS register (restored from the interrupt routine's stack image in step 1) determines whether the processor resumes the interrupted routine in Protected mode of Virtual 8086 mode.

5. FUNCTIONAL DATA

5.1 INTRODUCTION

The 386 Microprocessor features a straightforward functional interface to the external hardware. The 386 Microprocessor has separate, parallel buses for data and address. The data bus is 32-bits in width, and bidirectional. The address bus outputs 32-bit address values in the most directly usable form for the high-speed local bus: 4 individual byte enable signals, and the 30 upper-order bits as a binary value. The data and address buses are interpreted and controlled with their associated control signals.

A **dynamic data bus sizing** feature allows the processor to handle a mix of 32- and 16-bit external buses on a cycle-by-cycle basis (see **5.3.4 Data Bus Sizing**). If 16-bit bus size is selected, the 386 Microprocessor automatically makes any adjustment needed, even performing another 16-bit bus cycle to complete the transfer if that is necessary. 8-bit peripheral devices may be connected to 32-bit or 16-bit buses with no loss of performance. A **new address pipelining option** is provided and applies to 32-bit and 16-bit buses for substantially improved memory utilization, especially for the most heavily used memory resources.

The **address pipelining option**, when selected, typically allows a given memory interface to operate with one less wait state than would otherwise be required (see **5.4.2 Address Pipelining**). The pipelined bus is also well suited to interleaved memory designs. For 16 MHz interleaved memory designs with 100 ns access time DRAMs, zero wait states can be achieved when pipelined addressing is selected. When address pipelining is requested by the external hardware, the 386 Microprocessor will output the address and bus cycle definition of the next bus cycle (if it is internally available) even while waiting for the current cycle to be acknowledged.

Non-pipelined address timing, however, is ideal for external cache designs, since the cache memory will typically be fast enough to allow non-pipelined cycles. For maximum design flexibility, the address pipelining option is selectable on a cycle-by-cycle basis.

The processor's bus cycle is the basic mechanism for information transfer, either from system to processor, or from processor to system. 386 Microprocessor bus cycles perform data transfer in a minimum of only two clock periods. On a 32-bit data bus, the maximum 386 Microprocessor transfer bandwidth at 16 MHz is therefore 32 Mbytes/sec, at 20 MHz bandwidth is 40 MBytes/sec and at 25 MHz bandwidth is 50 Mbytes/sec. Any bus cycle will be extended for more than two clock periods, however, if external hardware withholds acknowledgement of the cycle. At the appropriate time, acknowledgement is signalled by asserting the 386 Microprocessor READY# input.

The 386 Microprocessor can relinquish control of its local buses to allow mastership by other devices, such as direct memory access channels. When relinquished, HLDA is the only output pin driven by the 386 Microprocessor, providing near-complete isolation of the processor from its system. The near-complete isolation characteristic is ideal when driving the system from test equipment, and in fault-tolerant applications.

Functional data covered in this chapter describes the processor's hardware interface. First, the set of signals available at the processor pins is described (see **5.2 Signal Description**). Following that are the signal waveforms occurring during bus cycles (see **5.3 Bus Transfer Mechanism**, **5.4 Bus Functional Description** and **5.5 Other Functional Descriptions**).

5.2 SIGNAL DESCRIPTION

5.2.1 Introduction

Ahead is a brief description of the 386 Microprocessor input and output signals arranged by functional groups. Note the # symbol at the end of a signal name indicates the active, or asserted, state occurs when the signal is at a low voltage. When no # is present after the signal name, the signal is asserted when at the high voltage level.

Example signal: M/IO# — High voltage indicates Memory selected

— Low voltage indicates I/O selected

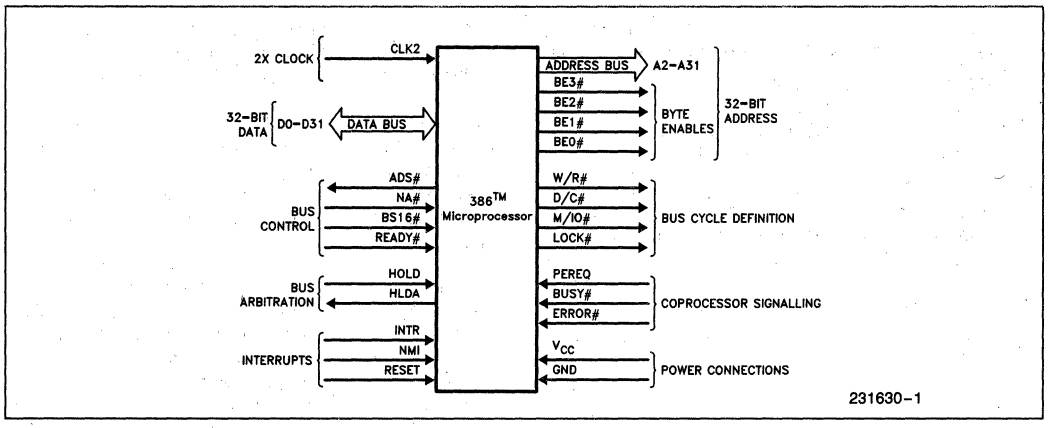


Figure 5-1. Functional Signal Groups

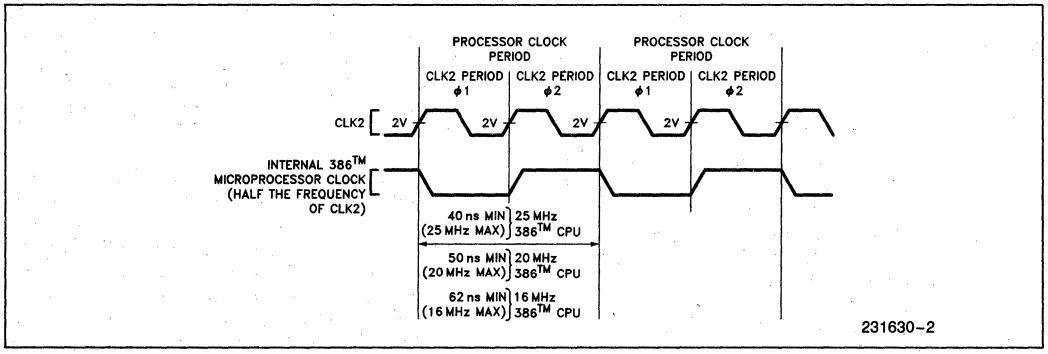


Figure 5-2. CLK2 Signal and Internal Processor Clock

The signal descriptions sometimes refer to AC timing parameters, such as "t_{25} Reset Setup Time" and "t_{26} Reset Hold Time." The values of these parameters can be found in Tables 7-4 and 7-5.

5.2.2 Clock (CLK2)

CLK2 provides the fundamental timing for the 386 Microprocessor. It is divided by two internally to generate the internal processor clock used for instruction execution. The internal clock is comprised of two phases, "phase one" and "phase two." Each CLK2 period is a phase of the internal clock. Figure 5-2 illustrates the relationship. If desired, the phase of the internal processor clock can be synchronized to a known phase by ensuring the RESET signal falling edge meets its applicable setup and hold times, t_{25} and t_{26}.

5.2.3 Data Bus (D0 through D31)

These three-state bidirectional signals provide the general purpose data path between the 386 Microprocessor and other devices. Data bus inputs and outputs indicate "1" when HIGH. The data bus can transfer data on 32- and 16-bit buses using a data bus sizing feature controlled by the BS16# input. See section **5.2.6 Bus Contol**. Data bus reads require that read data setup and hold times t_{21} and t_{22} be met for correct operation. During any write operation (and during halt cycles and shutdown cycles), the 386 Microprocessor always drives all 32 signals of the data bus even if the current bus size is 16-bits.

5.2.4 Address Bus (BE0# through BE3#, A2 through A31)

These three-state outputs provide physical memory addresses or I/O port addresses. The address bus is capable of addressing 4 gigabytes of physical memory space (00000000H through FFFFFFFFH), and 64 kilobytes of I/O address space (00000000H through 0000FFFFH) for programmed I/O. I/O

transfers automatically generated for 386 Microprocessor-to-coprocessor communication use I/O addresses 800000F8H through 800000FFH, so A31 HIGH in conjunction with M/IO# LOW allows simple generation of the coprocessor select signal.

The Byte Enable outputs, BE0#–BE3#, directly indicate which bytes of the 32-bit data bus are involved with the current transfer. This is most convenient for external hardware.

BE0# applies to D0–D7
BE1# applies to D8–D15
BE2# applies to D16–D23
BE3# applies to D24–D31

The number of Byte Enables asserted indicates the physical size of the operand being transferred (1, 2, 3, or 4 bytes). Refer to section **5.3.6 Operand Alignment**.

When a memory write cycle or I/O write cycle is in progress, and the operand being transferred occupies **only** the upper 16 bits of the data bus (D16–D31), duplicate data is simultaneously presented on the corresponding lower 16-bits of the data bus (D0–D15). This duplication is performed for optimum write performance on 16-bit buses. The pattern of write data duplication is a function of the Byte Enables asserted during the write cycle. Table 5-1 lists the write data present on D0–D31, as a function of the asserted Byte Enable outputs BE0#–BE3#.

5.2.5 Bus Cycle Definition Signals (W/R#, D/C#, M/IO#, LOCK#)

These three-state outputs define the type of bus cycle being performed. W/R# distinguishes between write and read cycles. D/C# distinguishes between data and control cycles. M/IO# distinguishes between memory and I/O cycles. LOCK# distinguishes between locked and unlocked bus cycles.

The primary bus cycle definition signals are W/R#, D/C# and M/IO#, since these are the signals driven valid as the ADS# (Address Status output) is driven asserted. The LOCK# is driven valid at the same time as the first locked bus cycle begins, which due to address pipelining, could be later than ADS# is driven asserted. See **5.4.3.4 Pipelined Address.** The LOCK# is negated when the READY# input terminates the last bus cycle which was locked.

Exact bus cycle definitions, as a function of W/R#, D/C#, and MI/IO#, are given in Table 5-2. Note one combination of W/R#, D/C# and M/IO# is never given when ADS# is asserted (however, that combination, which is listed as "does not occur," will occur during **idle** bus states when ADS# is **not** asserted). If M/IO#, D/C#, and W/R# are qualified by ADS# asserted, then a decoding scheme may use the non-occurring combination to its best advantage.

Table 5-1. Write Data Duplication as a Function of BE0#–BE3#

386™ CPU Byte Enables				386™ CPU Write Data				Automatic Duplication?
BE3#	BE2#	BE1#	BE0#	D24–D31	D16–D23	D8–D15	D0–D7	
High	High	High	Low	undef	undef	undef	A	No
High	High	Low	High	undef	undef	B	undef	No
High	Low	High	High	undef	C	undef	C	Yes
Low	High	High	High	D	undef	D	undef	Yes
High	High	Low	Low	undef	undef	B	A	No
High	Low	Low	High	undef	C	B	undef	No
Low	Low	High	High	D	C	D	C	Yes
High	Low	Low	Low	undef	C	B	A	No
Low	Low	Low	High	D	C	B	undef	No
Low	Low	Low	Low	D	C	B	A	No

Key:
D = logical write data d24–d31
C = logical write data d16–d23
B = logical write data d8–d15
A = logical write data d0–d7

Table 5-2. Bus Cycle Definition

M/IO#	D/C#	W/R#	Bus Cycle Type	Locked?
Low	Low	Low	INTERRUPT ACKNOWLEDGE	Yes
Low	Low	High	does not occur	—
Low	High	Low	I/O DATA READ	No
Low	High	High	I/O DATA WRITE	No
High	Low	Low	MEMORY CODE READ	No
High	Low	High	HALT: Address = 2 (BE0# High, BE1# High, BE2# Low, BE3# High, A2–A31 Low) / SHUTDOWN: Address = 0 (BE0# Low, BE1# High, BE2# High, BE3# High, A2–A31 Low)	No
High	High	Low	MEMORY DATA READ	Some Cycles
High	High	High	MEMORY DATA WRITE	Some Cycles

5.2.6 Bus Control Signals

5.2.6.1 INTRODUCTION

The following signals allow the processor to indicate when a bus cycle has begun, and allow other system hardware to control address pipelining, data bus width and bus cycle termination.

5.2.6.2 ADDRESS STATUS (ADS#)

This three-state output indicates that a valid bus cycle definition, and address (W/R#, D/C#, M/IO#, BE0#–BE3#, and A2–A31) is being driven at the 386 Microprocessor pins. It is asserted during T1 and T2P bus states (see **5.4.3.2 Non-pipelined Address** and **5.4.3.4 Pipelined Address** for additional information on bus states).

5.2.6.3 TRANSFER ACKNOWLEDGE (READY#)

This input indicates the current bus cycle is complete, and the active bytes indicated by BE0#–BE3# and BS16# are accepted or provided. When READY# is sampled asserted during a read cycle or interrupt acknowledge cycle, the 386 Microprocessor latches the input data and terminates the cycle. When READY# is sampled asserted during a write cycle, the processor terminates the bus cycle.

READY# is ignored on the first bus state of all bus cycles, and sampled each bus state thereafter until asserted. READY# must eventually be asserted to acknowledge every bus cycle, including Halt Indication and Shutdown Indication bus cycles. When being sampled, READY must always meet setup and hold times t_{19} and t_{20} for correct operation. See all sections of **5.4 Bus Functional Description**.

5.2.6.4 NEXT ADDRESS REQUEST (NA#)

This is used to request address pipelining. This input indicates the system is prepared to accept new values of BE0#–BE3#, A2–A31, W/R#, D/C# and M/IO# from the 386 Microprocessor even if the end of the current cycle is not being acknowledged on READY#. If this input is asserted when sampled, the next address is driven onto the bus, provided the next bus request is already pending internally. See **5.4.2 Address Pipelining** and **5.4.3 Read and Write Cycles**.

5.2.6.5 BUS SIZE 16 (BS16#)

The BS16# feature allows the 386 Microprocessor to directly connect to 32-bit and 16-bit data buses. Asserting this input constrains the current bus cycle to use only the lower-order half (D0–D15) of the data bus, corresponding to BE0# and BE1#. Asserting BS16# has no additional effect if only BE0# and/or BE1# are asserted in the current cycle. However, during bus cycles asserting BE2# or BE3#, asserting BS16# will automatically cause the 386 Microprocessor to make adjustments for correct transfer of the upper bytes(s) using only physical data signals D0–D15.

If the operand spans both halves of the data bus and BS16# is asserted, the 386 Microprocessor will automatically perform another 16-bit bus cycle. BS16# must always meet setup and hold times t_{17} and t_{18} for correct operation.

386 Microprocessor I/O cycles are automatically generated for coprocessor communication. Since the 386 Microprocessor must transfer 32-bit quantities between itself and the 80387, BS16# *must not* be asserted during 80387 communication cycles.

5.2.7 Bus Arbitration Signals

5.2.7.1 INTRODUCTION

This section describes the mechanism by which the processor relinquishes control of its local buses when requested by another bus master device. See **5.5.1 Entering and Exiting Hold Acknowledge** for additional information.

5.2.7.2 BUS HOLD REQUEST (HOLD)

This input indicates some device other than the 386 Microprocessor requires bus mastership.

HOLD must remain asserted as long as any other device is a local bus master. HOLD is not recognized while RESET is asserted. If RESET is asserted while HOLD is asserted, RESET has priority and places the bus into an idle state, rather than the hold acknowledge (high impedance) state.

HOLD is level-sensitive and is a synchronous input. HOLD signals must always meet setup and hold times t_{23} and t_{24} for correct operation.

5.2.7.3 BUS HOLD ACKNOWLEDGE (HLDA)

Assertion of this output indicates the 386 Microprocessor has relinquished control of its local bus in response to HOLD asserted, and is in the bus Hold Acknowledge state.

The Hold Acknowledge state offers near-complete signal isolation. In the Hold Acknowledge state, HLDA is the only signal being driven by the 386 Microprocessor. The other output signals or bidirectional signals (D0–D31, BE0#–BE3#, A2–A31, W/R#, D/C#, M/IO#, LOCK# and ADS#) are in a high-impedance state so the requesting bus master may control them. Pullup resistors may be desired on several signals to avoid spurious activity when no bus master is driving them. See **7.2.3 Resistor Recommendations**. Also, one rising edge occuring on the NMI input during Hold Acknowledge is remembered, for processing after the HOLD input is negated.

In addition to the normal usage of Hold Acknowledge with DMA controllers or master peripherals, the near-complete isolation has particular attractiveness during system test when test equipment drives the system, and in hardware-fault-tolerant applications.

5.2.8 Coprocessor Interface Signals

5.2.8.1 INTRODUCTION

In the following sections are descriptions of signals dedicated to the numeric coprocessor interface. In addition to the data bus, address bus, and bus cycle definition signals, these following signals control communication between the 386 Microprocessor and its 80387 processor extension.

5.2.8.2 COPROCESSOR REQUEST (PEREQ)

When asserted, this input signal indicates a coprocessor request for a data operand to be transferred to/from memory by the 386 Microprocessor. In response, the 386 Microprocessor transfers information between the coprocessor and memory. Because the 386 Microprocessor has internally stored the coprocessor opcode being executed, it performs the requested data transfer with the correct direction and memory address.

PEREQ is level-sensitive and is allowed to be asynchronous to the CLK2 signal.

5.2.8.3 COPROCESSOR BUSY (BUSY#)

When asserted, this input indicates the coprocessor is still executing an instruction, and is not yet able to accept another. When the 386 Microprocessor encounters any coprocessor instruction which operates on the numeric stack (e.g. load, pop, or arithmetic operation), or the WAIT instruction, this input is first automatically sampled until it is seen to be negated. This sampling of the BUSY# input prevents overrunning the execution of a previous coprocessor instruction.

The FNINIT and FNCLEX coprocessor instructions are allowed to execute even if BUSY# is asserted, since these instructions are used for coprocessor initialization and exception-clearing.

BUSY# is level-sensitive and is allowed to be asynchronous to the CLK2 signal.

BUSY# serves an additional function. If BUSY# is sampled LOW at the falling edge of RESET, the 386 Microprocessor performs an internal self-test (see **5.5.3 Bus Activity During and Following Reset**). If BUSY# is sampled HIGH, no self-test is performed.

5.2.8.4 COPROCESSOR ERROR (ERROR#)

This input signal indicates that the previous coprocessor instruction generated a coprocessor error of a type not masked by the coprocessor's control register. This input is automatically sampled by the 386 Microprocessor when a coprocessor instruction is encountered, and if asserted, the 386 Microprocessor generates exception 16 to access the error-handling software.

Several coprocessor instructions, generally those which clear the numeric error flags in the coprocessor or save coprocessor state, do execute without the 386 Microprocessor generating exception 16 even if ERROR# is asserted. These instructions are FNINIT, FNCLEX, FSTSW, FSTSWAX, FSTCW, FSTENV, FSAVE, FESTENV and FESAVE.

ERROR# is level-sensitive and is allowed to be asynchronous to the CLK2 signal.

ERROR# serves an additional function. If ERROR# is LOW no later than 20 CLK2 periods after the falling edge of RESET and remains LOW at least until the 386 Microprocessor begins its first bus cycle, an 80387 is assumed to be present (ET bit in CR0 automatically gets set to 1). Otherwise, an 80287 (or no coprocessor) is assumed to be present (ET bit in CR0 automatically is reset to 0). See **5.5.3 Bus Activity During and After Reset.** Only the ET bit is set by this ERROR# pin test. Software must set the EM and MP bits in CR0 as needed. Therefore, distinguishing 80287 presence from no coprocessor requires a software test and appropriately resetting or setting the EM bit of CR0 (set EM = 1 when no coprocessor is present). If ERROR# is sampled LOW after reset (indicating 80387) but software later sets EM = 1, the 386 Microprocessor will behave as if no coprocessor is present.

5.2.9 Interrupt Signals

5.2.9.1 INTRODUCTION

The following descriptions cover inputs that can interrupt or suspend execution of the processor's current instruction stream.

5.2.9.2 MASKABLE INTERRUPT REQUEST (INTR)

When asserted, this input indicates a request for interrupt service, which can be masked by the 386 Microprocessor Flag Register IF bit. When the 386 Microprocessor responds to the INTR input, it performs two interrupt acknowledge bus cycles, and at the end of the second, latches an 8-bit interrupt vector on D0–D7 to identify the source of the interrupt.

INTR is level-sensitive and is allowed to be asynchronous to the CLK2 signal. To assure recognition of an INTR request, INTR should remain asserted until the first interrupt acknowledge bus cycle begins.

5.2.9.3 NON-MASKABLE INTERRUPT REQUEST (NMI)

This input indicates a request for interrupt service, which cannot be masked by software. The non-maskable interrupt request is always processed according to the pointer or gate in slot 2 of the interrupt table. Because of the fixed NMI slot assignment, no interrupt acknowledge cycles are perfomed when processing NMI.

NMI is rising edge-sensitive and is allowed to be asynchronous to the CLK2 signal. To assure recognition of NMI, it must be negated for at least eight CLK2 periods, and then be asserted for at least eight CLK2 periods.

Once NMI processing has begun, no additional NMI's are processed until after the next IRET instruction, which is typically the end of the NMI service routine. If NMI is re-asserted prior to that time, however, one rising edge on NMI will be remembered for processing after executing the next IRET instruction.

5.2.9.4 RESET (RESET)

This input signal suspends any operation in progress and places the 386 Microprocessor in a known reset state. The 386 Microprocessor is reset by asserting RESET for 15 or more CLK2 periods (80 or more CLK2 periods before requesting self test). When RESET is asserted, all other input pins are ignored, and all other bus pins are driven to an idle bus state as shown in Table 5-3. If RESET and HOLD are both asserted at a point in time, RESET takes priority even if the 386 Microprocessor was in a Hold Acknowledge state prior to RESET asserted.

RESET is level-sensitive and must be synchronous to the CLK2 signal. If desired, the phase of the internal processor clock, and the entire 386 Microprocessor state can be completely synchronized to external circuitry by ensuring the RESET signal falling edge meets its applicable setup and hold times, t_{25} and t_{26}.

Table 5-3. Pin State (Bus Idle) During Reset

Pin Name	Signal Level During Reset
ADS#	High
D0–D31	High Impedance
BE0#–BE3#	Low
A2–A31	High
W/R#	Low
D/C#	High
M/IO#	Low
LOCK#	High
HLDA	Low

5.2.10 Signal Summary

Table 5-4 summarizes the characteristics of all 386 Microprocessor signals.

Table 5-4. 386™ Microprocessor Signal Summary

Signal Name	Signal Function	Active State	Input/ Output	Input Synch or Asynch to CLK2	Output High Impedance During HLDA?
CLK2	Clock	—	I	—	—
D0–D31	Data Bus	High	I/O	S	Yes
BE0#–BE3#	Byte Enables	Low	O	—	Yes
A2–A31	Address Bus	High	O	—	Yes
W/R#	Write-Read Indication	High	O	—	Yes
D/C#	Data-Control Indication	High	O	—	Yes
M/IO#	Memory-I/O Indication	High	O	—	Yes
LOCK#	Bus Lock Indication	Low	O	—	Yes
ADS#	Address Status	Low	O	—	Yes
NA#	Next Address Request	Low	I	S	—
BS16#	Bus Size 16	Low	I	S	—
READY#	Transfer Acknowledge	Low	I	S	—
HOLD	Bus Hold Request	High	I	S	—
HLDA	Bus Hold Acknowledge	High	O	—	No
PEREQ	Coprocessor Request	High	I	A	—
BUSY#	Coprocessor Busy	Low	I	A	—
ERROR#	Coprocessor Error	Low	I	A	—
INTR	Maskable Interrupt Request	High	I	A	—
NMI	Non-Maskable Intrpt Request	High	I	A	—
RESET	Reset	High	I	S	—

5.3 BUS TRANSFER MECHANISM

5.3.1 Introduction

All data transfers occur as a result of one or more bus cycles. Logical data operands of byte, word and double-word lengths may be transferred without restrictions on physical address alignment. Any byte boundary may be used, although two or even three physical bus cycles are performed as required for unaligned operand transfers. See **5.3.4 Dynamic Data Bus Sizing** and **5.3.6 Operand Alignment**.

The 386 Microprocessor address signals are designed to simplify external system hardware. Higher-order address bits are provided by A2–A31. Lower-order address in the form of BE0#–BE3# directly provides linear selects for the four bytes of the 32-bit data bus. Physical operand size information is thereby implicitly provided each bus cycle in the most usable form.

Byte Enable outputs BE0#–BE3# are asserted when their associated data bus bytes are involved with the present bus cycle, as listed in Table 5-5. During a bus cycle, any possible pattern of contiguous, asserted Byte Enable outputs can occur, but never patterns having a negated Byte Enable separating two or three asserted Enables.

Address bits A0 and A1 of the physical operand's base address can be created when necessary (for instance, for MULTIBUS® I or MULTIBUS® II interface), as a function of the lowest-order asserted Byte Enable. This is shown by Table 5-6. Logic to generate A0 and A1 is given by Figure 5-3.

Table 5-5. Byte Enables and Associated Data and Operand Bytes

Byte Enable Signal	Associated Data Bus Signals
BE0#	D0–D7 (byte 0—least significant)
BE1#	D8–D15 (byte 1)
BE2#	D16–D23 (byte 2)
BE3#	D24–D31 (byte 3—most significant)

Table 5-6. Generating A0–A31 from BE0#–BE3# and A2–A31

386™ CPU Address Signals							
A31 ..... A2				BE3#	BE2#	BE1#	BE0#
Physical Base Address							
A31	A2	A1	A0				
A31	A2	0	0	X	X	X	Low
A31	A2	0	1	X	X	Low	High
A31	A2	1	0	X	Low	High	High
A31	A2	1	1	Low	High	High	High

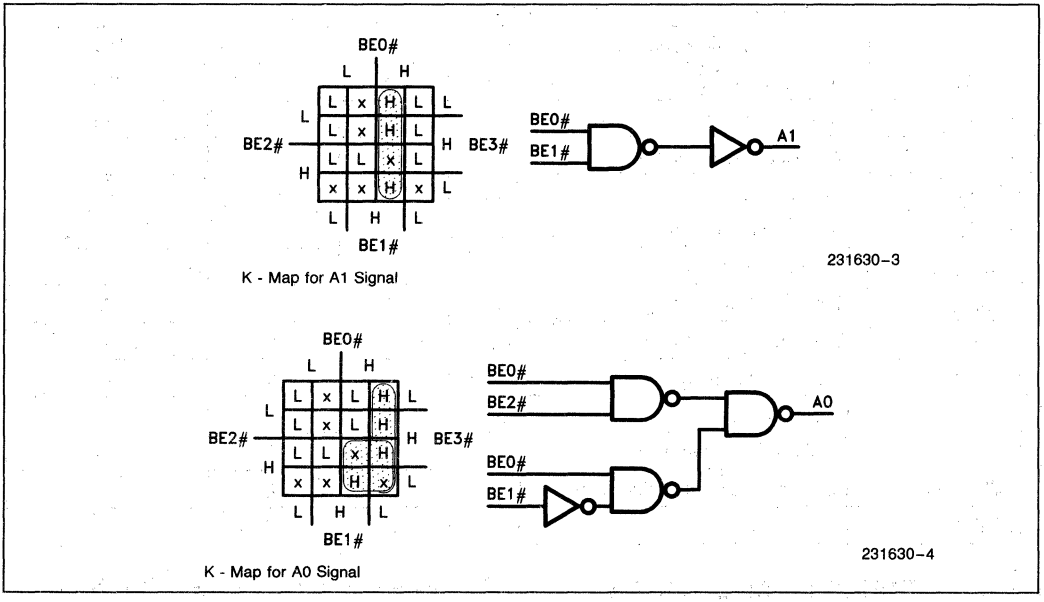


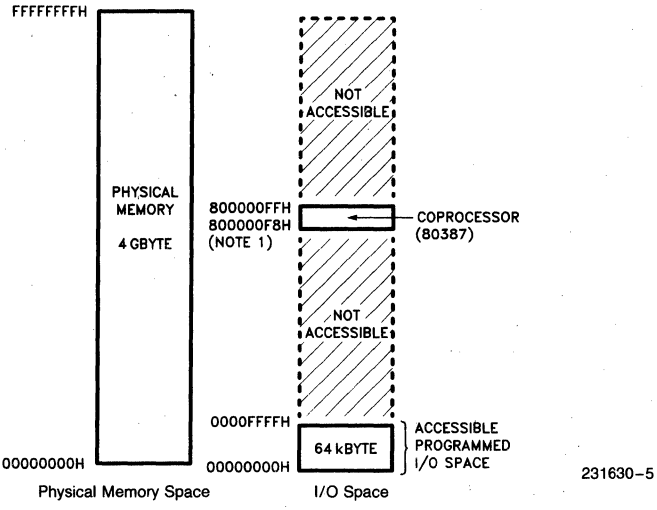
Figure 5-3. Logic to Generate A0, A1 from BE0#–BE3#

Each bus cycle is composed of at least two bus states. Each bus state requires one processor clock period. Additional bus states added to a single bus cycle are called wait states. See **5.4 Bus Functional Description**.

Since a bus cycle requires a minimum of two bus states (equal to two processor clock periods), data can be transferred between external devices and the 386 Microprocessor at a maximum rate of one 4-byte Dword every two processor clock periods, for a maximum bus bandwidth of 50 megabytes/second (386 Microprocessor operating at 25 MHz processor clock rate).

5.3.2 Memory and I/O Spaces

Bus cycles may access physical memory space or I/O space. Peripheral devices in the system may either be memory-mapped, or I/O-mapped, or both. As shown in Figure 5-4, physical memory addresses range from 00000000H to FFFFFFFFH (4 gigabytes) and I/O addresses from 00000000H to 0000FFFFH (64 kilobytes) for programmed I/O. Note the I/O addresses used by the automatic I/O cycles for coprocessor communication are 800000F8H to 800000FFH, beyond the address range of programmed I/O, to allow easy generation of a coprocessor chip select signal using the A31 and M/IO# signals.



NOTE:
Since A31 is HIGH during automatic communication with coprocessor, A31 HIGH and M/IO# LOW can be used to easily generate a coprocessor select signal.

Figure 5-4. Physical Memory and I/O Spaces

5.3.3 Memory and I/O Organization

The 386 Microprocessor datapath to memory and I/O spaces can be 32 bits wide or 16 bits wide. When 32-bits wide, memory and I/O spaces are organized naturally as arrays of physical 32-bit Dwords. Each memory or I/O Dword has four individually addressable bytes at consecutive byte addresses. The lowest-addressed byte is associated with data signals D0–D7; the highest-addressed byte with D24–D31.

The 386 Microprocessor includes a bus control input, BS16#, that also allows direct connection to 16-bit memory or I/O spaces organized as a sequence of 16-bit words. Cycles to 32-bit and 16-bit memory or I/O devices may occur in any sequence, since the BS16# control is sampled during each bus cycle. See **5.3.4 Dynamic Data Bus Sizing**. The Byte Enable signals, BE0#–BE3#, allow byte granularity when addressing any memory or I/O structure, whether 32 or 16 bits wide.

5.3.4 Dynamic Data Bus Sizing

Dynamic data bus sizing is a feature allowing direct processor connection to 32-bit or 16-bit data buses for memory or I/O. A single processor may connect to both size buses. Transfers to or from 32- or 16-bit ports are supported by dynamically determining the bus width during each bus cycle. During each bus cycle an address decoding circuit or the slave device itself may assert BS16# for 16-bit ports, or negate BS16# for 32-bit ports.

With BS16# asserted, the processor automatically converts operand transfers larger than 16 bits, or misaligned 16-bit transfers, into two or three transfers as required. All operand transfers physically occur on D0–D15 when BS16# is asserted. Therefore, 16-bit memories or I/O devices only connect on data signals D0–D15. No extra transceivers are required.

Asserting BS16# only affects the processor when BE2# and/or BE3# are asserted during the current cycle. If only D0–D15 are involved with the transfer, asserting BS16# has no affect since the transfer can proceed normally over a 16-bit bus whether BS16# is asserted or not. In other words, asserting BS16# has no effect when only the lower half of the bus is involved with the current cycle.

There are two types of situations where the processor is affected by asserting BS16#, depending on which Byte Enables are asserted during the current bus cycle:

Upper Half Only:
Only BE2# and/or BE3# asserted.

Upper and Lower Half:
At least BE1#, BE2# asserted (and perhaps also BE0# and/or BE3#).

Effect of asserting BS16# during "upper half only" read cycles:

Asserting BS16# during "upper half only" reads causes the 386 Microprocessor to read data on the lower 16 bits of the data bus and ignore data on the upper 16 bits of the data bus. Data that would have been read from D16–D31 (as indicated by BE2# and BE3#) will instead be read from D0–D15 respectively.

Effect of asserting BS16# during "upper half only" write cycles:

Asserting BS16# during "upper half only" writes does not affect the 386 Microprocessor. When only BE2# and/or BE3# are asserted during a write cycle the 386 Microprocessor always duplicates data signals D16–D31 onto D0–D15 (see Table 5-1). Therefore, no further 386 Microprocessor action is required to perform these writes on 32-bit or 16-bit buses.

Effect of asserting BS16# during "upper and lower half" read cycles:

Asserting BS16# during "upper and lower half" reads causes the processor to perform two 16-bit read cycles for complete physical operand transfer. Bytes 0 and 1 (as indicated by BE0# and BE1#) are read on the first cycle using D0–D15. Bytes 2 and 3 (as indicated by BE2# and BE3#) are read during the second cycle, again using D0–D15. D16–D31 are ignored during both 16-bit cycles. BE0# and BE1# are always negated during the second 16-bit cycle (See **Figure 5-14, cycles 2 and 2a**).

Effect of asserting BS16# during "upper and lower half" write cycles:

Asserting BS16# during "upper and lower half" writes causes the 386 Microprocessor to perform two 16-bit write cycles for complete physical operand transfer. All bytes are available the first write cycle allowing external hardware to receive Bytes 0 and 1 (as indicated by BE0# and BE1#) using D0–D15. On the second cycle the 386 Microprocessor duplicates Bytes 2 and 3 on D0–D15 and Bytes 2 and 3 (as indicated by BE2# and BE3#) are written using D0–D15. BE0# and BE1# are always negated during the second 16-bit cycle. BS16# must be asserted during the second 16-bit cycle. See **Figure 5-14, cycles 1 and 1a**.

5.3.5 Interfacing with 32- and 16-Bit Memories

In 32-bit-wide physical memories such as Figure 5-5, each physical Dword begins at a byte address that is a multiple of 4. A2–A31 are directly used as a Dword select and BE0#–BE3# as byte selects. BS16# is negated for all bus cycles involving the 32-bit array.

When 16-bit-wide physical arrays are included in the system, as in Figure 5-6, each 16-bit physical word begins at a address that is a multiple of 2. Note the address is decoded, to assert BS16# only during bus cycles involving the 16-bit array. (If desiring to

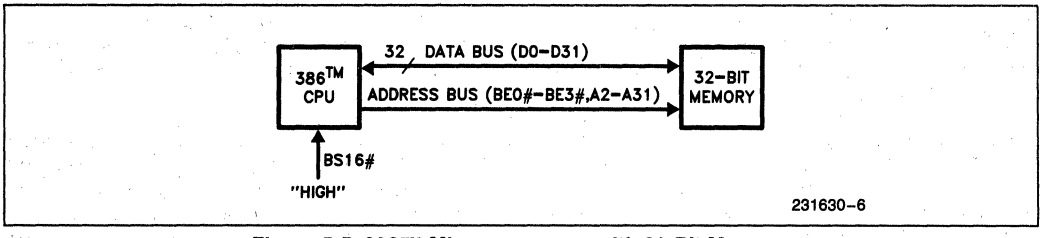


Figure 5-5. 386™ Microprocessor with 32-Bit Memory

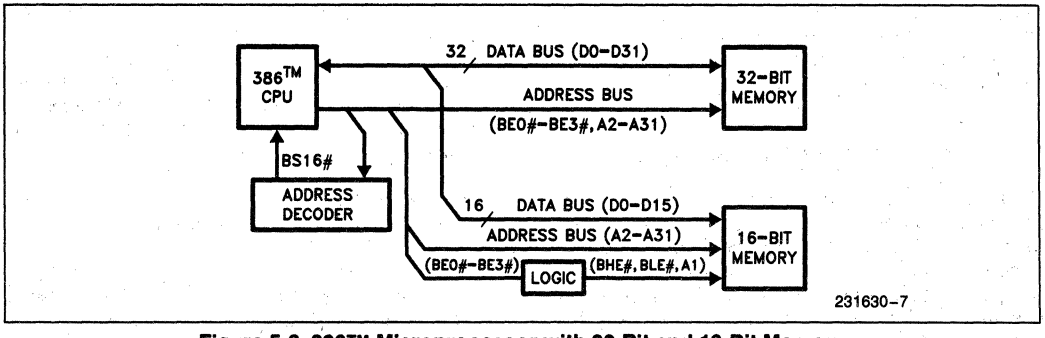


Figure 5-6. 386™ Microprocessor with 32-Bit and 16-Bit Memory

use pipelined address with 16-bit memories then BE0#–BE3# and W/R# are also decoded to determine when BS16# should be asserted. See **5.4.3.6 Pipelined Address with Dynamic Data Bus Sizing.**)

A2–A31 are directly usable for addressing 32-bit and 16-bit devices. To address 16-bit devices, A1 and two byte enable signals are also needed.

To generate an A1 signal and two Byte Enable signals for 16-bit access, BE0#–BE3# should be decoded as in Table 5-7. Note certain combinations of BE0#–BE3# are never generated by the 386 Microprocessor, leading to "don't care" conditions in the decoder. Any BE0#–BE3# decoder, such as Figure 5-7, may use the non-occurring BE0#–BE3# combinations to its best advantage.

5.3.6 Operand Alignment

With the flexibility of memory addressing on the 386 Microprocessor, it is possible to transfer a logical operand that spans more than one physical Dword or word of memory or I/O. Examples are 32-bit Dword operands beginning at addresses not evenly divisible by 4, or a 16-bit word operand split between two physical Dwords of the memory array.

Operand alignment and data bus size dictate when multiple bus cycles are required. Table 5-8 describes the transfer cycles generated for all combinations of logical operand lengths, alignment, and data bus sizing. When multiple bus cycles are required to transfer a multi-byte logical operand, the highest-order bytes are transferred first (but if BS16# asserted requires two 16-bit cycles be performed, that part of the transfer is low-order first).

5.4 BUS FUNCTIONAL DESCRIPTION

5.4.1 Introduction

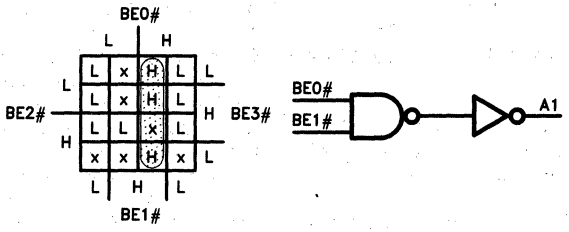
The 386 Microprocessor has separate, parallel buses for data and address. The data bus is 32-bits in width, and bidirectional. The address bus provides a 32-bit value using 30 signals for the 30 upper-order address bits and 4 Byte Enable signals to directly indicate the active bytes. These buses are interpreted and controlled via several associated definition or control signals.

Table 5-7. Generating A1, BHE# and BLE# for Addressing 16-Bit Devices

386™ CPU Signals				16-Bit Bus Signals			Comments
BE3#	BE2#	BE1#	BE0#	A1	BHE#	BLE# (A0)	
H*	H*	H*	H*	x	x	x	x—no active bytes
H	H	H	L	L	H	L	
H	H	L	H	L	L	H	
H	H	L	L	L	L	L	
H	L	H	H	H	H	L	
H*	L*	H*	L*	x	x	x	x—not contiguous bytes
H	L	L	H	L	L	H	
H	L	L	L	L	L	L	
L	H	H	H	H	L	H	
L*	H*	H*	L*	x	x	x	x—not contiguous bytes
L*	H*	L*	H*	x	x	x	x—not contiguous bytes
L*	H*	L*	L*	x	x	x	x—not contiguous bytes
L	L	H	H	H	L	L	
L*	L*	H*	L*	x	x	x	x—not continguous bytes
L	L	L	H	L	L	H	
L	L	L	L	L	L	L	

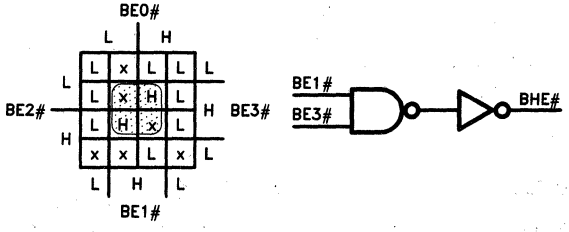
BLE# asserted when D0–D7 of 16-bit bus is active.
BHE# asserted when D8–D15 of 16-bit bus is active.
A1 low for all even words; A1 high for all odd words.

Key:
x = don't care
H = high voltage level
L = low voltage level
* = a non-occurring pattern of Byte Enables; either none are asserted, or the pattern has Byte Enables asserted for non-contiguous bytes



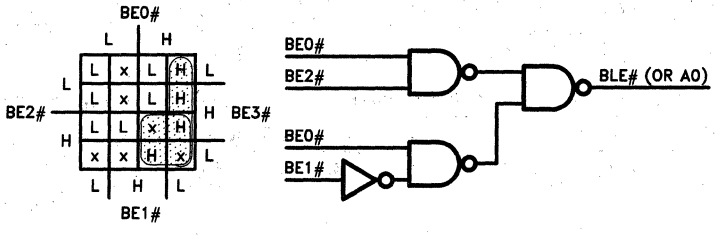
231630-8

K-map for A1 signal (same as Figure 5-3)



231630-9

K-map for 16-bit BHE# signal



231630-10

K-map for 16-bit BLE# signal (same as A0 signal in Figure 5-3)

Figure 5-7. Logic to Generate A1, BHE# and BLE# for 16-Bit Buses

Table 5-8. Transfer Bus Cycles for Bytes, Words and Dwords

	Byte-Length of Logical Operand								
	1	2				4			
Physical Byte Address in Memory (low-order bits)	xx	00	01	10	11	00	01	10	11
Transfer Cycles over 32-Bit Data Bus	b	w	w	w	hb,* lb	d	hb l3	hw, lw	h3, lb
Transfer Cycles over 16-Bit Data Bus	b	w	lb, hb	w	hb, lb	lw, hw	hb, lb, mw	hw, lw	mw, hb, lb

Key: b = byte transfer; w = word transfer; l = low-order portion; m = mid-order portion; x = don't care; 3 = 3-byte transfer; d = Dword transfer; h = high-order portion; ▒ = BS16# asserted causes second bus cycle

*For this case, 8086, 8088, 80186, 80188, 80286 transfer lb first, then hb.

The definition of each bus cycle is given by three definition signals: M/IO#, W/R# and D/C#. At the same time, a valid address is present on the byte enable signals BE0# – BE3# and other address signals A2–A31. A status signal, ADS#, indicates when the 386 Microprocessor issues a new bus cycle definition and address.

Collectively, the address bus, data bus and all associated control signals are referred to simply as "the bus".

When active, the bus performs one of the bus cycles below:

1) read from memory space
2) locked read from memory space
3) write to memory space
4) locked write to memory space
5) read from I/O space (or coprocessor)
6) write to I/O space (or coprocessor)
7) interrupt acknowledge
8) indicate halt, or indicate shutdown

Table 5-2 shows the encoding of the bus cycle definition signals for each bus cycle. See section **5.2.5 Bus Cycle Definition**.

The data bus has a dynamic sizing feature supporting 32- and 16-bit bus size. Data bus size is indicated to the 386 Microprocessor using its Bus Size 16 (BS16#) input. All bus functions can be performed with either data bus size.

When the 386 Microprocessor bus is not performing one of the activities listed above, it is either Idle or in the Hold Acknowledge state, which may be detected by external circuitry. The idle state can be identified by the 386 Microprocessor giving no further assertions on its address strobe output (ADS#) since the beginning of its most recent bus cycle, and the most recent bus cycle has been terminated. The hold acknowledge state is identified by the 386 Microprocessor asserting its hold acknowledge (HLDA) output.

The shortest time unit of bus activity is a bus state. A bus state is one processor clock period (two CLK2 periods) in duration. A complete data transfer occurs during a bus cycle, composed of two or more bus states.

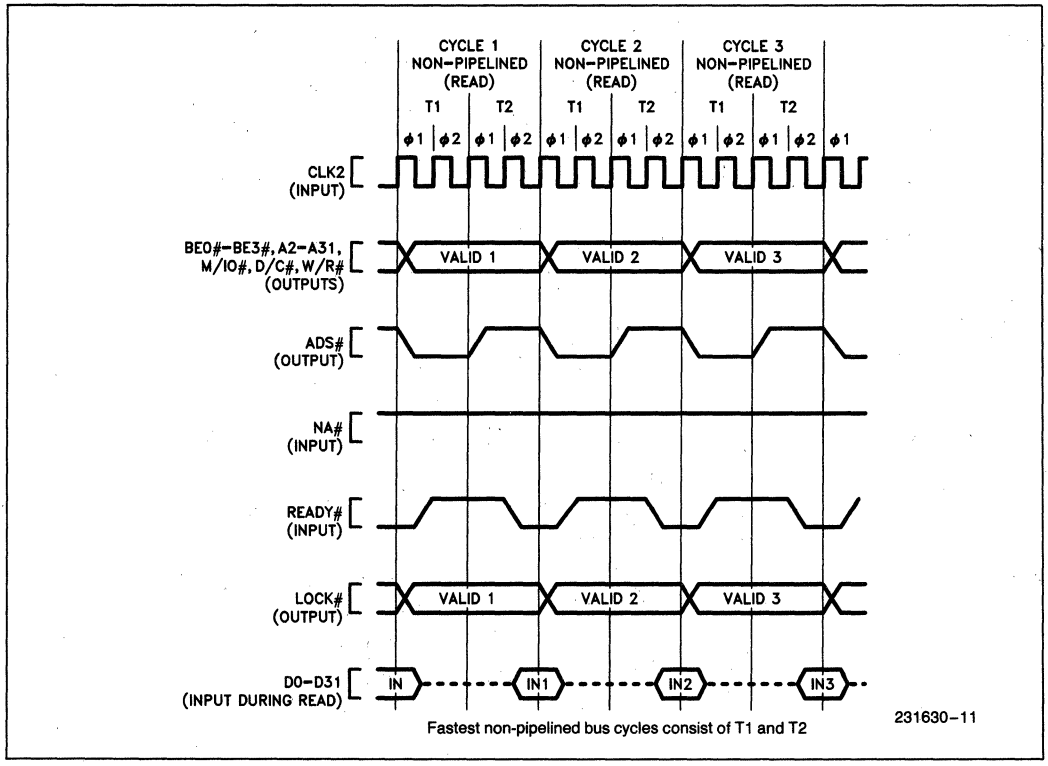


Figure 5-8. Fastest Read Cycles with Non-Pipelined Address Timing

The fastest 386 Microprocessor bus cycle requires only two bus states. For example, three consecutive bus read cycles, each consisting of two bus states, are shown by Figure 5-8. The bus states in each cycle are named **T1** and **T2**. Any memory or I/O address may be accessed by such a two-state bus cycle, if the external hardware is fast enough. The high-bandwidth, two-clock bus cycle realizes the full potential of fast main memory, or cache memory.

Every bus cycle continues until it is acknowledged by the external system hardware, using the 386 Microprocessor READY# input. Acknowledging the bus cycle at the end of the first T2 results in the shortest bus cycle, requiring only T1 and T2. If READY# is not immediately asserted, however, T2 states are repeated indefinitely until the READY# input is sampled asserted.

5.4.2 Address Pipelining

The address pipelining option provides a choice of bus cycle timings. Pipelined or non-pipelined address timing is selectable on a cycle-by-cycle basis with the Next Address (NA#) input.

When address pipelining is not selected, the current address and bus cycle definition remain stable throughout the bus cycle.

When address pipelining is selected, the address (BE0#–BE3#, A2–A31) and definition (W/R#, D/C# and M/IO#) of the next cycle are available before the end of the current cycle. To signal their availability, the 386 Microprocessor address status output (ADS#) is also asserted. Figure 5-9 illustrates the fastest read cycles with pipelined address timing.

Note from Figure 5-9 the fastest bus cycles using pipelined address require only two bus states, named **T1P** and **T2P**. Therefore cycles with pipelined address timing allow the same data bandwidth as non-pipelined cycles, but address-to-data access time is increased compared to that of a non-pipelined cycle.

By increasing the address-to-data access time, pipelined address timing reduces wait state requirements. For example, if one wait state is required with non-pipelined address timing, no wait states would be required with pipelined address.

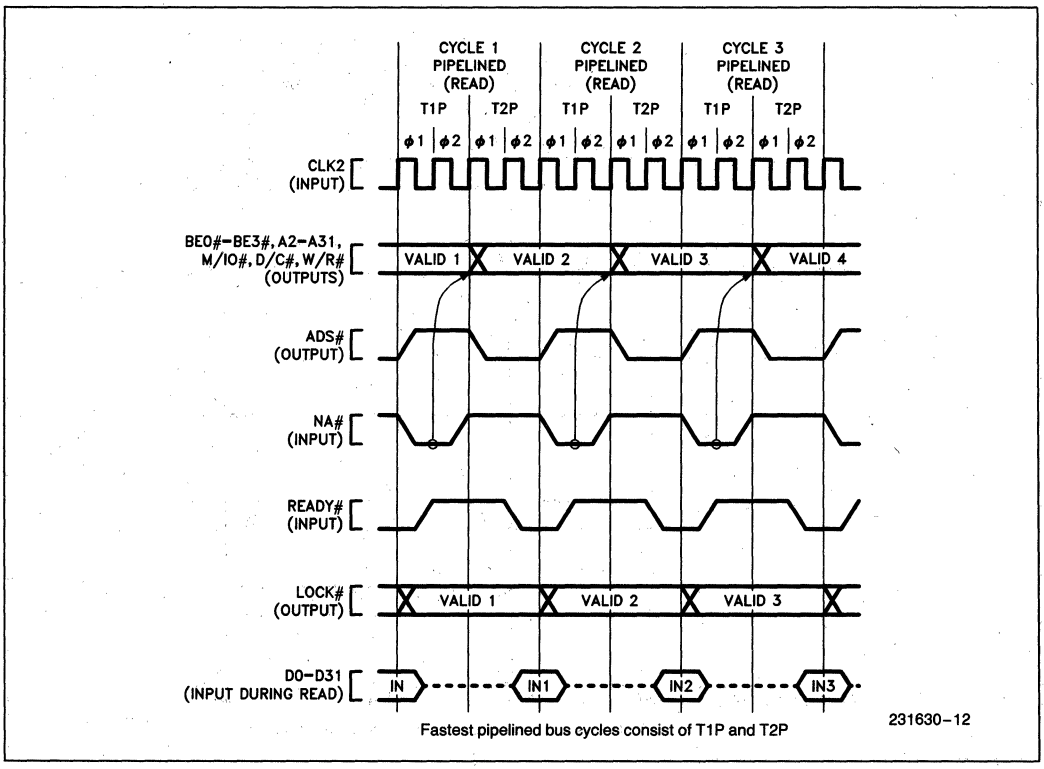


Figure 5-9. Fastest Read Cycles with Pipelined Address Timing

Pipelined address timing is useful in typical systems having address latches. In those systems, once an address has been latched, pipelined availability of the next address allows decoding circuitry to generate chip selects (and other necessary select signals) in advance, so selected devices are accessed immediately when the next cycle begins. In other words, the decode time for the next cycle can be overlapped with the end of the current cycle.

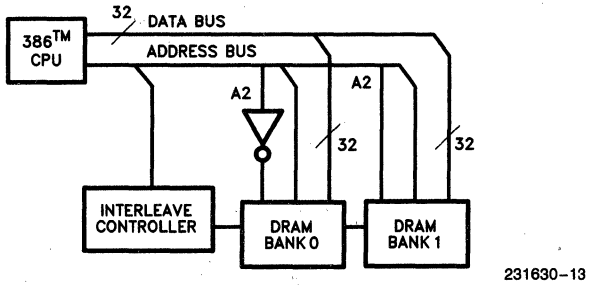
If a system contains a memory structure of two or more interleaved memory banks, pipelined address timing potentially allows even more overlap of activity. This is true when the interleaved memory controller is designed to allow the next memory operation to begin in one memory bank while the current bus cycle is still activating another memory bank. Figure 5-10 shows the general structure of the 386 Microprocessor with 2-bank and 4-bank interleaved memory. Note each memory bank of the interleaved memory has full data bus width (32-bit data width typically, unless 16-bit bus size is selected).

Further details of pipelined address timing are given in **5.4.3.4 Pipelined Address**, **5.4.3.5 Initiating and Maintaining Pipelined Address**, **5.4.3.6 Pipelined Address with Dynamic Bus Sizing**, and **5.4.3.7 Maximum Pipelined Address Usage with 16-Bit Bus Size**.

TWO-BANK INTERLEAVED MEMORY

a) Address signal A2 selects bank

b) 32-bit datapath to each bank



FOUR-BANK INTERLEAVED MEMORY

a) Address signals A3 and A2 select bank

b) 32-bit datapath to each bank

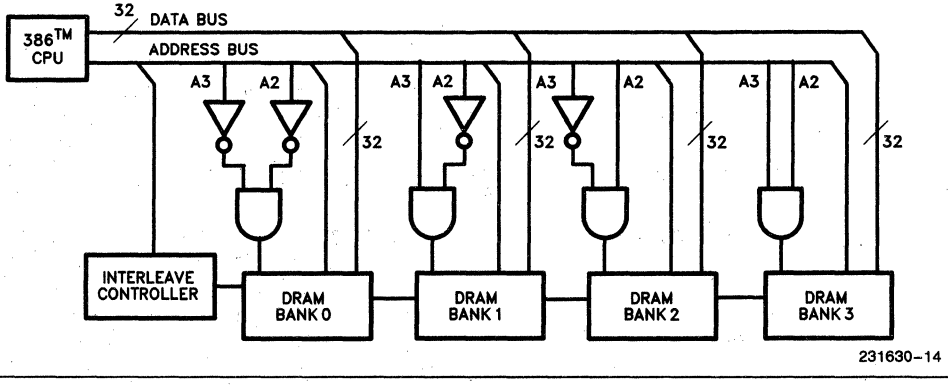


Figure 5-10. 2-Bank and 4-Bank Interleaved Memory Structure

5.4.3 Read and Write Cycles

5.4.3.1 INTRODUCTION

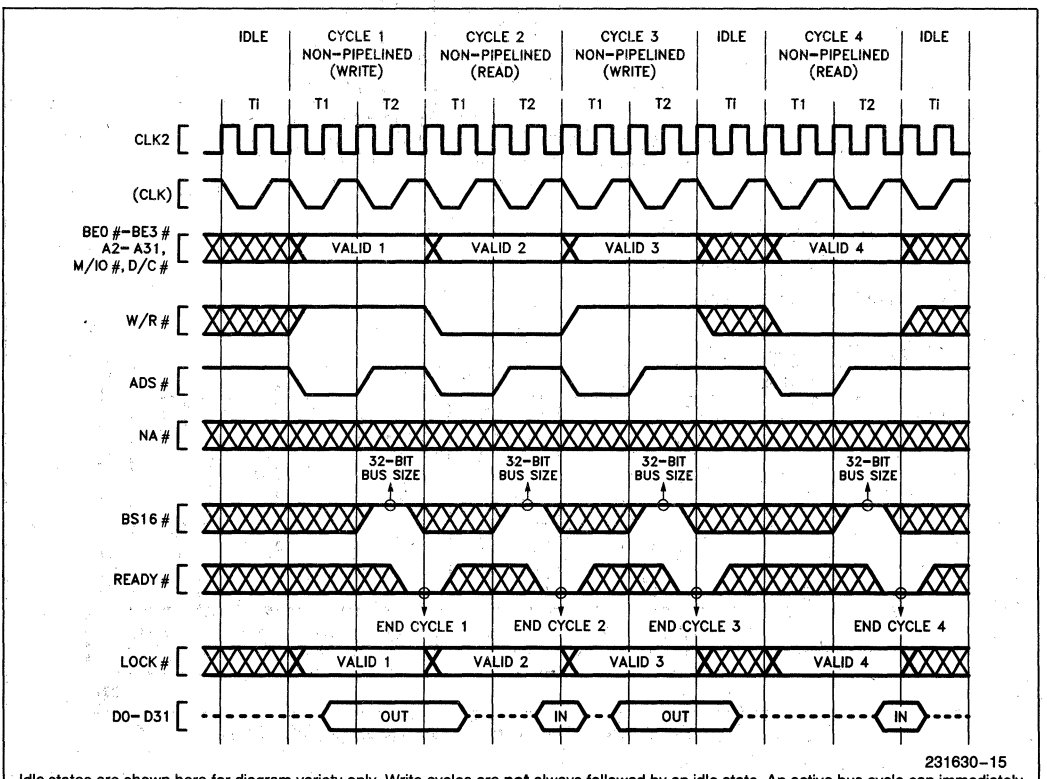
Data transfers occur as a result of bus cycles, classified as read or write cycles. During read cycles, data is transferred from an external device to the processor. During write cycles data is transferred in the other direction, from the processor to an external device.

Two choices of address timing are dynamically selectable: non-pipelined, or pipelined. After a bus idle state, the processor always uses non-pipelined address timing. However, the NA# (Next Address) input may be asserted to select pipelined address timing for the next bus cycle. When pipelining is selected and the 386 Microprocessor has a bus request pending internally, the address and definition of the next cycle is made available even before the current bus cycle is acknowledged by READY#. Generally, the NA# input is sampled each bus cycle to select the desired address timing for the next bus cycle.

Two choices of physical data bus width are dynamically selectable: 32 bits, or 16 bits. Generally, the BS16# (Bus Size 16) input is sampled near the end of the bus cycle to confirm the physical data bus size applicable to the current cycle. Negation of BS16# indicates a 32-bit size, and assertion indicates a 16-bit bus size.

If 16-bit bus size is indicated, the 386 Microprocessor automatically responds as required to complete the transfer on a 16-bit data bus. Depending on the size and alignment of the operand, another 16-bit bus cycle may be required. Table 5-7 provides all details. When necessary, the 386 Microprocessor performs an additional 16-bit bus cycle, using D0–D15 in place of D16–D31.

Terminating a read cycle or write cycle, like any bus cycle, requires acknowledging the cycle by asserting the READY# input. Until acknowledged, the processor inserts wait states into the bus cycle, to allow adjustment for the speed of any external device. External hardware, which has decoded the address and bus cycle type asserts the READY# input at the appropriate time.



231630-15

Idle states are shown here for diagram variety only. Write cycles are **not** always followed by an idle state. An active bus cycle can immediately follow the write cycle.

Figure 5-11. Various Bus Cycles and Idle States with Non-Pipelined Address (zero wait states)

At the end of the second bus state within the bus cycle, READY# is sampled. At that time, if external hardware acknowledges the bus cycle by asserting READY#, the bus cycle terminates as shown in Figure 5-11. If READY# is negated as in Figure 5-12, the cycle continues another bus state (a wait state) and READY# is sampled again at the end of that state. This continues indefinitely until the cycle is acknowledged by READY# asserted.

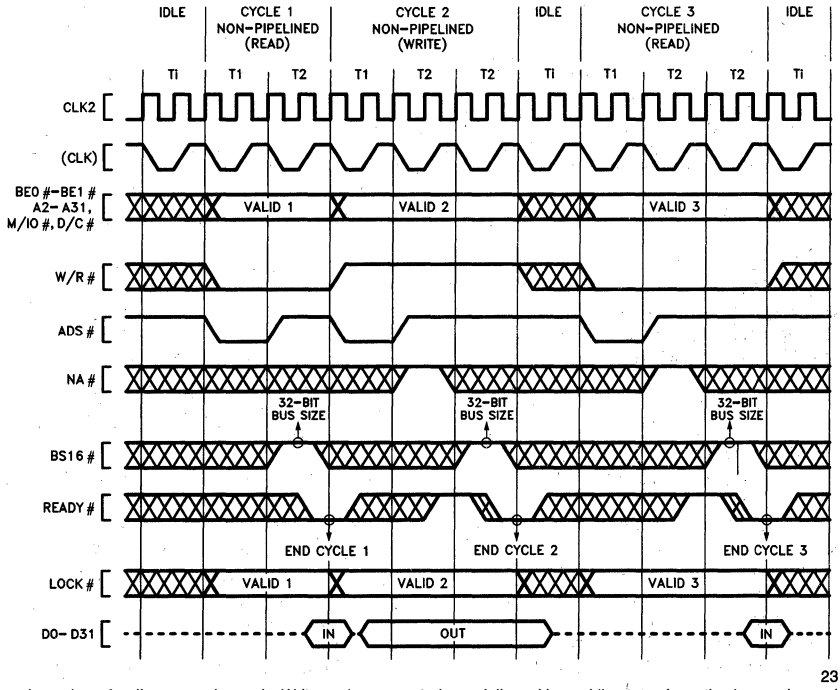
When the current cycle is acknowledged, the 386 Microprocessor terminates it. When a read cycle is acknowledged, the 386 Microprocessor latches the information present at its data pins. When a write cycle is acknowledged, the 386 Microprocessor write data remains valid throughout phase one of the next bus state, to provide write data hold time.

5.4.3.2 NON-PIPELINED ADDRESS

Any bus cycle may be performed with non-pipelined address timing. For example, Figure 5-11 shows a mixture of read and write cycles with non-pipelined address timing. Figure 5-11 shows the fastest possible cycles with non-pipelined address have two bus states per bus cycle. The states are named T1 and T2. In phase one of the T1, the address signals and bus cycle definition signals are driven valid, and to signal their availability, address status (ADS#) is simultaneously asserted.

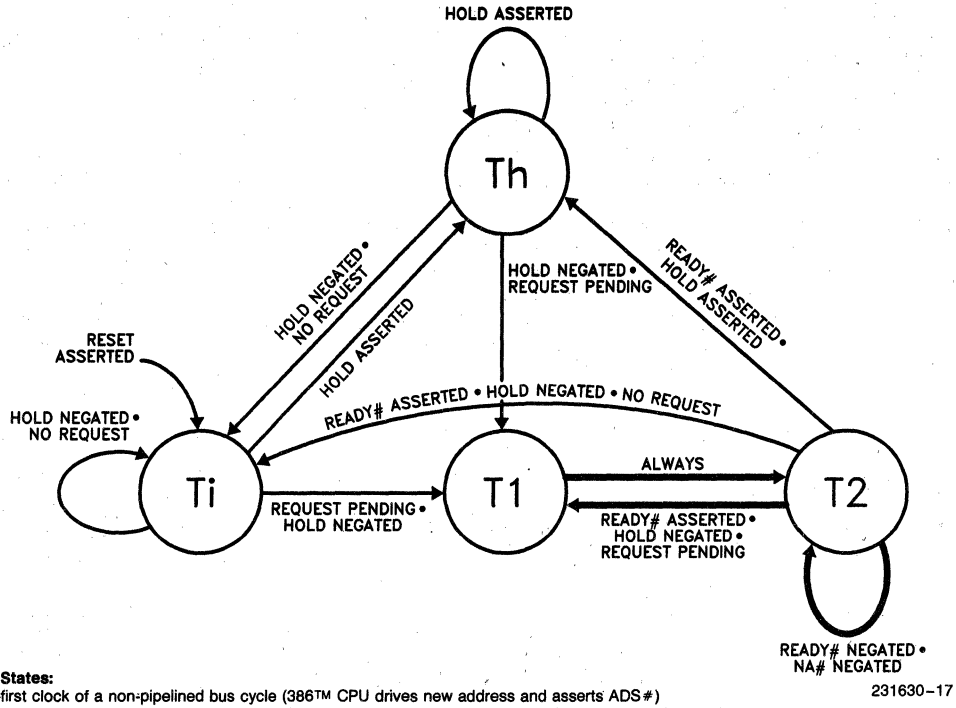
During read or write cycles, the data bus behaves as follows. If the cycle is a read, the 386 Microprocessor floats its data signals to allow driving by the external device being addressed. **The 386 Microprocessor requires that all data bus pins be at a valid logic state (high or low) at the end of each read cycle, when READY# is asserted. The system MUST be designed to meet this requirement.** If the cycle is a write, data signals are driven by the 386 Microprocessor beginning in phase two of T1 until phase one of the bus state following cycle acknowledgment.

Figure 5-12 illustrates non-pipelined bus cycles with one wait added to cycles 2 and 3. READY# is sampled negated at the end of the first T2 in cycles 2 and 3. Therefore cycles 2 and 3 have T2 repeated. At the end of the second T2, READY# is sampled asserted.



Idle states are shown here for diagram variety only. Write cycles are **not** always followed by an idle state. An active bus cycle can immediately follow the write cycle.

Figure 5-12. Various Bus Cycles and Idle States with Non-Pipelined Address (various number of wait states)



Bus States:
T1—first clock of a non-pipelined bus cycle (386™ CPU drives new address and asserts ADS#)
T2—subsequent clocks of a bus cycle when NA# has not been sampled asserted in the current bus cycle
Ti— idle state
Th—hold acknowledge state (386™ CPU asserts HLDA)
The fastest bus cycle consists of two states: T1 and T2.
Four basic bus states describe bus operation when not using pipelined address. These states do include BS16# usage for 32-bit and 16-bit bus size. If asserting BS16# requires a second 16-bit bus cycle to be performed, it is performed before HOLD asserted is acknowledged.

Figure 5-13. 386™ Microprocessor Bus States (not using pipelined address)

When address pipelining is not used, the address and bus cycle definition remain valid during all wait states. When wait states are added and you desire to maintain non-pipelined address timing, it is necessary to negate NA# during each T2 state except the last one, as shown in Figure 5-12 cycles 2 and 3. If NA# is sampled asserted during a T2 other than the last one, the next state would be T2I (for pipelined address) or T2P (for pipelined address) instead of another T2 (for non-pipelined address).

When address pipelining is not used, the bus states and transitions are completely illustrated by Figure 5-13. The bus transitions between four possible states: T1, T2, Ti, and Th. Bus cycles consist of T1 and T2, with T2 being repeated for wait states. Otherwise, the bus may be idle, in the Ti state, or in hold acknowledge, the Th state.

When address pipelining is not used, the bus state diagram is as shown in Figure 5-13. When the bus is idle it is in state Ti. Bus cycles always begin with T1. T1 always leads to T2. If a bus cycle is not acknowledged during T2 and NA# is negated, T2 is repeated. When a cycle is acknowledged during T2, the following state will be T1 of the next bus cycle if a bus request is pending internally, or Ti if there is no bus request pending, or Th if the HOLD input is being asserted.

The bus state diagram in Figure 5-13 also applies to the use of BS16#. If the 386 Microprocessor makes internal adjustments for 16-bit bus size, the adjustments do not affect the external bus states. If an additional 16-bit bus cycle is required to complete a transfer on a 16-bit bus, it also follows the state transitions shown in Figure 5-13.

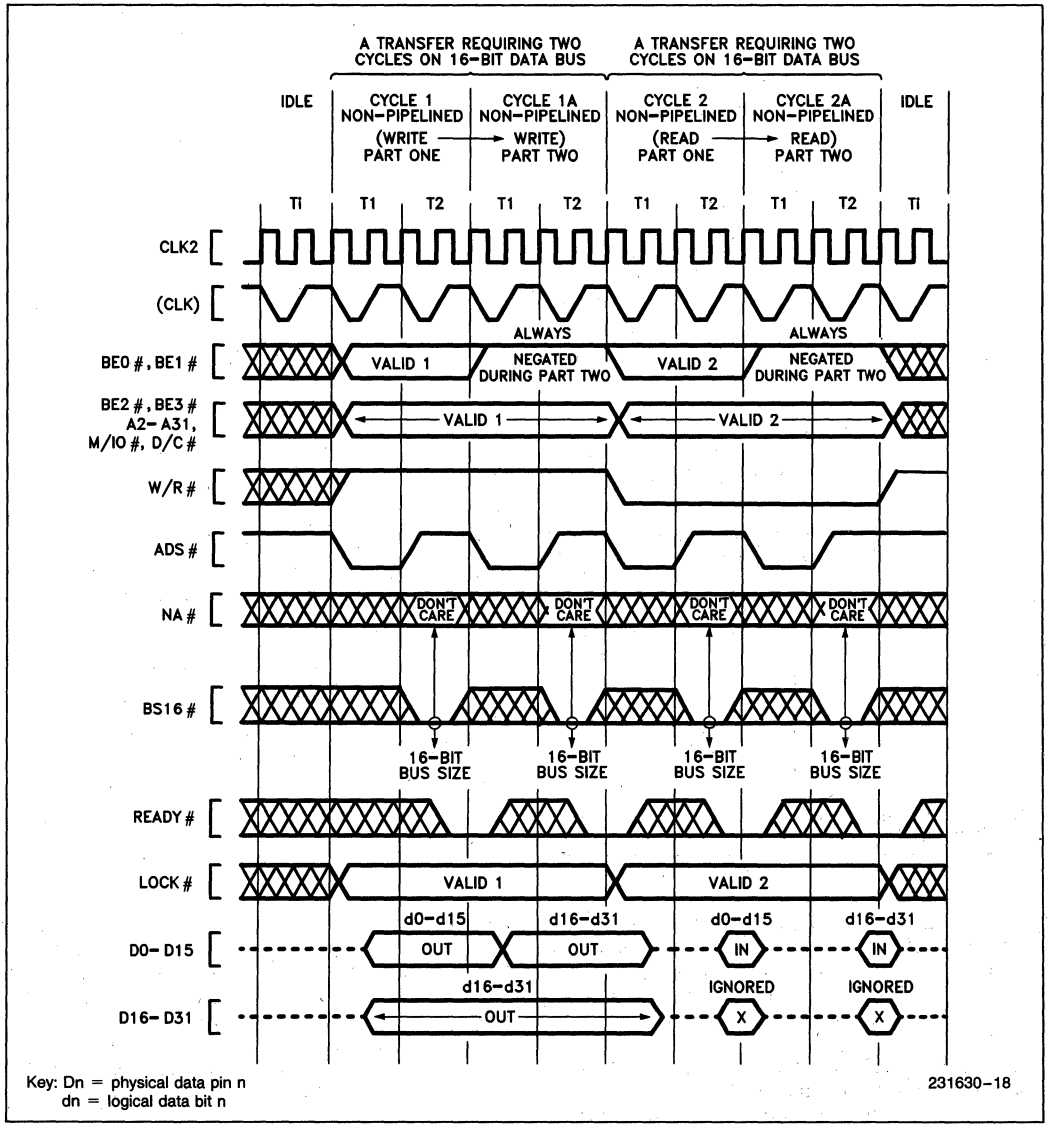
Use of pipelined address allows the 386 Microprocessor to enter three additional bus states not shown in Figure 5-13. Figure 5-20 in **5.4.3.4 Pipelined Address** is the complete bus state diagram, including pipelined address cycles.

5.4.3.3 NON-PIPELINED ADDRESS WITH DYNAMIC DATA BUS SIZING

The physical data bus width for any non-pipelined bus cycle can be either 32-bits or 16-bits. At the beginning of the bus cycle, the processor behaves as if the data bus is 32-bits wide. When the bus cycle is acknowledged, by asserting READY# at the end of a T2 state, the most recent sampling of BS16# determines the data bus size for the cycle being acknowledged. If BS16# was most recently negated, the physical data bus size is defined as 32 bits. If BS16# was most recently asserted, the size is defined as 16 bits.

When BS16# is asserted and two 16-bit bus cycles are required to complete the transfer, BS16# must be asserted during the second cycle; 16-bit bus size is not assumed. Like any bus cycle, the second 16-bit cycle must be acknowledged by asserting READY#.

When a second 16-bit bus cycle is required to complete the transfer over a 16-bit bus, the addresses



Key: Dn = physical data pin n
dn = logical data bit n

231630-18

Figure 5-14. Asserting BS16# (zero wait states, non-pipelined address)

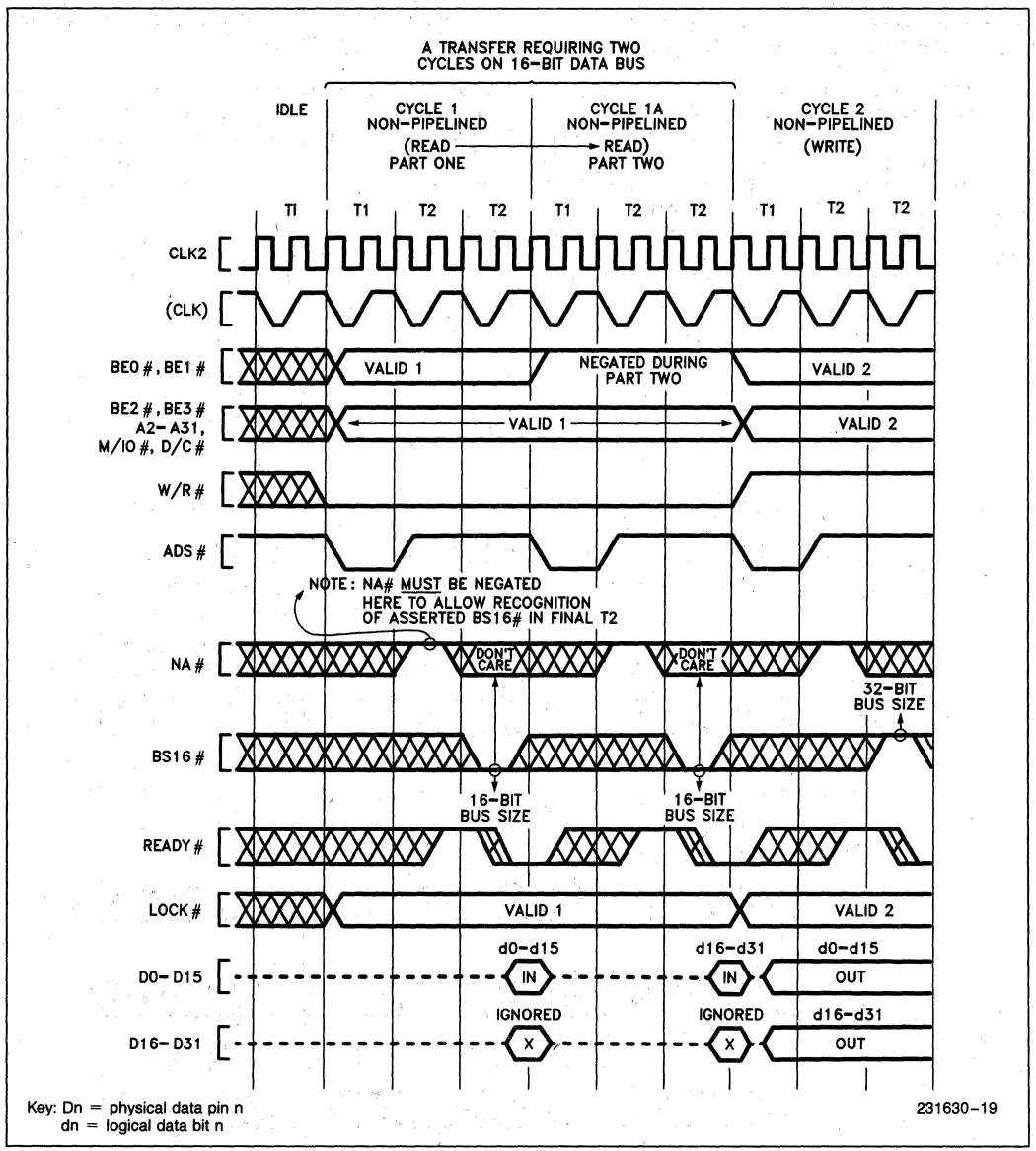


Figure 5-15. Asserting BS16# (one wait state, non-pipelined address)

generated for the two 16-bit bus cycles are closely related to each other. The addresses are the same except BE0# and BE1# are always negated for the second cycle. This is because data on D0–D15 was already transferred during the first 16-bit cycle.

Figures 5-14 and 5-15 show cases where assertion of BS16# requires a second 16-bit cycle for complete operand transfer. Figure 5-14 illustrates cycles without wait states. Figure 5-15 illustrates cycles with one wait state. In Figure 5-15 cycle 1, the bus cycle during which BS16# is asserted, note that NA# must be negated in the T2 state(s) prior to the last T2 state. This is to allow the recognition of BS16# asserted in the final T2 state. The relation of NA# and BS16# is given fully in **5.4.3.4 Pipelined Address**, but Figure 5-15 illustrates this only precaution you need to know when using BS16# with non-pipelined address.

5.4.3.4 PIPELINED ADDRESS

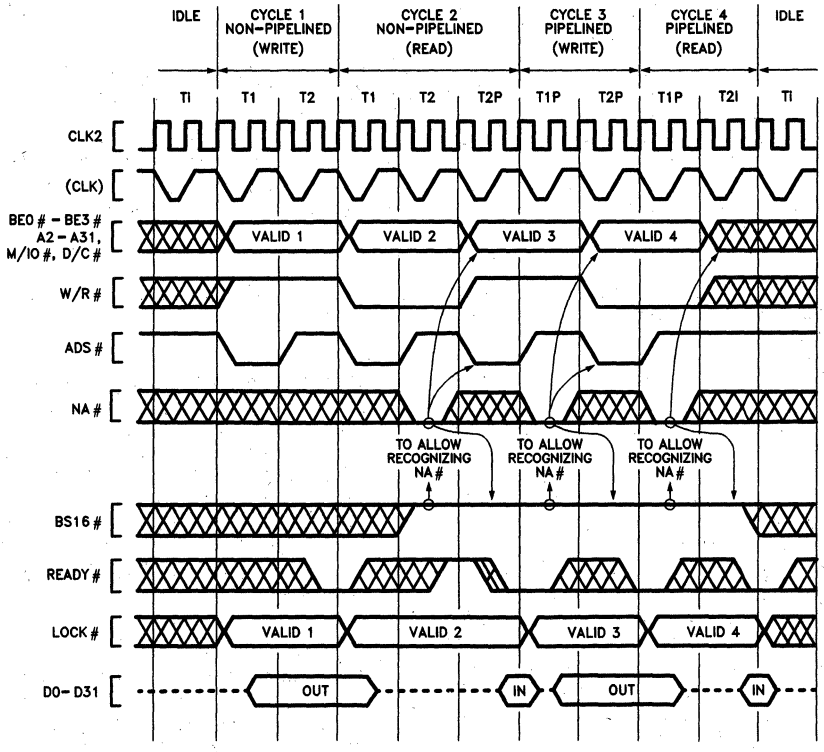
Address pipelining is the option of requesting the address and the bus cycle definition of the next, internally pending bus cycle before the current bus cycle is acknowledged with READY# asserted. ADS# is asserted by the 386 Microprocessor when the next address is issued. The address pipelining option is controlled on a cycle-by-cycle basis with the NA# input signal.

Once a bus cycle is in progress and the current address has been valid for at least one entire bus state, the NA# input is sampled at the end of every phase one until the bus cycle is acknowledged. During non-pipelined bus cycles, therefore, NA# is sampled at the end of phase one in every T2. An example is Cycle 2 in Figure 5-16, during which NA# is sampled at the end of phase one of every T2 (it was asserted once during the first T2 and has no further effect during that bus cycle).

If NA# is sampled asserted, the 386 Microprocessor is free to drive the address and bus cycle definition of the next bus cycle, and assert ADS#, as soon as it has a bus request internally pending. It may drive the next address as early as the next bus state, whether the current bus cycle is acknowledged at that time or not.

Regarding the details of address pipelining, the 386 Microprocessor has the following characteristics:

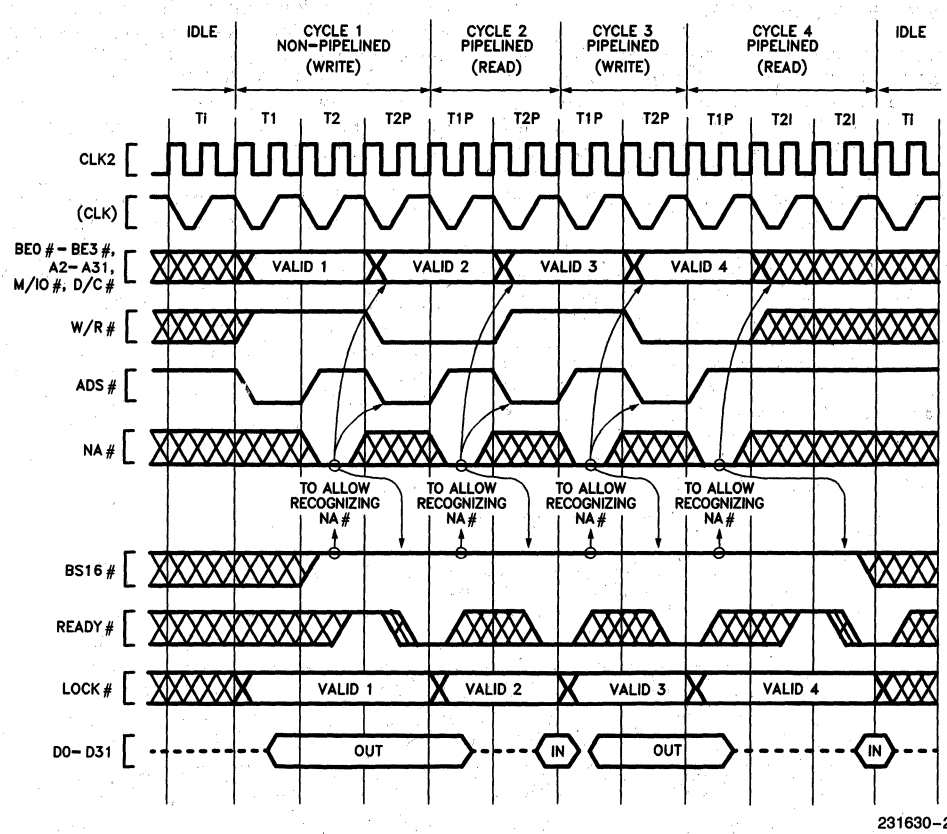
1) For NA# to be sampled asserted, BS16# must be negated at that sampling window (see Figure 5-16 Cycles 2 through 4, and Figure 5-17 Cycles 1 through 4). If NA# and BS16# are both sampled asserted during the last T2 period of a bus cycle, BS16# asserted has priority. Therefore, if both are asserted, the current bus size is taken to be 16 bits and the next address is not pipelined.



231630-20

Following any idle bus state (Ti), addresses are non-pipelined. Within non-pipelined bus cycles, NA# is only sampled during wait states. Therefore, to begin address pipelining during a group of non-pipelined bus cycles requires a non-pipelined cycle with at least one wait state (Cycle 2 above).

Figure 5-16. Transitioning to Pipelined Address During Burst of Bus Cycles



Following any idle bus state (Ti) the address is always non-pipelined and NA# is only sampled during wait states. To start address pipelining after an idle state requires a non-pipelined cycle with at least one wait state (cycle 1 above).
The pipelined cycles (2, 3, 4 above) are shown with various numbers of wait states.

Figure 5-17. Fastest Transition to Pipelined Address Following Idle Bus State

2) The next address may appear as early as the bus state after NA# was sampled asserted (see Figures 5-16 or 5-17). In that case, state T2P is entered immediately. However, when there is not an internal bus request already pending, the next address will not be available immediately after NA# is asserted and T2I is entered instead of T2P (see Figure 5-19 Cycle 3). Provided the current bus cycle isn't yet acknowledged by READY# asserted, T2P will be entered as soon as the 386 Microprocessor does drive the next address. External hardware should therefore observe the ADS# output as confirmation the next address is actually being driven on the bus.

3) Once NA# is sampled asserted, the 386 Microprocessor commits itself to the highest priority bus request that is pending internally. It can no longer perform another 16-bit transfer to the same address should BS16# be asserted externally, so thereafter must assume the current bus size is 32 bits. Therefore if NA# is sampled asserted within a bus cycle, BS16# must be negated thereafter in that bus cycle (see Figures 5-16, 5-17, 5-19). Consequently, do not assert NA# during bus cycles which must have BS16# driven asserted. See **5.4.3.6 Dynamic Bus Sizing with Pipelined Address.**

4) Any address which is validated by a pulse on the 386 Microprocessor ADS# output will remain stable on the address pins for at least two processor clock periods. The 386 Microprocessor cannot produce a new address more frequently than every two processor clock periods (see Figures 5-16, 5-17, 5-19).

5) Only the address and bus cycle definition of the very next bus cycle is available. The pipelining capability cannot look further than one bus cycle ahead (see Figure 5-19 Cycle 1).

The complete bus state transition diagram, including operation with pipelined address is given by 5-20. Note it is a superset of the diagram for non-pipelined address only, and the three additional bus states for pipelined address are drawn in bold.

The fastest bus cycle with pipelined address consists of just two bus states, T1P and T2P (recall for non-pipelined address it is T1 and T2). T1P is the first bus state of a pipelined cycle.

5.4.3.5 INITIATING AND MAINTAINING PIPELINED ADDRESS

Using the state diagram Figure 5-20, observe the transitions from an idle state, Ti, to the beginning of a pipelined bus cycle, T1P. From an idle state Ti, the first bus cycle must begin with T1, and is therefore a non-pipelined bus cycle. The next bus cycle will be pipelined, however, provided NA# is asserted and the first bus cycle ends in a T2P state (the address for the next bus cycle is driven during T2P). The fastest path from an idle state to a bus cycle with pipelined address is shown in bold below:

Ti, Ti, Ti	T1 - T2 - T2P,	T1P - T2P,
idle states	non-pipelined cycle	pipelined cycle

T1-T2-T2P are the states of the bus cycle that establishes address pipelining for the next bus cycle, which begins with T1P. The same is true after a bus hold state, shown below:

Th, Th, Th,	T1 - T2 - T2P,	T1P - T2P,
hold acknowledge states	non-pipelined cycle	pipelined cycle

The transition to pipelined address is shown functionally by Figure 5-17 Cycle 1. Note that Cycle 1 is used to transition into pipelined address timing for the subsequent Cycles 2, 3 and 4, which are pipelined. The NA# input is asserted at the appropriate time to select address pipelining for Cycles 2, 3 and 4.

Once a bus cycle is in progress and the current address has been valid for one entire bus state, the NA# input is sampled at the end of every phase one until the bus cycle is acknowledged. During Figure 5-17 Cycle 1 therefore, sampling begins in T2. Once NA# is sampled asserted during the current cycle, the 386 Microprocessor is free to drive a new address and bus cycle definition on the bus as early as the next bus state. In Figure 5-16 Cycle 1 for example, the next address is driven during state T2P. Thus Cycle 1 makes the transition to pipelined address timing, since it begins with T1 but ends with T2P. Because the address for Cycle 2 is available before Cycle 2 begins, Cycle 2 is called a pipelined bus cycle, and it begins with T1P. Cycle 2 begins as soon as READY# asserted terminates Cycle 1.

Example transition bus cycles are Figure 5-17 Cycle 1 and Figure 5-16 Cycle 2. Figure 5-17 shows transition during the very first cycle after an idle bus state, which is the fastest possible transition into address pipelining. Figure 5-16 Cycle 2 shows a transition cycle occurring during a burst of bus cycles. In any case, a transition cycle is the same whenever it occurs: it consists at least of T1, T2 (you assert NA# at that time), and T2P (provided the 386 Microprocessor has an internal bus request already pending, which it almost always has). T2P states are repeated if wait states are added to the cycle.

Note three states (T1, T2 and T2P) are only required in a bus cycle performing a **transition** from non-pipelined address into pipelined address timing, for example Figure 5-17 Cycle 1. Figure 5-17 Cycles 2, 3 and 4 show that address pipelining can be maintained with two-state bus cycles consisting only of T1P and T2P.

Once a pipelined bus cycle is in progress, pipelined timing is maintained for the next cycle by asserting NA# and detecting that the 386 Microprocessor enters T2P during the current bus cycle. The current bus cycle must end in state T2P for pipelining to be maintained in the next cycle. T2P is identified by the assertion of ADS#. Figures 5-16 and 5-17 however, each show pipelining ending after Cycle 4 because Cycle 4 ends in T2I. This indicates the 386 Microprocessor didn't have an internal bus request prior to the acknowledgement of Cycle 4. If a cycle ends with a T2 or T2I, the next cycle will not be pipelined.

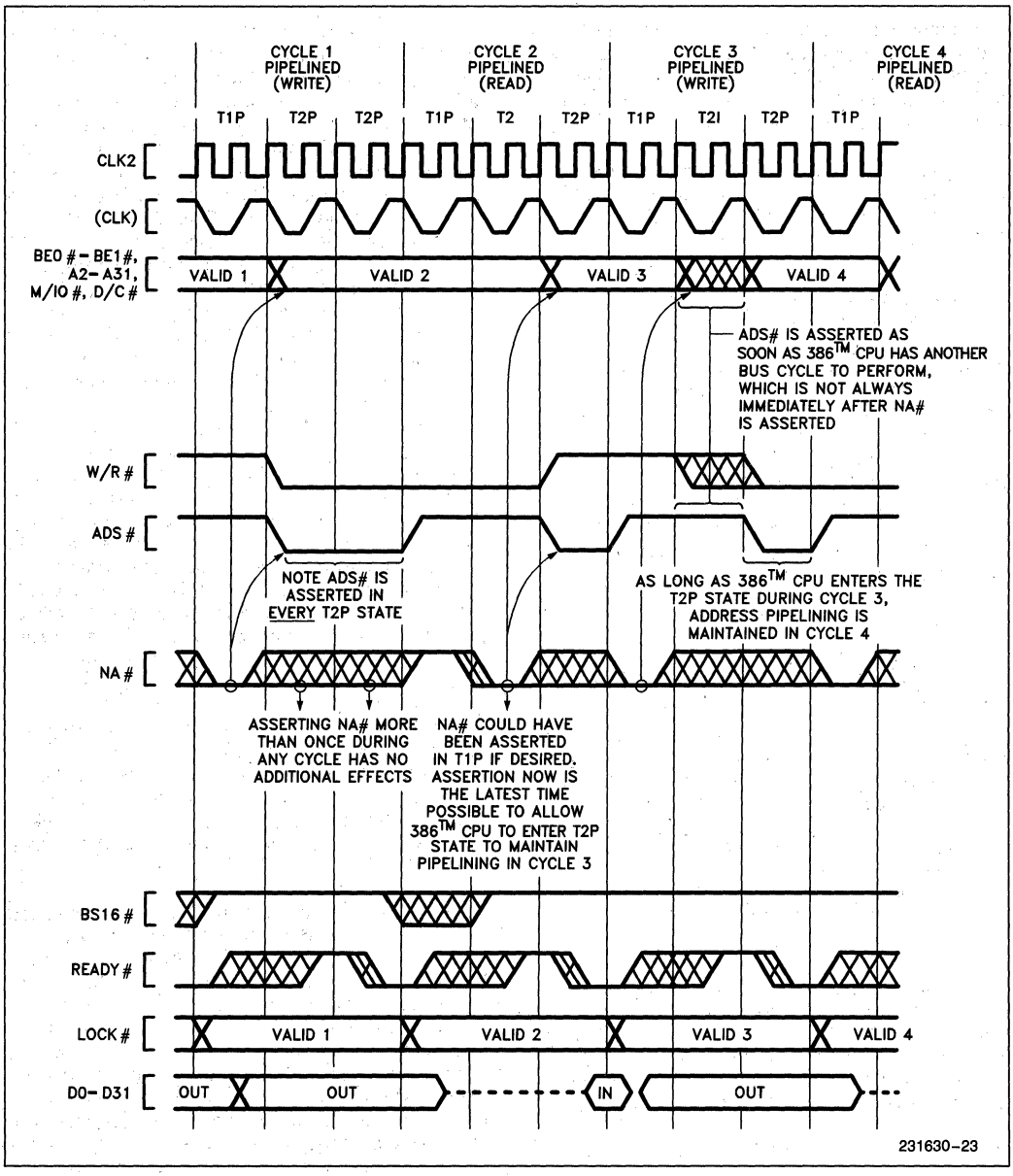
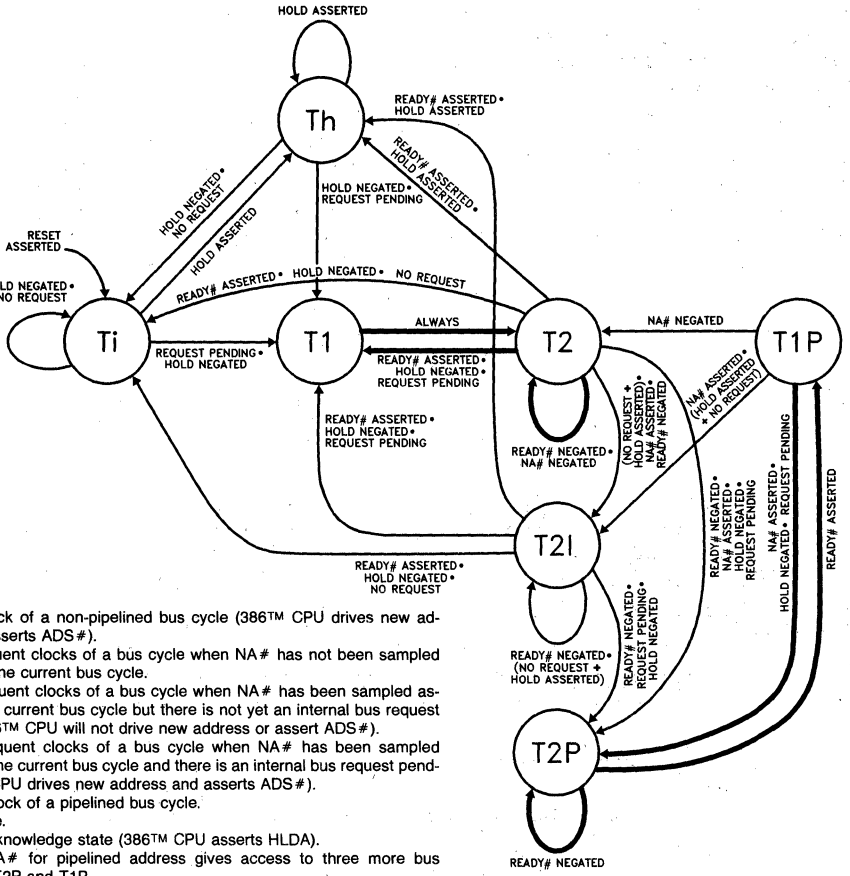


Figure 5-19. Details of Address Pipelining During Cycles with Wait States



Bus States:
T1—first clock of a non-pipelined bus cycle (386™ CPU drives new address and asserts ADS#).
T2—subsequent clocks of a bus cycle when NA# has not been sampled asserted in the current bus cycle.
T2I—subsequent clocks of a bus cycle when NA# has been sampled asserted in the current bus cycle but there is not yet an internal bus request pending (386™ CPU will not drive new address or assert ADS#).
T2P—subsequent clocks of a bus cycle when NA# has been sampled asserted in the current bus cycle and there is an internal bus request pending (386™ CPU drives new address and asserts ADS#).
T1P—first clock of a pipelined bus cycle.
Ti—idle state.
Th—hold acknowledge state (386™ CPU asserts HLDA).
Asserting NA# for pipelined address gives access to three more bus states: T2I, T2P and T1P.
Using pipelined address, the fastest bus cycle consists of T1P and T2P.

231630-24

Figure 5-20. 386™ Microprocessor Complete Bus States (including pipelined address)

Realistically, address pipelining is almost always maintained as long as NA# is sampled asserted. This is so because in the absence of any other request, a code prefetch request is always internally pending until the instruction decoder and code prefetch queue are completely full. Therefore address pipelining is maintained for long bursts of bus cycles, if the bus is available (i.e., HOLD negated) and NA# is sampled asserted in each of the bus cycles.

5.4.3.6 PIPELINED ADDRESS WITH DYNAMIC DATA BUS SIZING

The BS16# feature allows easy interface to 16-bit data buses. When asserted, the 386 Microprocessor bus interface hardware performs appropriate action to make the transfer using a 16-bit data bus connected on D0–D15.

There is a degree of interaction, however, between the use of Address Pipelining and the use of Bus Size 16. The interaction results from the multiple bus cycles required when transferring 32-bit operands over a 16-bit bus. If the operand requires both 16-bit halves of the 32-bit bus, the appropriate 386 Microprocessor action is a second bus cycle to complete the operand's transfer. It is this necessity that conflicts with NA# usage.

When NA# is sampled asserted, the 386 Microprocessor commits itself to perform the next internally

pending bus request, and is allowed to drive the next internally pending address onto the bus. Asserting NA# therefore makes it impossible for the next bus cycle to again access the current address on A2–A31, such as may be required when BS16# is asserted by the external hardware.

To avoid conflict, the 386 Microprocessor is designed with following two provisions:

1) To avoid conflict, BS16# must be negated in the current bus cycle if NA# has already been sampled asserted in the current cycle. If NA# is sampled asserted, the current data bus size is assumed to be 32 bits.
2) To also avoid conflict, if NA# and BS16# are both asserted during the same sampling window, BS16# asserted has priority and the 386 Microprocessor acts as if NA# was negated at that time. Internal 386 Microprocessor circuitry, shown conceptually in Figure 5-18, assures that BS16# is sampled asserted and NA# is sampled negated if both inputs are externally asserted at the same sampling window.

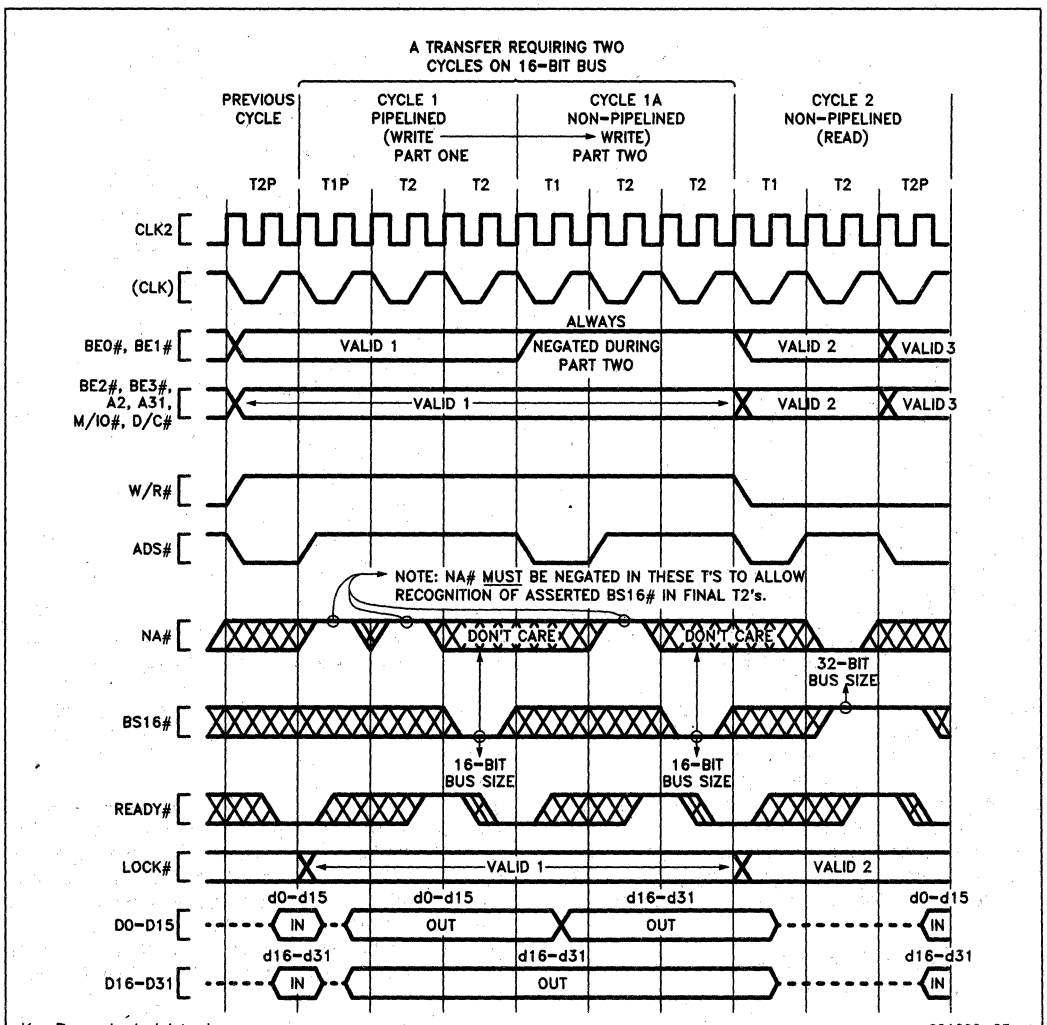


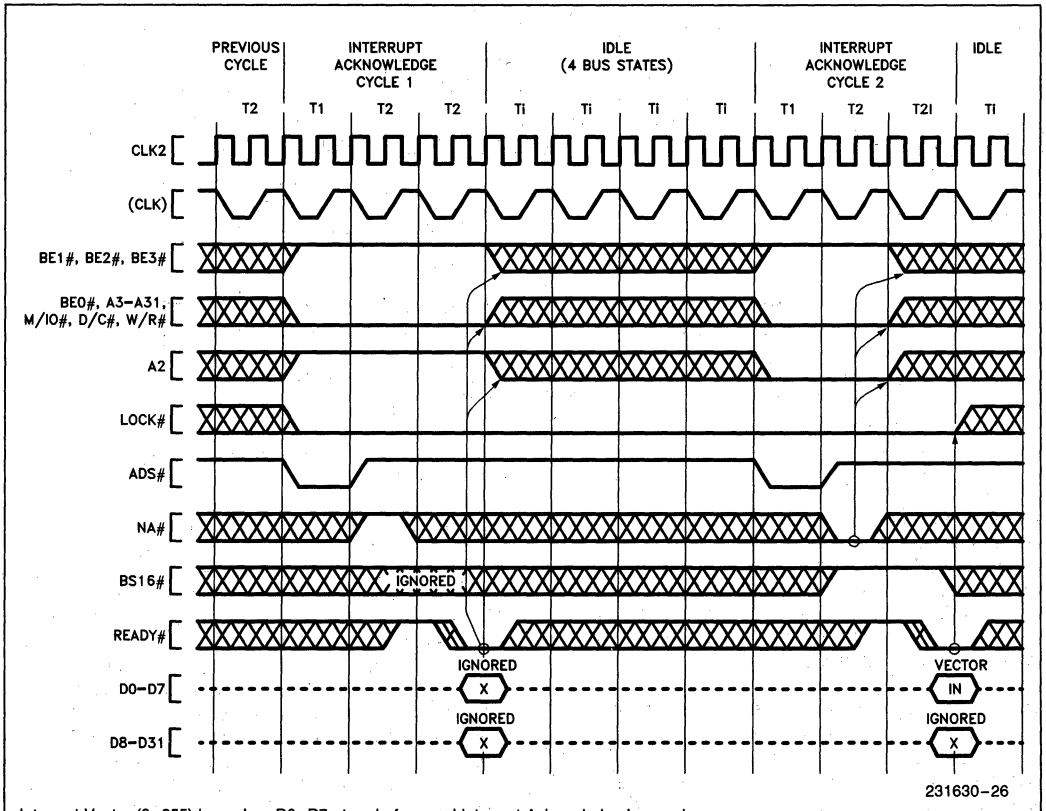
Figure 5-21. Using NA# and BS16#

Certain types of 16-bit or 8-bit operands require no adjustment for correct transfer on a 16-bit bus. Those are read or write operands using only the lower half of the data bus, and write operands using only the upper half of the bus since the 386 Microprocessor simultaneously duplicates the write data on the lower half of the data bus. For these patterns of Byte Enables and the R/W# signals, BS16# need not be asserted at the 386 Microprocessor, allowing NA# to be asserted during the bus cycle if desired.

5.4.4 Interrupt Acknowledge (INTA) Cycles

In response to an interrupt request on the INTR input when interrupts are enabled, the 386 Microprocessor performs two interrupt acknowledge cycles. These bus cycles are similar to read cycles in that bus definition signals define the type of bus activity taking place, and each cycle continues until acknowledged by READY# sampled asserted.

The state of A2 distinguishes the first and second interrupt acknowledge cycles. The byte address driven during the first interrupt acknowledge cycle is 4 (A31–A3 low, A2 high, BE3#–BE1# high, and BE0# low). The address driven during the second interrupt acknowledge cycle is 0 (A31–A2 low, BE3#–BE1# high, BE0# low).



Interrupt Vector (0–255) is read on D0–D7 at end of second Interrupt Acknowledge bus cycle.
Because each Interrupt Acknowledge bus cycle is followed by idle bus states, asserting NA# has no practical effect. Choose the approach which is simplest for your system hardware design.

Figure 5-22. Interrupt Acknowledge Cycles

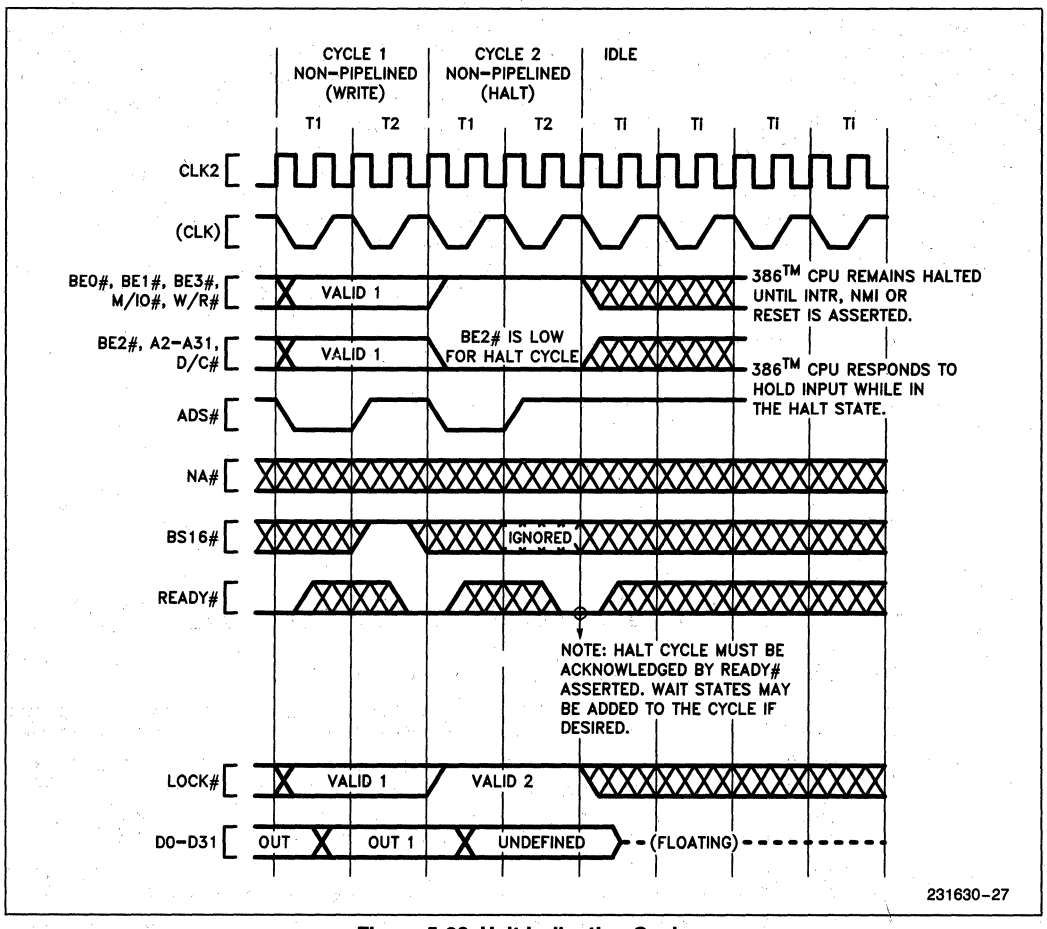


Figure 5-23. Halt Indication Cycle

The LOCK# output is asserted from the beginning of the first interrupt acknowledge cycle until the end of the second interrupt acknowledge cycle. Four idle bus states, Ti, are inserted by the 386 Microprocessor between the two interrupt acknowledge cycles, allowing for compatibility with spec TRHRL of the 8259A Interrupt Controller.

During both interrupt acknowledge cycles, D0–D31 float. No data is read at the end of the first interrupt acknowledge cycle. At the end of the second interrupt acknowledge cycle, the 386 Microprocessor will read an external interrupt vector from D0–D7 of the data bus. The vector indicates the specific interrupt number (from 0–255) requiring service.

5.4.5 Halt Indication Cycle

The 386 Microprocessor halts as a result of executing a HALT instruction. Signaling its entrance into the halt state, a halt indication cycle is performed. The halt indication cycle is identified by the state of the bus definition signals shown in **5.2.5 Bus Cycle Definition** and a byte address of 2. BE0# and BE2# are the only signals distinguishing halt indication from shutdown indication, which drives an address of 0. During the halt cycle undefined data is driven on D0–D31. The halt indication cycle must be acknowledged by READY# asserted.

A halted 386 Microprocessor resumes execution when INTR (if interrupts are enabled) or NMI or RESET is asserted.

5.4.6 Shutdown Indication Cycle

The 386 Microprocessor shuts down as a result of a protection fault while attempting to process a double fault. Signaling its entrance into the shutdown state, a shutdown indication cycle is performed. The shutdown indication cycle is identified by the state of the bus definition signals shown in **5.2.5 Bus Cycle Definition** and a byte address of 0. BE0# and BE2# are the only signals distinguishing shutdown indication from halt indication, which drives an address of 2. During the shutdown cycle undefined data is driven on D0–D31. The shutdown indication cycle must be acknowledged by READY# asserted.

A shutdown 386 Microprocessor resumes execution when NMI or RESET is asserted.

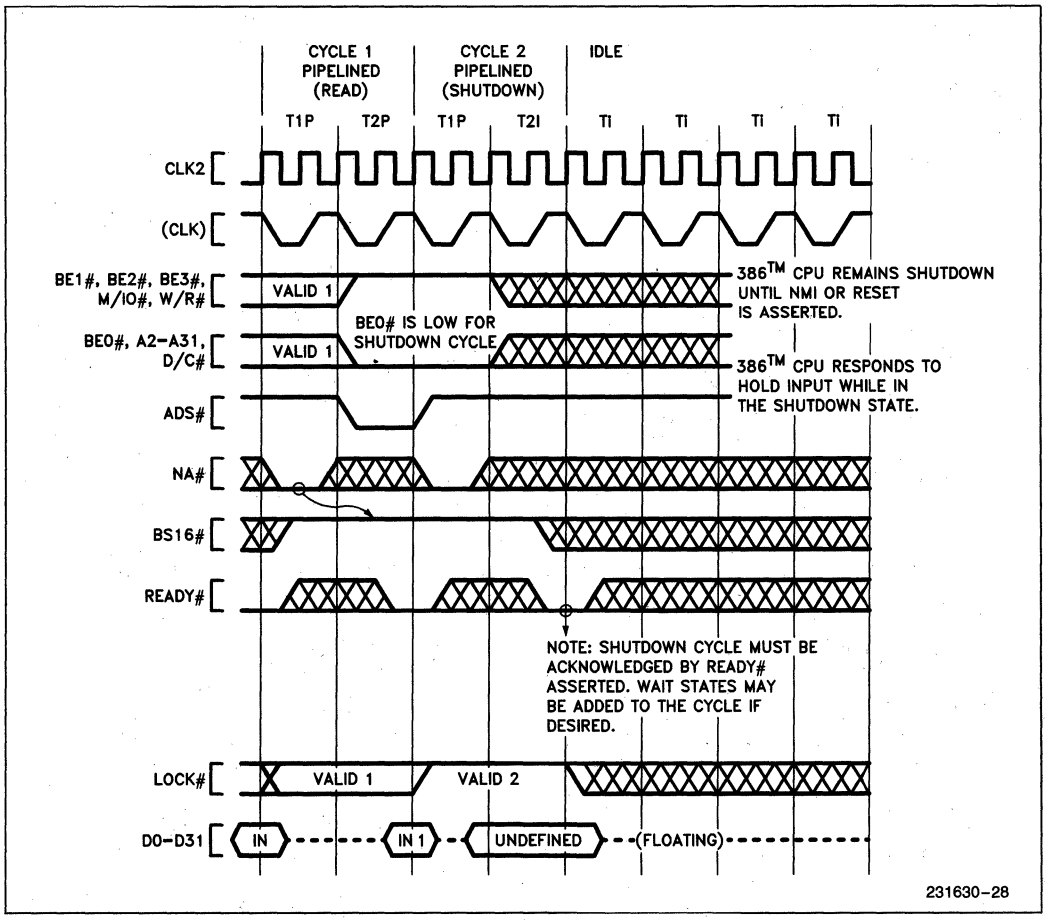
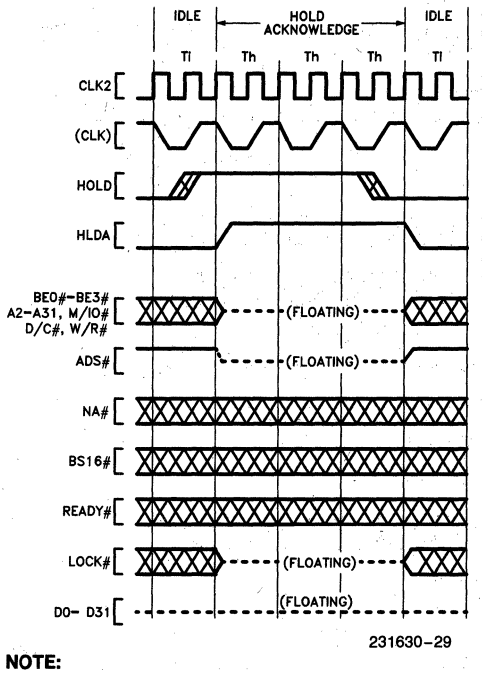


Figure 5-24. Shutdown Indication Cycle

5.5 OTHER FUNCTIONAL DESCRIPTIONS

5.5.1 Entering and Exiting Hold Acknowledge

The bus hold acknowledge state, Th, is entered in response to the HOLD input being asserted. In the bus hold acknowledge state, the 386 Microprocessor floats all output or bidirectional signals, except for HLDA. HLDA is asserted as long as the 386 Microprocessor remains in the bus hold acknowledge state. In the bus hold acknowledge state, all inputs except HOLD, RESET, BUSY#, ERROR#, and PEREQ are ignored (also up to one rising edge on NMI is remembered for processing when HOLD is no longer asserted).



NOTE:
For maximum design flexibility the 386™ Microprocessor has no internal pullup resistors on its outputs. Your design may require an external pullup on ADS# and other 386™ CPU outputs to keep them negated during float periods.

Figure 5-25. Requesting Hold from Idle Bus

Th may be entered from a bus idle state as in Figure 5-25 or after the acknowledgement of the current physical bus cycle if the LOCK# signal is not asserted, as in Figures 5-26 and 5-27. If HOLD is asserted during a locked bus cycle, the 386 Microprocessor may execute one unlocked bus cycle before acknowledging HOLD. If asserting BS16# requires a second 16-bit bus cycle to complete a physical operand transfer, it is performed before HOLD is acknowledged, although the bus state diagrams in Figures 5-13 and 5-20 do not indicate that detail.

Th is exited in response to the HOLD input being negated. The following state will be Ti as in Figure 5-25 if no bus request is pending. The following bus state will be T1 if a bus request is internally pending, as in Figures 5-26 and 5-27.

Th is also exited in response to RESET being asserted.

If a rising edge occurs on the edge-triggered NMI input while in Th, the event is remembered as a nonmaskable interrupt 2 and is serviced when Th is exited, unless of course, the 386 Microprocessor is reset before Th is exited.

5.5.2 Reset During Hold Acknowledge

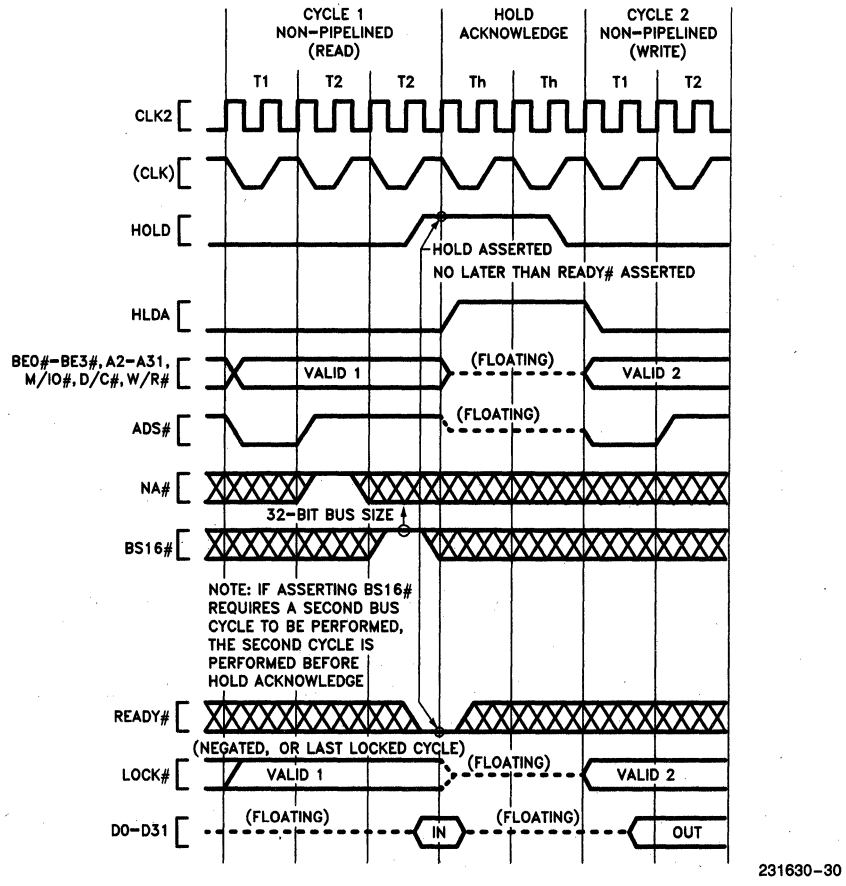
RESET being asserted takes priority over HOLD being asserted. Therefore, Th is exited in reponse to the RESET input being asserted. If RESET is asserted while HOLD remains asserted, the 386 Microprocessor drives its pins to defined states during reset, as in **Table 5-3 Pin State During Reset**, and performs internal reset activity as usual.

If HOLD remains asserted when RESET is negated, the 386 Microprocessor enters the hold acknowledge state before performing its first bus cycle, provided HOLD is still asserted when the 386 Microprocessor would otherwise perform its first bus cycle. If HOLD remains asserted when RESET is negated, the BUSY# input is still sampled as usual to determine whether a self test is being requested, and ERROR# is still sampled as usual to determine whether an 80387 vs. an 80287 (or none) is present.

5.5.3 Bus Activity During and Following Reset

RESET is the highest priority input signal, capable of interrupting any processor activity when it is asserted. A bus cycle in progress can be aborted at any stage, or idle states or bus hold acknowledge states discontinued so that the reset state is established.

RESET should remain asserted for at least 15 CLK2 periods to ensure it is recognized throughout the 386 Microprocessor, and at least 78 CLK2 periods if 386 Microprocessor self-test is going to be requested at the falling edge. RESET asserted pulses less than 15 CLK2 periods may not be recognized. RESET pulses less than 78 CLK2 periods followed by a



NOTE:
HOLD is a synchronous input and can be asserted at any CLK2 edge, provided setup and hold (t_{23} and t_{24}) requirements are met. This waveform is useful for determining Hold Acknowledge latency.

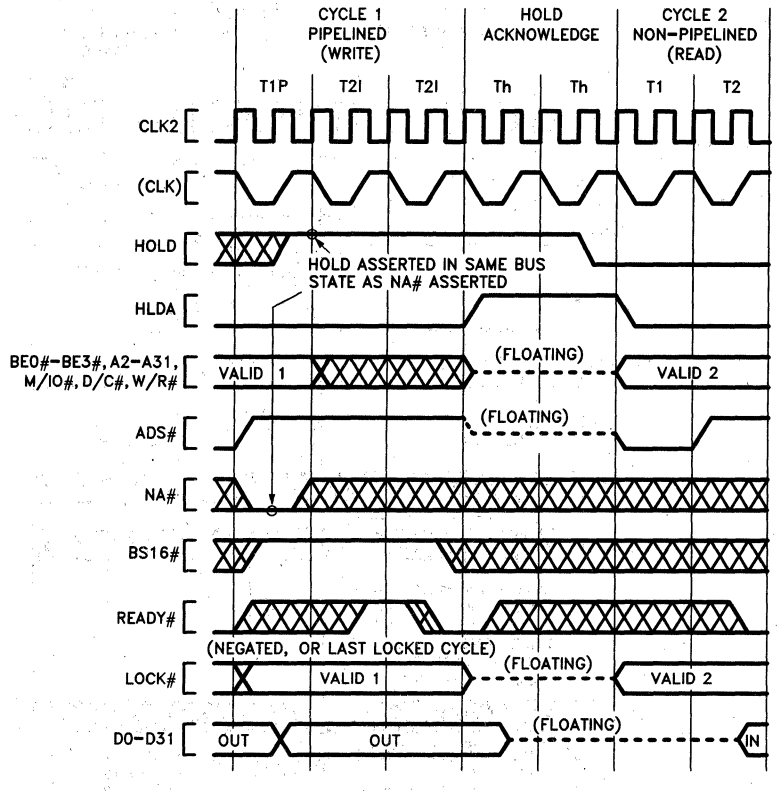
Figure 5-26. Requesting Hold from Active Bus (NA# negated)

self-test may cause the self-test to report a failure when no true failure exists. The additional RESET pulse width is required to clear additional state prior to a valid self-test.

Provided the RESET falling edge meets setup and hold times t_{25} and t_{26}, the internal processor clock phase is defined at that time, as illustrated by Figure 5-28 and Figure 7-7.

A 386 Microprocessor self-test may be requested at the time RESET is negated by having the BUSY# input at a LOW level, as shown in Figure 5-28. The self-test requires (2^{20}) + approximately 60 CLK2 periods to complete. The self-test duration is not affected by the test results. Even if the self-test indicates a problem, the 386 Microprocessor attempts to proceed with the reset sequence afterwards.

After the RESET falling edge (and after the self-test if it was requested) the 386 Microprocessor performs an internal initialization sequence for approximately 350 to 450 CLK2 periods. Also during the initialization, between the 20th CLK2 period and the first bus cycle, the ERROR# input is sampled to determine the presence of an 80387 coprocessor versus the presence of an 80287 (or no coprocessor). During this time period, BUSY# must be HIGH. To distinguish between an 80287 being present and no coprocessor being present requires a software test.



NOTE:
HOLD is a synchronous input and can be asserted at any CLK2 edge, provided setup and hold (t_{23} and t_{24}) requirements are met. This waveform is useful for determining Hold Acknowledge latency.

Figure 5-27. Requesting Hold from Active Bus (NA# asserted)

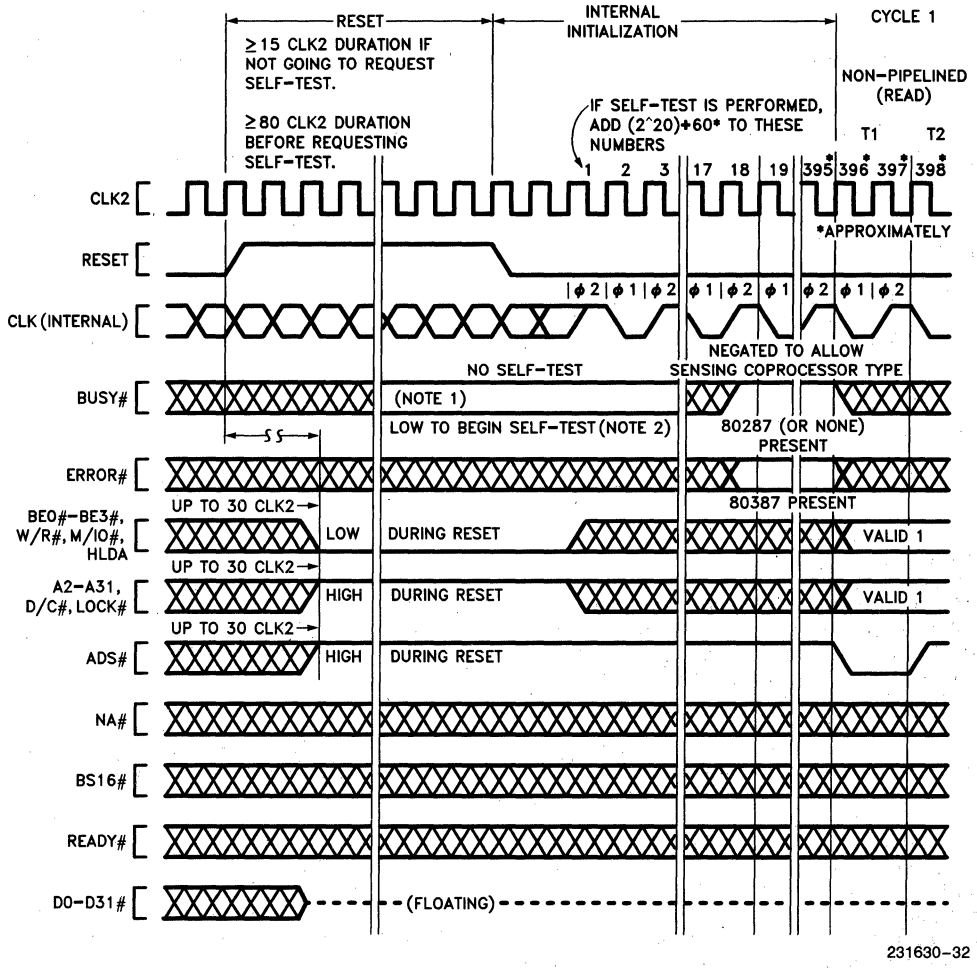
5.6 SELF-TEST SIGNATURE

Upon completion of self-test, (if self-test was requested by holding BUSY# LOW at least eight CLK2 periods before and after the falling edge of RESET), the EAX register will contain a signature of 00000000h indicating the 386 Microprocessor passed its self-test of microcode and major PLA contents with no problems detected. The passing signature in EAX, 00000000h, applies to all 386 Microprocessor revision levels. Any non-zero signature indicates the 386 Microprocessor unit is faulty.

5.7 COMPONENT AND REVISION IDENTIFIERS

To assist 386 Microprocessor users, the 386 Microprocessor after reset holds a component identifier and a revision identifier in its DX register. The upper 8 bits of DX hold 03h as identification of the 386 Microprocessor component. The lower 8 bits of DX hold an 8-bit unsigned binary number related to the component revision level. The revision identifier begins chronologically with a value zero and is subject to change (typically it will be incremented) with component steppings intended to have certain improvements or distinctions from previous steppings.

These features are intended to assist 386 Microprocessor users to a practical extent. However, the revision identifier value is not guaranteed to change with every stepping revision, or to follow a completely uniform numerical sequence, depending on the type or intention of revision, or manufacturing materials required to be changed. Intel has sole discretion over these characteristics of the component.



NOTES:
1. BUSY# should be held stable for 8 CLK2 periods before and after the CLK2 period in which RESET falling edge occurs.
2. If self-test is requested, the 386™ Microprocessor outputs remain in their reset state as shown here and in Table 5-3.

Figure 5-28. Bus Activity from Reset Until First Code Fetch

Table 5-10. Component and Revision Identifier History

386™ CPU Stepping Name	Component Identifier	Revision Identifier	386™ CPU Stepping Name	Component Identifier	Revision Identifier
B0	03	03	D0	03	05
B1	03	03			

5.8 COPROCESSOR INTERFACING

The 386 Microprocessor provides an automatic interface for the Intel 80387 numeric floating-point coprocessor. The 80387 coprocessor uses an I/O-mapped interface driven automatically by the 386 Microprocessor and assisted by three dedicated signals: BUSY#, ERROR#, and PEREQ.

As the 386 Microprocessor begins supporting a coprocessor instruction, it tests the BUSY# and ERROR# signals to determine if the coprocessor can accept its next instruction. Thus, the BUSY# and ERROR# inputs eliminate the need for any "preamble" bus cycles for communication between processor and coprocessor. The 80387 can be given its command opcode immediately. The dedicated signals provide instruction synchronization, and eliminate the need of using the 386 Microprocessor WAIT opcode (9Bh) for 80387 instruction synchronization (the WAIT opcode was required when 8086 or 8088 was used with the 8087 coprocessor).

Custom coprocessors can be included in 386 Microprocessor-based systems, via memory-mapped or I/O-mapped interfaces. Such coprocessor interfaces allow a completely custom protocol, and are not limited to a set of coprocessor protocol "primitives". Instead, memory-mapped or I/O-mapped interfaces may use all applicable 386 Microprocessor instructions for high-speed coprocessor communication. The BUSY# and ERROR# inputs of the 386 Microprocessor may also be used for the custom coprocessor interface, if such hardware assist is desired. These signals can be tested by the 386 Microprocessor WAIT opcode (9Bh). The WAIT instruction will wait until the BUSY# input is negated (interruptable by an NMI or enabled INTR input), but generates an exception 16 fault if the ERROR# pin is in the asserted state when the BUSY# goes (or is) negated. If the custom coprocessor interface is memory-mapped, protection of the addresses used for the interface can be provided with the 386 Microprocessor on-chip paging or segmentation mechanisms. If the custom interface is I/O-mapped, protection of the interface can be provided with the 386 Microprocessor IOPL (I/O Privilege Level) mechanism.

The 80387 numeric coprocessor interface is I/O mapped as shown in Table 5-11. Note that the 80387 coprocessor interface addresses are beyond the 0h-FFFFh range for programmed I/O. When the 386 Microprocessor supports the 80387 coprocessor, the 386 Microprocessor automatically generates bus cycles to the coprocessor interface addresses.

Table 5-11. Numeric Coprocessor Port Addresses

Address in 386™ CPU I/O Space	80387 Coprocessor Register
800000F8h	Opcode Register (32-bit port)
800000FCh	Operand Register (32-bit port)

To correctly map the 80387 registers to the appropriate I/O addresses, connect the 80387 CMD0# pin directly to the A2 output of the 386 Microprocessor.

5.8.1 Software Testing for Coprocessor Presence

When software is used to test for coprocessor (80387) presence, it should use only the following coprocessor opcodes: FINIT, FNINIT, FSTCW mem, FSTSW mem, FSTSW AX. To use other coprocessor opcodes when a coprocessor is known to be not present, first set EM = 1 in 386 Microprocessor CR0.

6. MECHANICAL DATA

6.1 INTRODUCTION

In this section, the physical packaging and its connections are described in detail.

6.2 PIN ASSIGNMENT

The 386 Microprocessor pinout as viewed from the top side of the component is shown by Figure 6-1. Its pinout as viewed from the Pin side of the component is Figure 6-2.

V_{CC} and GND connections must be made to multiple V_{CC} and V_{SS} (GND) pins. Each V_{CC} and V_{SS} must be connected to the appropriate voltage level. The circuit board should include V_{CC} and GND planes for power distribution and all V_{CC} and V_{SS} pins must be connected to the appropriate plane.

NOTE:
Pins identified as "N.C." should remain completely unconnected.

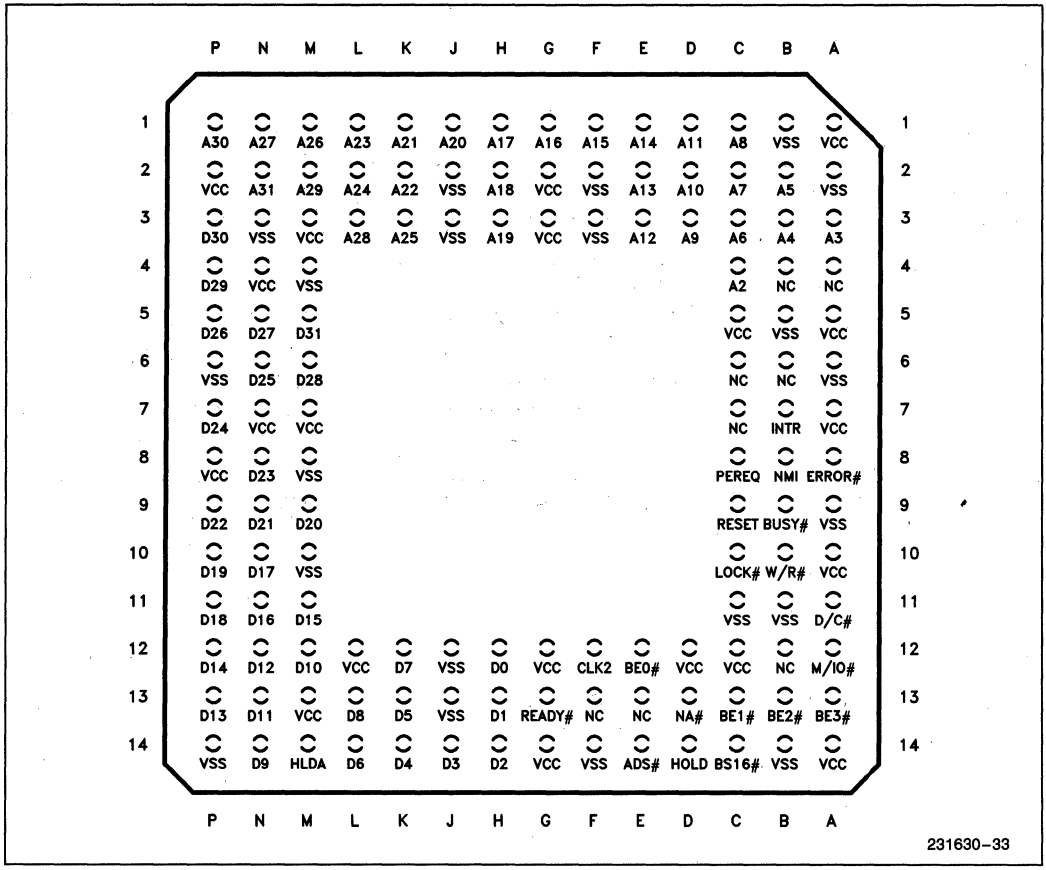


Figure 6-1. 386™ Microprocessor PGA Pinout—View from Top Side

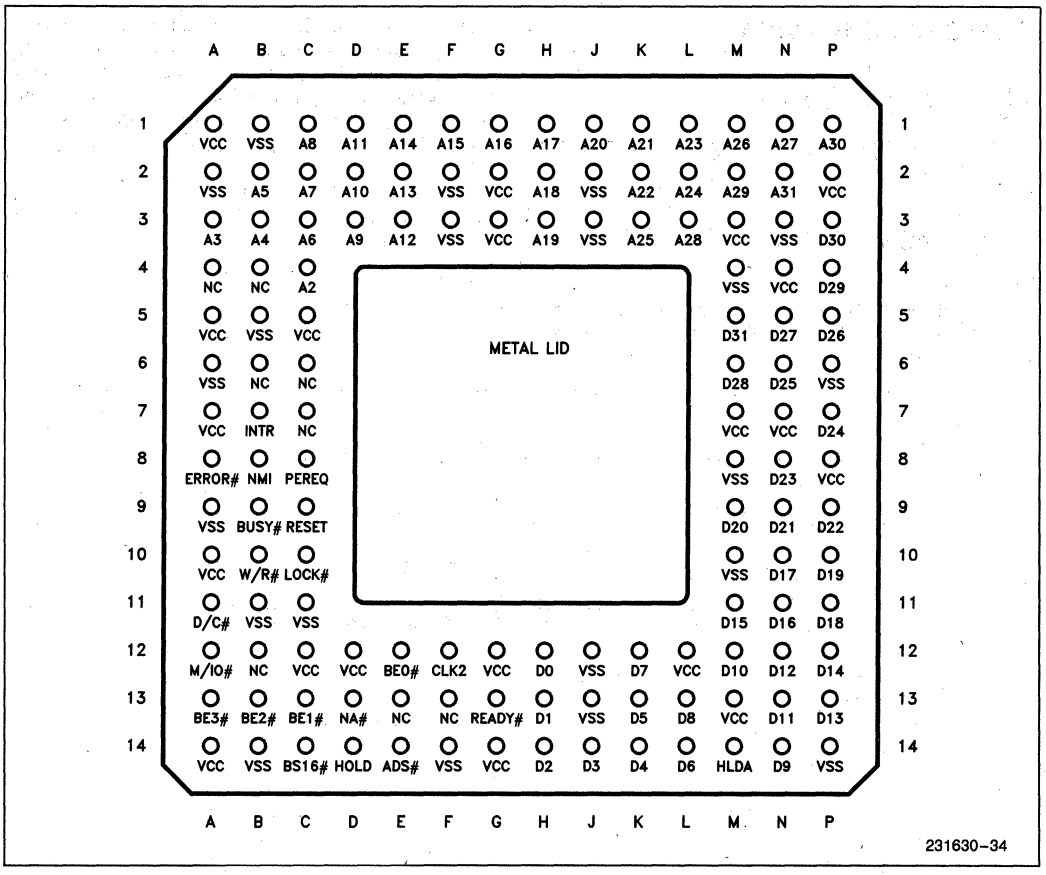


Figure 6-2. 386™ Microprocessor PGA Pinout—View from Pin Side

Table 6-1. 386™ Microprocessor PGA Pinout—Functional Grouping

Pin / Signal	Pin / Signal	Pin / Signal	Pin / Signal
N2 A31	M5 D31	A1 V_{CC}	A2 V_{SS}
P1 A30	P3 D30	A5 V_{CC}	A6 V_{SS}
M2 A29	P4 D29	A7 V_{CC}	A9 V_{SS}
L3 A28	M6 D28	A10 V_{CC}	B1 V_{SS}
N1 A27	N5 D27	A14 V_{CC}	B5 V_{SS}
M1 A26	P5 D26	C5 V_{CC}	B11 V_{SS}
K3 A25	N6 D25	C12 V_{CC}	B14 V_{SS}
L2 A24	P7 D24	D12 V_{CC}	C11 V_{SS}
L1 A23	N8 D23	G2 V_{CC}	F2 V_{SS}
K2 A22	P9 D22	G3 V_{CC}	F3 V_{SS}
K1 A21	N9 D21	G12 V_{CC}	F14 V_{SS}
J1 A20	M9 D20	G14 V_{CC}	J2 V_{SS}
H3 A19	P10 D19	L12 V_{CC}	J3 V_{SS}
H2 A18	P11 D18	M3 V_{CC}	J12 V_{SS}
H1 A17	N10 D17	M7 V_{CC}	J13 V_{SS}
G1 A16	N11 D16	M13 V_{CC}	M4 V_{SS}
F1 A15	M11 D15	N4 V_{CC}	M8 V_{SS}
E1 A14	P12 D14	N7 V_{CC}	M10 V_{SS}
E2 A13	P13 D13	P2 V_{CC}	N3 V_{SS}
E3 A12	N12 D12	P8 V_{CC}	P6 V_{SS}
D1 A11	N13 D11		P14 V_{SS}
D2 A10	M12 D10		
D3 A9	N14 D9	F12 CLK2	A4 N.C.
C1 A8	L13 D8		B4 N.C.
C2 A7	K12 D7	E14 ADS#	B6 N.C.
C3 A6	L14 D6		B12 N.C.
B2 A5	K13 D5	B10 W/R#	C6 N.C.
B3 A4	K14 D4	A11 D/C#	C7 N.C.
A3 A3	J14 D3	A12 M/IO#	E13 N.C.
C4 A2	H14 D2	C10 LOCK#	F13 N.C.
A13 BE3#	H13 D1		
B13 BE2#	H12 D0	D13 NA#	C8 PEREQ
C13 BE1#		C14 BS16#	B9 BUSY#
E12 BE0#		G13 READY#	A8 ERROR#
	D14 HOLD		
C9 RESET	M14 HLDA	B7 INTR	B8 NMI

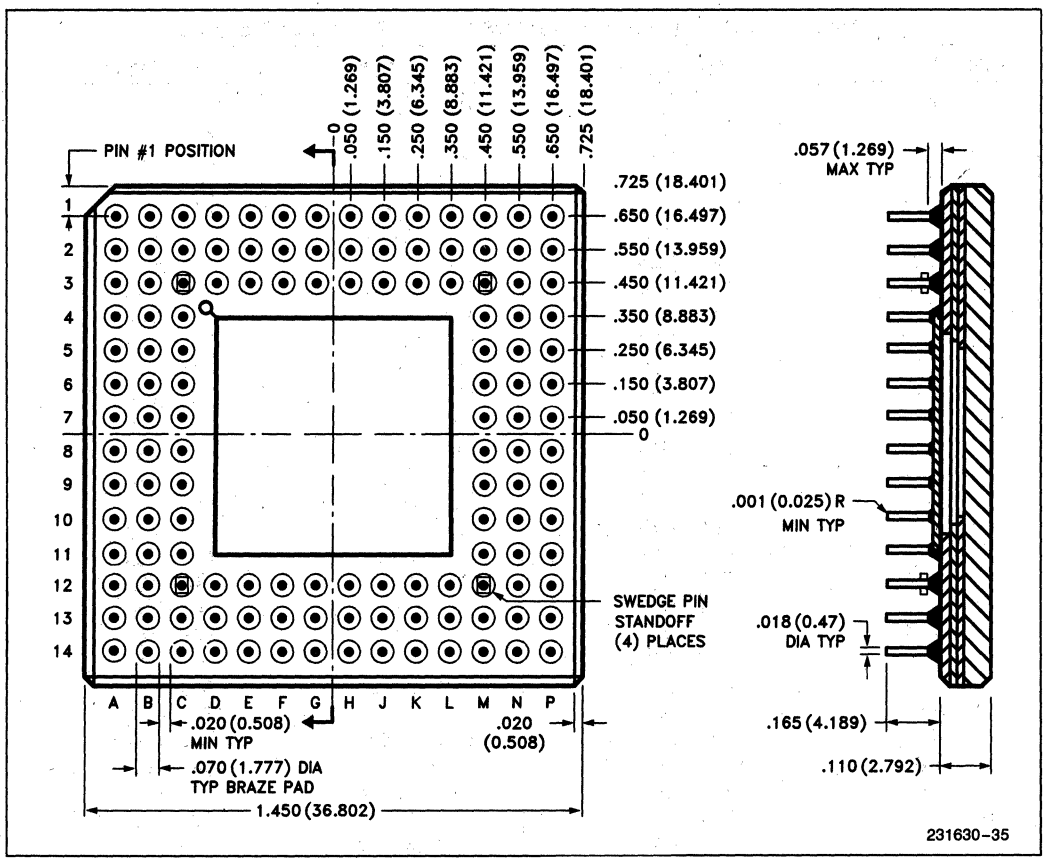


Figure 6-3. 132-Pin Ceramic PGA Package Dimensions

6.3 Package Dimensions and Mounting

The initial 386 Microprocessor package is a 132-pin ceramic pin grid array (PGA). Pins of this package are arranged 0.100 inch (2.54mm) center-to-center, in a 14 x 14 matrix, three rows around.

A wide variety of available sockets allow low insertion force or zero insertion force mountings, and a choice of terminals such as soldertail, surface mount, or wire wrap. Several applicable sockets are listed in Table 6-2.

6.4 PACKAGE THERMAL SPECIFICATION

The 386 Microprocessor is specified for operation when case temperature is within the range of 0°C–85°C. The case temperature may be measured in any environment, to determine whether the 386 Microprocessor is within specified operating range.

The PGA case temperature should be measured at the center of the top surface opposite the pins, as in Figure 6-4.

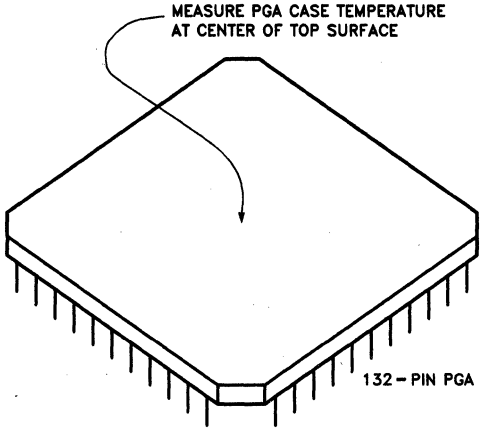


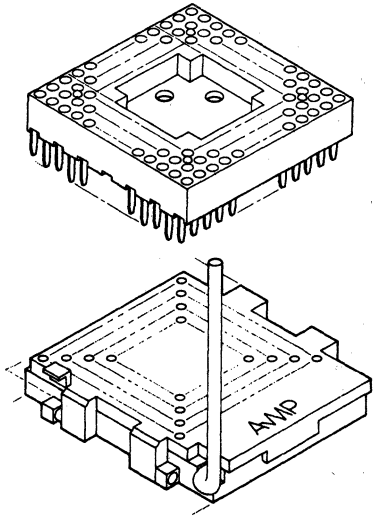
Figure 6-4. Measuring 386™ Microprocessor PGA Case Temperature

Table 6-2. Several Socket Options for 132-Pin PGA

* Low insertion force (LIF) soldertail 55274-1
* Amp tests indicate 50% reduction in insertion force compared to machined sockets

Other socket options
* Zero insertion force (ZIF) soldertail 55583-1
* Zero insertion force (ZIF) Burn-in version 55573-2

Amp Incorporated
(Harrisburg, PA 17105 U.S.A.
Phone 717-564-0100)



231630–45
Cam handle locks in low profile position when substrate is installed (handle UP for open and DOWN for closed positions)

courtesy Amp Incorporated

Table 6-2. Several Socket Options for 132-Pin PGA (Continued)

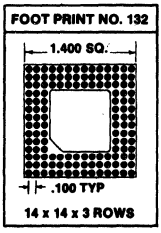
Peel-A-Way™ Mylar and Kapton Socket Terminal Carriers

* Low insertion force surface mount CS132-37TG
* Low insertion force soldertail CS132-01TG
* Low insertion force wire-wrap CS132-02TG (two level) CS132-03TG (three-level)
* Low insertion force press-fit CS132-05TG

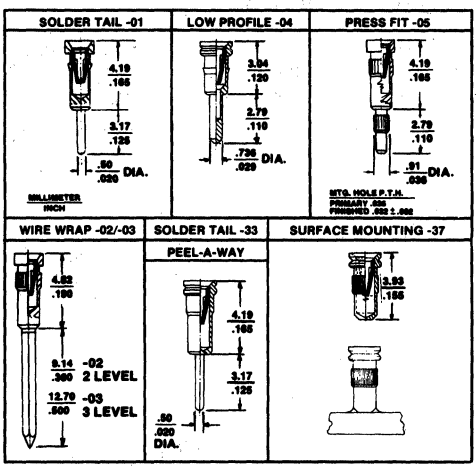
Advanced Interconnections
(5 Division Street
Warwick, RI 02818 U.S.A.
Phone 401-885-0485)

Peel-A-Way Carrier No. 132:
Kapton Carrier is KS132
Mylar Carrier is MS132

Molded Plastic Body KS132 is shown below:



231630-46

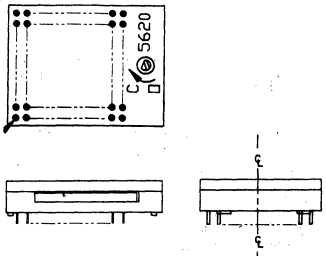


231630-47

courtesy Advanced Interconnections
(Peel-A-Way Terminal Carriers
U.S. Patent No. 4442938)

* Low insertion force socket soldertail (for production use)
  2XX-6576-00-3308 (new style)
  2XX-6003-00-3302 (older style)
* Zero insertion force soldertail (for test and burn-in use)
  2XX-6568-00-3302

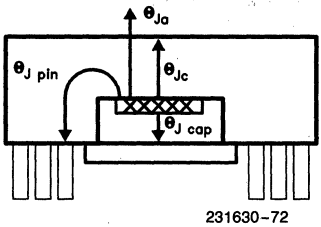
Textool Products
Electronic Products Division/3M
(1410 West Pioneer Drive
Irving, Texas 75601 U.S.A.
Phone 214-259-2676)



courtesy Textool Products/3M 231630-48

Table 6-3. 386™ Microprocessor PGA Package Thermal Characteristics

Parameter	Thermal Resistance — °C/Watt: Airflow — ft./min (m/sec) 0 (0)	50 (0.25)	100 (0.50)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)
θ Junction-to-Case (case measured as Fig. 6-4)	2	2	2	2	2	2	2
θ Case-to-Ambient (no heatsink)	19	18	17	15	12	10	9
θ Case-to-Ambient (with omnidirectional heatsink)	16	15	14	12	9	7	6
θ Case-to-Ambient (with unidirectional heatsink)	15	14	13	11	8	6	5



231630-72

NOTES:
1. Table 6-3 applies to 386™ CPU PGA plugged into socket or soldered directly into board.
2. $\theta_{JA} = \theta_{JC} + \theta_{CA}$.
3. $\theta_{J\text{-}CAP}$ = 4°C/w (approx.)
   $\theta_{J\text{-}PIN}$ = 4°C/w (inner pins) (approx.)
   $\theta_{J\text{-}PIN}$ = 8°C/w (outer pins) (approx.)

7. ELECTRICAL DATA

7.1 INTRODUCTION

The following sections describe recommended electrical connections for the 386 Microprocessor, and its electrical specifications.

7.2 POWER AND GROUNDING

7.2.1 Power Connections

The 386 Microprocessor is implemented in CHMOS III and CHMOS IV technology and has modest power requirements. However, its high clock frequency and 72 output buffers (address, data, control, and HLDA) can cause power surges as multiple output buffers drive new signal levels simultaneously. For clean on-chip power distribution at high frequency, 20 V_{CC} and 21 V_{SS} pins separately feed functional units of the 386 Microprocessor.

Power and ground connections must be made to all external V_{CC} and GND pins of the 386 Microprocessor. On the circuit board, all V_{CC} pins must be connected on a V_{CC} plane. All V_{SS} pins must be likewise connected on a GND plane.

7.2.2 Power Decoupling Recommendations

Liberal decoupling capacitance should be placed near the 386 Microprocessor. The 386 Microprocessor driving its 32-bit parallel address and data buses at high frequencies can cause transient power surges, particularly when driving large capacitive loads.

Low inductance capacitors and interconnects are recommended for best high frequency electrical performance. Inductance can be reduced by shortening circuit board traces between the 386 Microprocessor and decoupling capacitors as much as possible. Capacitors specifically for PGA packages are also commercially available, for the lowest possible inductance.

7.2.3 Resistor Recommendations

The ERROR# and BUSY# inputs have resistor pullups of approximately 20 KΩ built-in to the 386 Microprocessor to keep these signals negated when neither 80287 or 80387 are present in the system (or temporarily removed from its socket). The BS16# input also has an internal pullup resistor of approximately 20 KΩ, and the PEREQ input has an internal pulldown resistor of approximately 20 KΩ.

In typical designs, the external pullup resistors shown in Table 7-1 are recommended. However, a particular design may have reason to adjust the resistor values recommended here, or alter the use of pullup resistors in other ways.

7.2.4 Other Connection Recommendations

For reliable operation, always connect unused inputs to an appropriate signal level. N.C. pins should always remain unconnected.

Particularly when not using interrupts or bus hold, (as when first prototyping, perhaps) prevent any chance of spurious activity by connecting these associated inputs to GND:

Pin	Signal
B7	INTR
B8	NMI
D14	HOLD

If not using address pipelining, pullup D13 NA# to V_{CC}.

If not using 16-bit bus size, pullup C14 BS16# to V_{CC}.

Pullups in the range of 20 KΩ are recommended.

Table 7-1. Recommended Resistor Pullups to V_{CC}

Pin and Signal	Pullup Value	Purpose
E14 ADS#	20 KΩ ±10%	Lightly Pull ADS# Negated During 386™ Microprocessor Hold Acknowledge States
C10 LOCK#	20 KΩ ±10%	Lightly Pull LOCK# Negated During 386™ Microprocessor Hold Acknowledge States

7.3 MAXIMUM RATINGS

Table 7-2. Maximum Ratings

Parameter	386™ CPU 16, 20, 25 MHz Maximum Rating
Storage Temperature	−65°C to +150°C
Case Temperature Under Bias	−65°C to +110°C
Supply Voltage with Respect to V_{SS}	−0.5V to +6.5V
Voltage on Other Pins	−0.5V to V_{CC} + 0.5V

Table 7-2 is a stress rating only, and functional operation at the maximums is not guaranteed. Functional operating conditions are given in **7.4 D.C. Specifications** and **7.5 A.C. Specifications**.

Extended exposure to the Maximum Ratings may affect device reliability. Furthermore, although the 386 Microprocessor contains protective circuitry to resist damage from static electric discharge, always take precautions to avoid high static voltages or electric fields.

7.4 D.C. SPECIFICATIONS

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to 85°C

Table 7-3. 386™ Microprocessor D.C. Characteristics

Symbol	Parameter	386™ CPU 16 MHz, 20 MHz, 25 MHz		Unit	Test Conditions
		Min	Max		
V_{IL}	Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{ILC}	CLK2 Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IHC}	CLK2 Input High Voltage 16 MHz and 20 MHz 25 MHz	 V_{CC} − 0.8 3.7	 V_{CC} + 0.3 V_{CC} + 0.3	 V V	
V_{OL}	Output Low Voltage I_{OL} = 4 mA: A2–A31, D0–D31 I_{OL} = 5 mA: BE0#–BE3#, W/R#, D/C#, M/IO#, LOCK#, ADS#, HLDA	 	 0.45 0.45	 V V	
V_{OH}	Output High Voltage I_{OH} = 1 mA: A2–A31, D0–D31 I_{OH} = 0.9 mA: BE0#–BE3#, W/R#, D/C#, M/IO#, LOCK#, ADS#, HLDA	 2.4 2.4	 	 V V	
I_{LI}	Input Leakage Current (For All Pins except BS16#, PEREQ, BUSY#, and ERROR#)		±15	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{IH}	Input Leakage Current (PEREQ Pin)		200	μA	V_{IH} = 2.4V (Note 2)
I_{IL}	Input Leakage Current (BS16#, BUSY#, and ERROR# Pins)		−400	μA	V_{IL} = 0.45 (Note 3)
I_{LO}	Output Leakage Current		±15	μA	0.45V ≤ V_{OUT} ≤ V_{CC}
I_{CC}	Supply Current CLK2 = 32 MHz: with 16 MHz 386™ CPU CLK2 = 40 MHz: with 20 MHz 386™ CPU CLK2 = 50 MHz: with 25 MHz 386™ CPU	 	 450 500 550	 mA mA mA	 I_{CC} Typ. = 370 mA I_{CC} Typ. = 460 mA I_{CC} Typ. = 580 mA
C_{IN}	Output Capacitance		10	pF	F_C = 1 MHz (Note 4)
C_{OUT}	Output or I/O Capacitance		12	pF	F_C = 1 MHz (Note 4)
C_{CLK}	CLK2 Capacitance		20	pF	F_C = 1 MHz (Note 4)

NOTES:
1. The min value, −0.3, is not 100% tested.
2. PEREQ input has an internal pulldown resistor.
3. BS16#, BUSY# and ERROR# inputs each have an internal pullup resistor.
4. Not 100% tested.

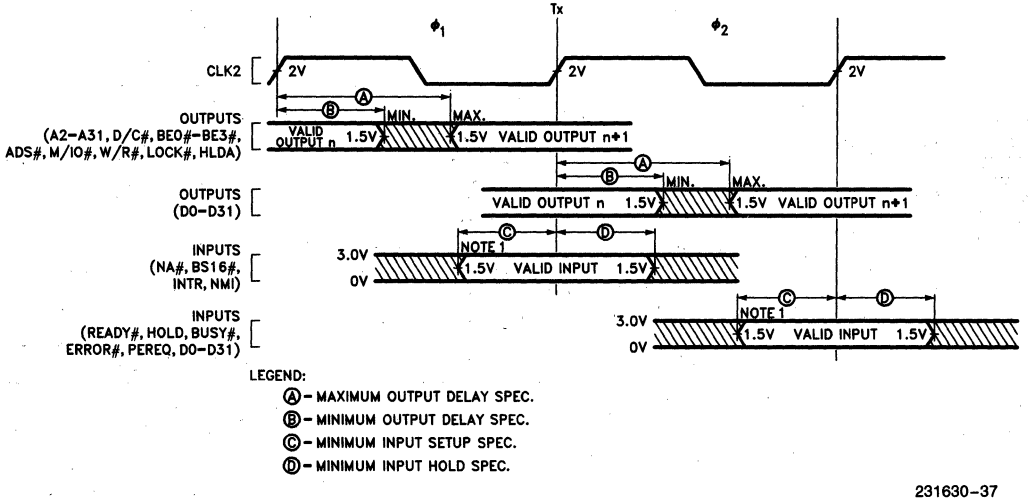
7.5 A.C. SPECIFICATIONS

7.5.1 A.C. Spec Definitions

The A.C. specifications, given in Tables 7-4, 7-5, and 7-6, consist of output delays, input setup requirements and input hold requirements. All A.C. specifications are relative to the CLK2 rising edge crossing the 2.0V level.

A.C. spec measurement is defined by Figure 7-1. Inputs must be driven to the voltage levels indicated by Figure 7-1 when A.C. specifications are measured. 386 Microprocessor output delays are specified with minimum and maximum limits, measured as shown. The minimum 386 Microprocessor delay times are hold times provided to external circuitry. 386 Microprocessor input setup and hold times are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct 386 Microprocessor operation.

Outputs NA#, W/R#, D/C#, M/IO#, LOCK#, BE0#-BE3#, A2-A31 and HLDA only change at the beginning of phase one. D0-D31 (write cycles) only change at the beginning of phase two. The READY#, HOLD, BUSY#, ERROR#, PEREQ and D0-D31 (read cycles) inputs are sampled at the beginning of phase one. The NA#, BS16#, INTR and NMI inputs are sampled at the beginning of phase two.



NOTES:
1. Input waveforms have tr ≤ 2.0 ns from 0.8V to 2.0V.
2. See section 7.5.6 for typical output rise time versus load capacitance.

Figure 7-1. Drive Levels and Measurement Points for A.C. Specifications

7.5.2 A.C. Specification Tables

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C

Table 7-4. 25 MHz 386™ Microprocessor A.C. Characteristics

Symbol	Parameter	25 MHz 386™ CPU		Unit	Ref. Fig.	Notes
		Min	Max			
	Operating Frequency	4	25	MHz		Half of CLK2 Frequency
t1	CLK2 Period	20	125	ns	7-3	
t2a	CLK2 High Time	7		ns	7-3	at 2V
t2b	CLK2 High Time	4		ns	7-3	at 3.7V
t3a	CLK2 Low Time	7		ns	7-3	at 2V
t3b	CLK2 Low Time	5		ns	7-3	at 0.8V
t4	CLK2 Fall Time		7	ns	7-3	3.7V to 0.8V
t5	CLK2 Rise Time		7	ns	7-3	0.8V to 3.7V
t6*	A2–A31 Valid Delay	4	21*	ns	7-5	C_L = 50 pF
t7	A2–A31 Float Delay	4	30	ns	7-6	(Note 1)
t8	BE0#–BE3# Valid Delay	4	24	ns	7-5	C_L = 50 pF
t8a	LOCK# Valid Delay	4	21	ns	7-5	C_L = 50 pF
t9	BE0#–BE3#, LOCK# Float Delay	4	30	ns	7-6	(Note 1)
t10*	W/R#, M/IO#, D/C#, ADS# Valid Delay	4	21*	ns	7-5	C_L = 50 pF
t11	W/R#, M/IO#, D/C#, ADS# Float Delay	4	30	ns	7-6	(Note 1)
t12*	D0–D31 Write Data Valid Delay	7	27*	ns	7-5A	C_L = 50 pF
t12a	D0–D31 Write Data Hold Time	2			7-5B	C_L = 50 pF
t13	D0–D31 Float Delay	4	22	ns	7-6	(Note 1)
t14	HLDA Valid Delay	4	22	ns	7-6	C_L = 50 pF
t15	NA# Setup Time	7		ns	7-4	
t16	NA# Hold Time	3		ns	7-4	
t17	BS16# Setup Time	7		ns	7-4	
t18	BS16# Hold Time	3		ns	7-4	
t19*	READY# Setup Time	9*		ns	7-4	
t20	READY# Hold Time	4		ns	7-4	

7.5.2 A.C. Specification Tables (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C

Table 7-4. 25 MHz 386™ Microprocessor A.C. Characteristics (Continued)

Symbol	Parameter	25 MHz 386™ CPU		Unit	Ref. Fig.	Notes
		Min	Max			
t21*	D0–D31 Read Setup Time	7*		ns	7-4	
t22	D0–D31 Read Hold Time	5		ns	7-4	
t23	HOLD Setup Time	16		ns	7-4	
t24	HOLD Hold Time	3		ns	7-4	
t25	RESET Setup Time	10		ns	7-7	
t26	RESET Hold Time	3		ns	7-7	
t27	NMI, INTR Setup Time	6		ns	7-4	(Note 2)
t28	NMI, INTR Hold Time	6		ns	7-4	(Note 2)
t29	PEREQ, ERROR#, BUSY# Setup Time	6		ns	7-4	(Note 2)
t30	PEREQ, ERROR#, BUSY# Hold Time	5		ns	7-4	(Notes 2, 3)

NOTES:

Timing Specifications marked with a "" are tested to guarantee reliable operation with the 25 MHz 82385 (See Note 4 for more information).

1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3.

	Symbol	Parameter	Min
T_C = 0°C	t30	PEREQ, ERROR#, BUSY# Hold Time	4
T_C = +85°C	t30	PEREQ, ERROR#, BUSY# Hold Time	5

4. The following 386™ Microprocessor timing values are guaranteed for 82385 operation:
   t6 = (2CLK2 − t13B − LOGIC − CL), (2CLK2 − t7a − CL)
   t8a = (2CLK2 − t7B − CL)
   t10 = (2CLK2 − t9B − CL)
   t12min = (CLK2 − PD − t25C), (CLK2 − OD − t43B)
   t19 = (2CLK2 − t18 − LOGIC)
   t21 = (2CLK2 − t25A − OE − PD), (2CLK2 − t25B − OE − PD)
   CLK2 = 20 ns (period of 386™ CPU CLK2 input at 25 MHz)
   LOGIC = 10 ns (D-PAL)
   CL = 2 ns (Capacitive load derating for an 386™ Microprocessor output)
   CE = 9 ns (Output enable/disable time for an AS245)
   PD = 5.5 ns (Propagation delay for an AS08 gate)
   OD = 10 ns (Output disable for an AS646)
   82385 Parameters:
   t7a = A2–A31, BE0#–BE3# Setup Time
   t7b = LOCK# Setup Time
   t9B = ADS#, W/R# Setup Time
   t13B = LBA# Setup Time
   t18 = READYO# Valid Delay
   t25A = COEA#, COEB# Valid Delay
   t25B = COEA#, COEB# Valid Delay
   t25C = COEA#, COEB# Rising Delay
   t43B = DOE# Rising Delay

7.5.2 A.C. Specification Tables (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C

Table 7-5. 20 MHz 386™ Microprocessor A.C. Characteristics

Symbol	Parameter	20 MHz 386™ CPU		Unit	Ref. Fig.	Notes
		Min	Max			
	Operating Frequency	4	20	MHz		Half of CLK2 Frequency
t_1	CLK2 Period	25	125	ns	7-3	
t_{2a}	CLK2 High Time	8		ns	7-3	at 2V
t_{2b}	CLK2 High Time	5		ns	7-3	at (V_{CC} − 0.8V)
t_{3a}	CLK2 Low Time	8		ns	7-3	at 2V
t_{3b}	CLK2 Low Time	6		ns	7-3	at 0.8V
t_4	CLK2 Fall Time		8	ns	7-3	(V_{CC} − 0.8V) to 0.8V
t_5	CLK2 Rise Time		8	ns	7-3	0.8V to (V_{CC} − 0.8V)
t_6	A2–A31 Valid Delay	4	30	ns	7-5	C_L = 120 pF
t_7	A2–A31 Float Delay	4	32	ns	7-6	(Note 1)
t_8	BE0#–BE3#, LOCK# Valid Delay	4	30	ns	7-5	C_L = 75 pF
t_9	BE0#–BE3#, LOCK# Float Delay	4	32	ns	7-6	(Note 1)
t_{10}	W/R#, M/IO#, D/C#, ADS# Valid Delay	6	28	ns	7-5	C_L = 75 pF
t_{11}	W/R#, M/IO#, D/C#, ADS# Float Delay	6	30	ns	7-6	(Note 1)
t_{12}	D0–D31 Write Data Valid Delay	4	38	ns	7-5	C_L = 120 pF
t_{13}	D0–D31 Float Delay	4	27	ns	7-6	(Note 1)
t_{14}	HLDA Valid Delay	6	28	ns	7-6	C_L = 75 pF
t_{15}	NA# Setup Time	9		ns	7-4	
t_{16}	NA# Hold Time	14		ns	7-4	
t_{17}	BS16# Setup Time	13		ns	7-4	
t_{18}	BS16# Hold Time	21		ns	7-4	
t_{19}	READY# Setup Time	12		ns	7-4	
t_{20}	READY# Hold Time	4		ns	7-4	
t_{21}	D0–D31 Read Setup Time	11		ns	7-4	
t_{22}	D0–D31 Read Hold Time	6		ns	7-4	
t_{23}	HOLD Setup Time	17		ns	7-4	
t_{24}	HOLD Hold Time	5		ns	7-4	
t_{25}	RESET Setup Time	12		ns	7-7	

7.5.2 A.C. Specification Tables (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C

Table 7-5. 20 MHz 386™ Microprocessor A.C. Characteristics (Continued)

Symbol	Parameter	20 MHz 386™ CPU		Unit	Ref. Fig.	Notes
		Min	Max			
t_{26}	RESET Hold Time	4		ns	7-7	
t_{27}	NMI, INTR Setup Time	16		ns	7-4	(Note 2)
t_{28}	NMI, INTR Hold Time	16		ns	7-4	(Note 2)
t_{29}	PEREQ, ERROR#, BUSY# Setup Time	14		ns	7-4	(Note 2)
t_{30}	PEREQ, ERROR#, BUSY# Hold Time	5		ns	7-4	(Note 2)

NOTES:
1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.

Table 7-6. 16 MHz 386™ Microprocessor A.C. Characteristics

Symbol	Parameter	16 MHz 386™ CPU		Unit	Ref. Fig.	Notes
		Min	Max			
	Operating Frequency	4	16	MHz	—	Half of CLK2 Frequency
t_1	CLK2 Period	31	125	ns	7-3	
t_{2a}	CLK2 High Time	9		ns	7-3	at 2V
t_{2b}	CLK2 High Time	5		ns	7-3	at (V_{CC} − 0.8V)
t_{3a}	CLK2 Low Time	9		ns	7-3	at 2V
t_{3b}	CLK2 Low Time	7		ns	7-3	at 0.8V
t_4	CLK2 Fall Time		8	ns	7-3	(V_{CC} − 0.8V) to 0.8V
t_5	CLK2 Rise Time		8	ns	7-3	0.8V to (V_{CC} − 0.8V)
t_6	A2–A31 Valid Delay	4	36	ns	7-5	C_L = 120 pF
t_7	A2–A31 Float Delay	4	40	ns	7-6	(Note 1)
t_8	BE0#–BE3#, LOCK# Valid Delay	4	36	ns	7-5	C_L = 75 pF
t_9	BE0#–BE3#, LOCK# Float Delay	4	40	ns	7-6	(Note 1)
t_{10}	W/R#, M/IO#, D/C#, ADS# Valid Delay	6	33	ns	7-5	C_L = 75 pF
t_{11}	W/R#, M/IO#, D/C#, ADS# Float Delay	6	35	ns	7-6	(Note 1)
t_{12}	D0–D31 Write Data Valid Delay	4	48	ns	7-5	C_L = 120 pF

7.5.2 A.C. Specification Tables (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C

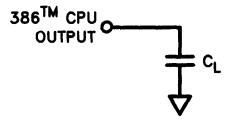
Table 7-6. 16 MHz 386™ Microprocessor A.C. Characteristics (Continued)

Symbol	Parameter	16 MHz 386™ CPU		Unit	Ref. Fig.	Notes
		Min	Max			
t_{13}	D0–D31 Float Delay	4	35	ns	7-6	(Note 1)
t_{14}	HLDA Valid Delay	6	33	ns	7-6	C_L = 75 pF
t_{15}	NA# Setup Time	11		ns	7-4	
t_{16}	NA# Hold Time	14		ns	7-4	
t_{17}	BS16# Setup Time	13		ns	7-4	
t_{18}	BS16# Hold Time	21		ns	7-4	
t_{19}	READY# Setup Time	21		ns	7-4	
t_{20}	READY# Hold Time	4		ns	7-4	
t_{21}	D0–D31 Read Setup Time	11		ns	7-4	
t_{22}	D0–D31 Read Hold Time	6		ns	7-4	
t_{23}	HOLD Setup Time	26		ns	7-4	
t_{24}	HOLD Hold Time	5		ns	7-4	
t_{25}	RESET Setup Time	13		ns	7-7	
t_{26}	RESET Hold Time	4		ns	7-7	
t_{27}	NMI, INTR Setup Time	16		ns	7-4	(Note 2)
t_{28}	NMI, INTR Hold Time	16		ns	7-4	(Note 2)
t_{29}	PEREQ, ERROR#, BUSY# Setup Time	16		ns	7-4	(Note 2)
t_{30}	PEREQ, ERROR#, BUSY# Hold Time	5		ns	7-4	(Note 2)

NOTES:

1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.

7.5.3 A.C. Test Loads



C_L = 120 pF on A2–A31, D0–D31
C_L = 75 pF on BE0#–BE3#, W/R#, M/IO#, D/C#, ADS#, LOCK#, HLDA
C_L includes all parasitic capacitances.

Figure 7-2. A.C. Test Load

7.5.4 A.C. Timing Waveforms

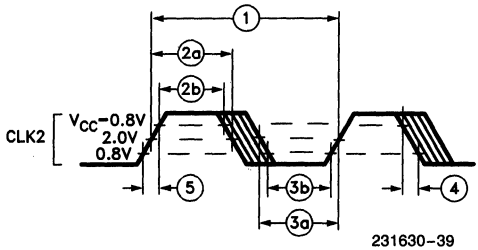


Figure 7-3. CLK2 Timing

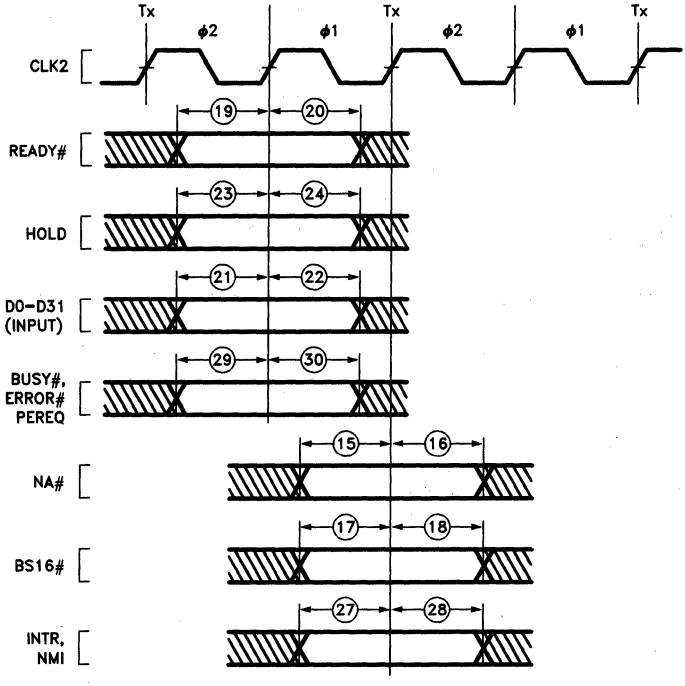


Figure 7-4. Input Setup and Hold Timing

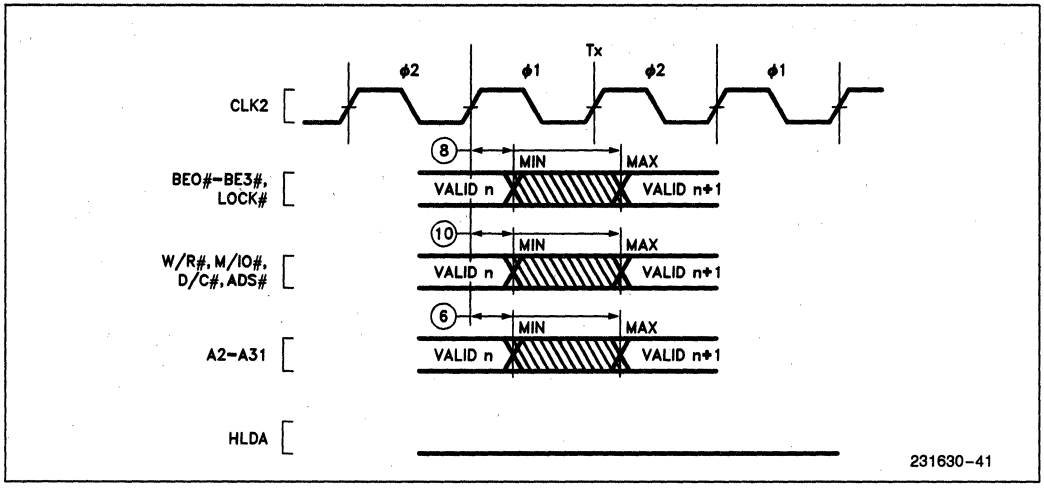


Figure 7-5. Output Valid Delay Timing

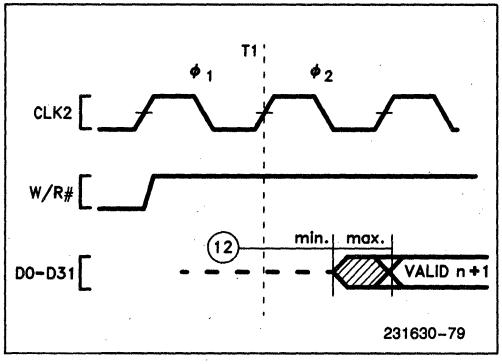


Figure 7-5a. Write Data Valid Delay Timing

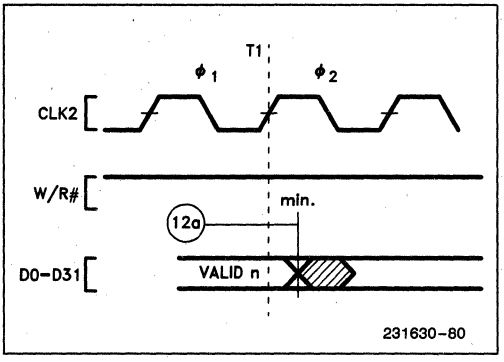
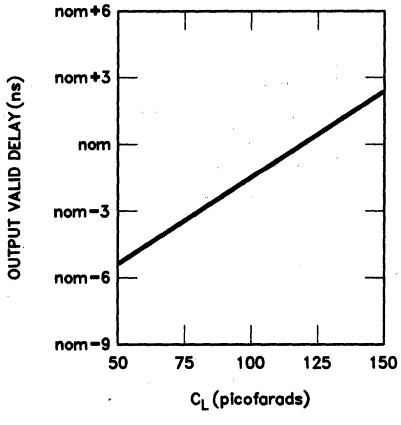


Figure 7-5b. Write Data Hold Timing

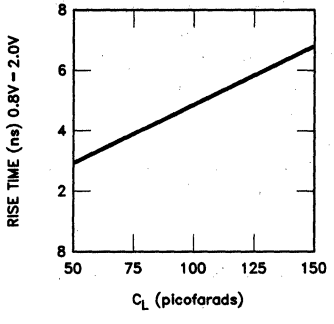
7.5.5 Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature



231630-77

NOTE:
This graph will not be linear outside of the C_L range shown.

7.5.6 Typical Output Rise Time Versus Load Capacitance at Maximum Operating Temperature



231630-78

NOTE:
This graph will not be linear outside of the C_L range shown.

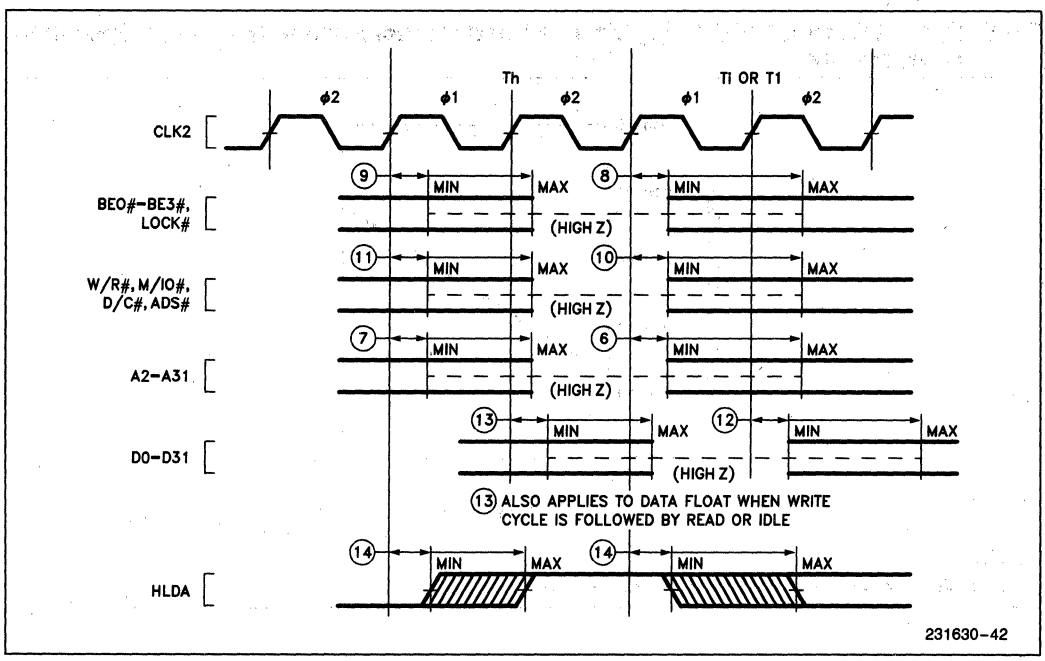
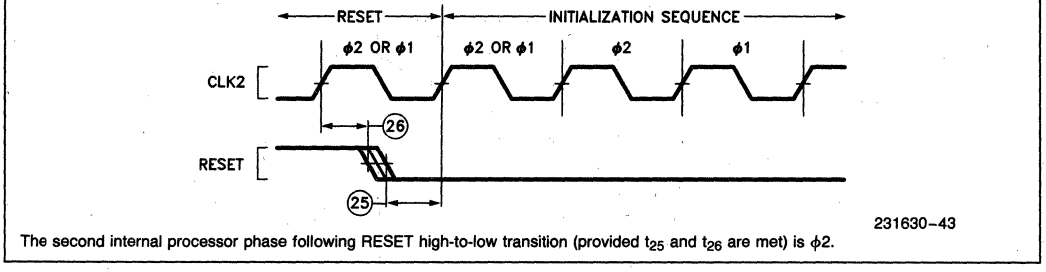


Figure 7-6. Output Float Delay and HLDA Valid Delay Timing



The second internal processor phase following RESET high-to-low transition (provided t_{25} and t_{26} are met) is $\phi 2$.

Figure 7-7. RESET Setup and Hold Timing, and Internal Phase

7.6 DESIGNING FOR ICE™-386 USE

The 386 Microprocessor in-circuit emulator product is ICE-386. Because of the high operating frequency of 386 Microprocessor systems and ICE-386, there is no cable separating the ICE-386 probe module from the target system. The ICE-386 probe module has several electrical and mechanical characteristics that should be taken into consideration when designing the hardware.

Capacitive loading: ICE-386 adds up to 25 pF to each line.

Drive requirement: ICE-386 adds one standard TTL load on the CLK2 line, up to one advanced low-power Schottky TTL load per control signal line, and one advanced low-power Schottky TTL load per address, byte enable, and data line. These loads are within the probe module and are driven by the probe's 386 Microprocessor, which has standard drive and loading capability listed in Tables 7-3 and 7-4.

Power requirement: For noise immunity the ICE-386 probe is powered by the user system. The high-speed probe circuitry draws up to 0.7A plus the maximum 386 Microprocessor I_{CC} from the user 386 Microprocessor socket.

386 Microprocessor location and orientation: The ICE-386 Processor Module (PM), and the Optional Isolation Board (OIB) used for extra electrical buffering of the ICE initially, require clearance as illustrated in Figures 7-8 and 7-9, respectively. Figures 7-8 and 7-9 also illustrate the via holes in these modules for recommended orientation of a screw-actuated ZIF socket. Figure 7-10 illustrates the recommended orientation for a lever-actuated ZIF socket.

READY# drive: The ICE-386 system may be able to clear a user system READY# hang if the user's READY# driver is implemented with an open-collector or tri-state device.

Optional Interface Board (OIB) and CLK2 speed reduction: When the ICE-386 processor probe is first attached to an unverified user system, the OIB helps ICE-386 function in user systems with bus faults (shorted signals, etc.). After electrical verification it may be removed. Only when the OIB is installed, the user system must have a reduced CLK2 frequency of 16 MHz maximum.

Cache coherence: ICE-386 loads user memory by performing 386 Microprocessor write cycles. Note that if the user system is not designed to update or invalidate its cache (if it has a cache) upon processor writes to memory, the cache could contain stale instruction code and/or data. For best use of ICE-386, the user should consider designing the cache (if any) to update itself automatically when processor writes occur, or find another method of maintaining cache data coherence with main user memory.

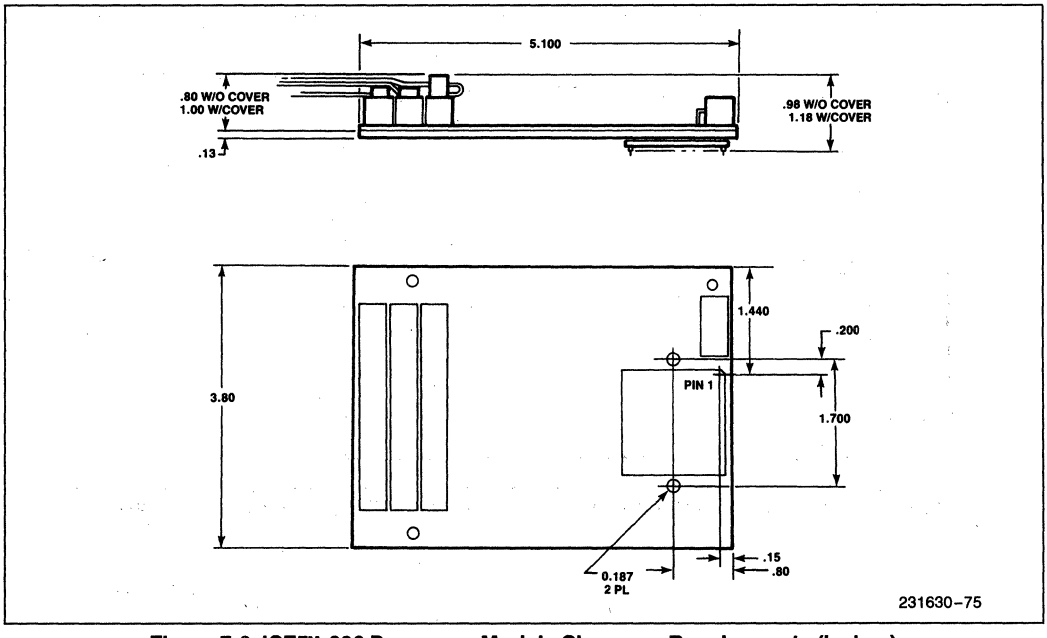


Figure 7-8. ICE™-386 Processor Module Clearance Requirements (inches)

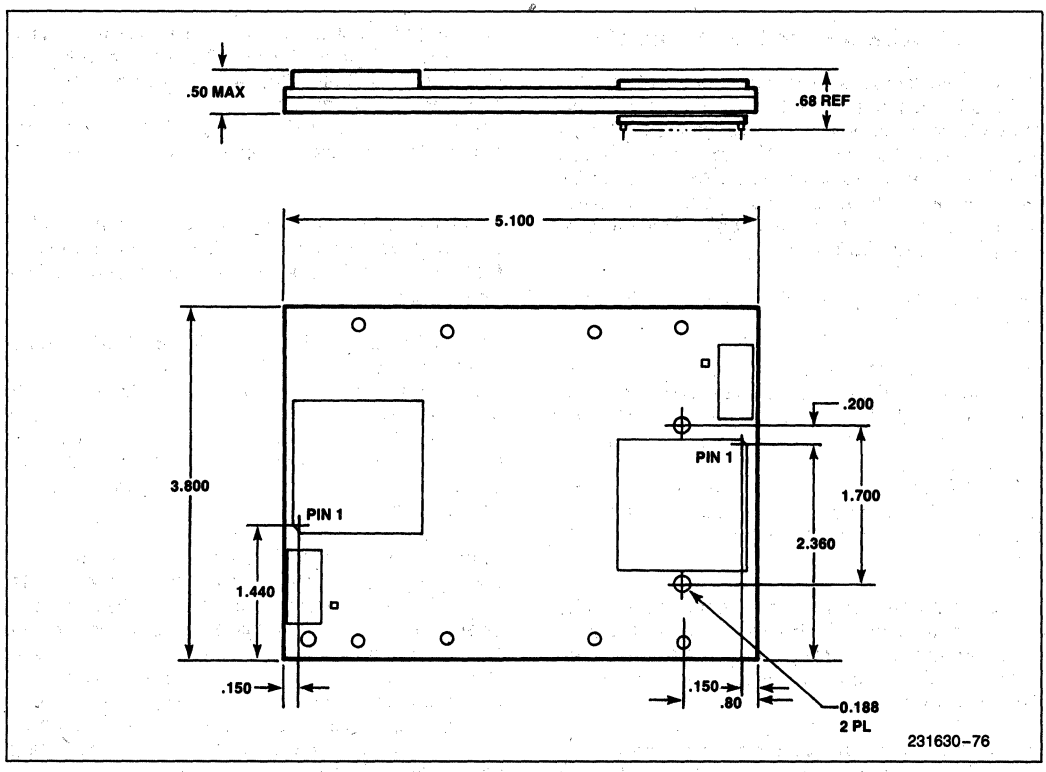


Figure 7-9. ICE™-386 Optional Interface Module Clearance Requirements (inches)

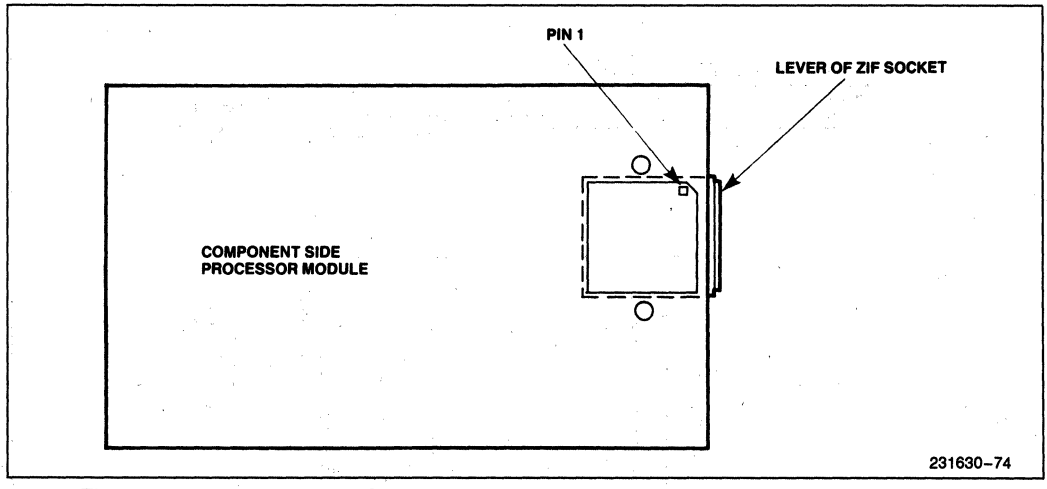


Figure 7-10. Recommended Orientation of Lever-Actuated ZIF Socket for ICE™-386 Use

8. INSTRUCTION SET

This section describes the 386 Microprocessor instruction set. A table lists all instructions along with instruction encoding diagrams and clock counts. Further details of the instruction encoding are then provided in the following sections, which completely describe the encoding structure and the definition of all fields occurring within 386 Microprocessor instructions.

8.1 386™ MICROPROCESSOR INSTRUCTION ENCODING AND CLOCK COUNT SUMMARY

To calculate elapsed time for an instruction, multiply the instruction clock count, as listed in Table 8-1 below, by the processor clock period (e.g. 62.5 ns for a 16 MHz 386 Microprocessor, 50 ns for a 20 MHz 386 Microprocessor and 40 ns for a 25 MHz 386 Microprocessor).

For more detailed information on the encodings of instructions refer to section 8.2 Instruction Encodings. Section 8.2 explains the general structure of instruction encodings, and defines exactly the encodings of all fields contained within the instruction.

Instruction Clock Count Assumptions

1. The instruction has been prefetched, decoded, and is ready for execution.
2. Bus cycles do not require wait states.
3. There are no local bus HOLD requests delaying processor access to the bus.
4. No exceptions are detected during instruction execution.
5. If an effective address is calculated, it does not use two general register components. One register, scaling and displacement can be used within the clock counts shown. However, if the effective address calculation uses two general register components, add 1 clock to the clock count shown.

Instruction Clock Count Notation

1. If two clock counts are given, the smaller refers to a register operand and the larger refers to a memory operand.
2. n = number of times repeated.
3. m = number of components in the next instruction executed, where the entire displacement (if any) counts as one component, the entire immediate data (if any) counts as one component, and each of the **other** bytes of the instruction and prefix(es) each count as one component.

Table 8-1. 386™ Microprocessor Instruction Set Clock Count Summary

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
GENERAL DATA TRANSFER					
MOV = Move:					
Register to Register/Memory	1000100w mod reg r/m	2/2	2/2	b	h
Register/Memory to Register	1000101w mod reg r/m	2/4	2/4	b	h
Immediate to Register/Memory	1100011w mod 000 r/m immediate data	2/2	2/2	b	h
Immediate to Register (short form)	1011 w reg immediate data	2	2		
Memory to Accumulator (short form)	1010000w full displacement	4	4	b	h
Accumulator to Memory (short form)	1010001w full displacement	2	2	b	h
Register Memory to Segment Register	10001110 mod sreg3 r/m	2/5	18/19	b	h, i, j
Segment Register to Register/Memory	10001100 mod sreg3 r/m	2/2	2/2	b	h
MOVSX = Move With Sign Extension					
Register From Register/Memory	00001111 1011111w mod reg r/m	3/6	3/6	b	h
MOVZX = Move With Zero Extension					
Register From Register/Memory	00001111 1011011w mod reg r/m	3/6	3/6	b	h
PUSH = Push:					
Register/Memory	11111111 mod 110 r/m	5	5	b	h
Register (short form)	01010 reg	2	2	b	h
Segment Register (ES, CS, SS or DS)	000sreg2110	2	2	b	h
Segment Register (FS or GS)	00001111 10sreg3000	2	2	b	h
Immediate	011010s0 immediate data	2	2	b	h
PUSHA = Push All	01100000	18	18	b	h
POP = Pop					
Register/Memory	10001111 mod 000 r/m	5	5	b	h
Register (short form)	01011 reg	4	4	b	h
Segment Register (ES, SS or DS)	000sreg2111	7	21	b	h, i, j
Segment Register (FS or GS)	00001111 10sreg3001	7	21	b	h, i, j
POPA = Pop All	01100001	24	24	b	h
XCHG = Exchange					
Register/Memory With Register	1000011w mod reg r/m	3/5	3/5	b, f	f, h
Register With Accumulator (short form)	10010 reg	3	3		
IN = Input from:	**Clk Count Virtual 8086 Mode**				
Fixed Port	1110010w port number †26	12	6*/26**		m
Variable Port	1110110w †27	13	7*/27**		m
OUT = Output to:					
Fixed Port	1110011w port number †24	10	4*/24**		m
Variable Port	1110111w †25	11	5*/25**		m
LEA = Load EA to Register	10001101 mod reg r/m	2	2		

* If CPL ≤ IOPL ** If CPL > IOPL

Table 8-1. 386™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
SEGMENT CONTROL					
LDS = Load Pointer to DS	11000101 | mod reg r/m	7	22	b	h, i, j
LES = Load Pointer to ES	11000100 | mod reg r/m	7	22	b	h, i, j
LFS = Load Pointer to FS	00001111 | 10110100 | mod reg r/m	7	25	b	h, i, j
LGS = Load Pointer to GS	00001111 | 10110101 | mod reg r/m	7	25	b	h, i, j
LSS = Load Pointer to SS	00001111 | 10110010 | mod reg r/m	7	22	b	h, i, j
FLAG CONTROL					
CLC = Clear Carry Flag	11111000	2	2		
CLD = Clear Direction Flag	11111100	2	2		
CLI = Clear Interrupt Enable Flag	11111010	8	8		m
CLTS = Clear Task Switched Flag	00001111 | 00000110	6	6	c	l
CMC = Complement Carry Flag	11110101	2	2		
LAHF = Load AH into Flag	10011111	2	2		
POPF = Pop Flags	10011101	5	5	b	h, n
PUSHF = Push Flags	10011100	4	4	b	h
SAHF = Store AH into Flags	10011110	3	3		
STC = Set Carry Flag	11111001	2	2		
STD = Set Direction Flag	11111101	2	2		
STI = Set Interrupt Enable Flag	11111011	8	8		m
ARITHMETIC					
ADD = Add					
Register to Register	000000dw | mod reg r/m	2	2		
Register to Memory	0000000w | mod reg r/m	7	7	b	h
Memory to Register	0000001w | mod reg r/m	6	6	b	h
Immediate to Register/Memory	100000sw | mod000 r/m | immediate data	2/7	2/7	b	h
Immediate to Accumulator (short form)	0000010w | immediate data	2	2		
ADC = Add With Carry					
Register to Register	000100dw | mod reg r/m	2	2		
Register to Memory	0001000w | mod reg r/m	7	7	b	h
Memory to Register	0001001w | mod reg r/m	6	6	b	h
Immediate to Register/Memory	100000sw | mod010 r/m | immediate data	2/7	2/7	b	h
Immediate to Accumulator (short form)	0001010w | immediate data	2	2		
INC = Increment					
Register/Memory	1111111w | mod000 r/m	2/6	2/6	b	h
Register (short form)	01000 reg	2	2		
SUB = Subtract					
Register from Register	001010dw | mod reg r/m	2	2		

Table 8-1. 386™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
ARITHMETIC (Continued)					
Register from Memory	0010100w | mod reg r/m	7	7	b	h
Memory from Register	0010101w | mod reg r/m	6	6	b	h
Immediate from Register/Memory	100000sw | mod101 r/m | immediate data	2/7	2/7	b	h
Immediate from Accumulator (short form)	0010110w | immediate data	2	2		
SBB = Subtract with Borrow					
Register from Register	000110dw | mod reg r/m	2	2		
Register from Memory	0001100w | mod reg r/m	7	7	b	h
Memory from Register	0001101w | mod reg r/m	6	6	b	h
Immediate from Register/Memory	100000sw | mod011 r/m | immediate data	2/7	2/7	b	h
Immediate from Accumulator (short form)	0001110w | immediate data	2	2		
DEC = Decrement					
Register/Memory	1111111w | reg001 r/m	2/6	2/6	b	h
Register (short form)	01001 reg	2	2		
CMP = Compare					
Register with Register	001110dw | mod reg r/m	2	2		
Memory with Register	0011100w | mod reg r/m	5	5	b	h
Register with Memory	0011101w | mod reg r/m	6	6	b	h
Immediate with Register/Memory	100000sw | mod111 r/m | immediate data	2/5	2/5	b	h
Immediate with Accumulator (short form)	0011110w | immediate data	2	2		
NEG = Change Sign	1111011w | mod011 r/m	2/6	2/6	b	h
AAA = ASCII Adjust for Add	00110111	4	4		
AAS = ASCII Adjust for Subtract	00111111	4	4		
DAA = Decimal Adjust for Add	00100111	4	4		
DAS = Decimal Adjust for Subtract	00101111	4	4		
MUL = Multiply (unsigned)					
Accumulator with Register/Memory	1111011w | mod100 r/m				
Multiplier-Byte		12-17/15-20	12-17/15-20	b, d	d, h
-Word		12-25/15-28	12-25/15-28	b, d	d, h
-Doubleword		12-41/15-44	12-41/15-44	b, d	d, h
IMUL = Integer Multiply (signed)					
Accumulator with Register/Memory	1111011w | mod101 r/m				
Multiplier-Byte		12-17/15-20	12-17/15-20	b, d	d, h
-Word		12-25/15-28	12-25/15-28	b, d	d, h
-Doubleword		12-41/15-44	12-41/15-44	b, d	d, h
Register with Register/Memory	00001111 | 10101111 | mod reg r/m				
Multiplier-Byte		12-17/15-20	12-17/15-20	b, d	d, h
-Word		12-25/15-28	12-25/15-28	b, d	d, h
-Doubleword		12-41/15-44	12-41/15-44	b, d	d, h
Register/Memory with Immediate to Register	011010s1 | mod reg r/m | immediate data				
-Word		13-26/14-27	13-26/14-27	b, d	d, h
-Doubleword		13-42/14-43	13-42/14-43	b, d	d, h

Table 8-1. 386™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
ARITHMETIC (Continued)					
DIV = Divide (Unsigned)					
Accumulator by Register/Memory	1111011w mod110 r/m				
Divisor—Byte		14/17	14/17	b,e	e,h
—Word		22/25	22/25	b,e	e,h
—Doubleword		38/41	38/41	b,e	e,h
IDIV = Integer Divide (Signed)					
Accumulator By Register/Memory	1111011w mod111 r/m				
Divisor—Byte		19/22	19/22	b,e	e,h
—Word		27/30	27/30	b,e	e,h
—Doubleword		43/46	43/46	b,e	e,h
AAD = ASCII Adjust for Divide	11010101 00001010	19	19		
AAM = ASCII Adjust for Multiply	11010100 00001010	17	17		
CBW = Convert Byte to Word	10011000	3	3		
CWD = Convert Word to Double Word	10011001	2	2		
LOGIC					
Shift Rotate Instructions					
Not Through Carry (**ROL, ROR, SAL, SAR, SHL,** and **SHR**)					
Register/Memory by 1	1101000w mod TTT r/m	3/7	3/7	b	h
Register/Memory by CL	1101001w mod TTT r/m	3/7	3/7	b	h
Register/Memory by Immediate Count	1100000w mod TTT r/m immed 8-bit data	3/7	3/7	b	h
Through Carry (**RCL** and **RCR**)					
Register/Memory by 1	1101000w mod TTT r/m	9/10	9/10	b	h
Register/Memory by CL	1101001w mod TTT r/m	9/10	9/10	b	h
Register/Memory by Immediate Count	1100000w mod TTT r/m immed 8-bit data	9/10	9/10	b	h
	TTT Instruction: 000 ROL; 001 ROR; 010 RCL; 011 RCR; 100 SHL/SAL; 101 SHR; 111 SAR				
SHLD = Shift Left Double					
Register/Memory by Immediate	00001111 10100100 mod reg r/m immed 8-bit data	3/7	3/7		
Register/Memory by CL	00001111 10100101 mod reg r/m	3/7	3/7		
SHRD = Shift Right Double					
Register/Memory by Immediate	00001111 10101100 mod reg r/m immed 8-bit data	3/7	3/7		
Register/Memory by CL	00001111 10101101 mod reg r/m	3/7	3/7		
AND = And					
Register to Register	001000dw mod reg r/m	2	2		

Table 8-1. 386™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	Clk Count Virtual 8086 Mode	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
LOGIC (Continued)						
Register to Memory	0010000w mod reg r/m		7	7	b	h
Memory to Register	0010001w mod reg r/m		6	6	b	h
Immediate to Register/Memory	1000000w mod 100 r/m immediate data		2/7	2/7	b	h
Immediate to Accumulator (Short Form)	0010010w immediate data		2	2		
TEST = And Function to Flags, No Result						
Register/Memory and Register	1000010w mod reg r/m		2/5	2/5	b	h
Immediate Data and Register/Memory	1111011w mod 000 r/m immediate data		2/5	2/5	b	h
Immediate Data and Accumulator (Short Form)	1010100w immediate data		2	2		
OR = Or						
Register to Register	000010dw mod reg r/m		2	2		
Register to Memory	0000100w mod reg r/m		7	7	b	h
Memory to Register	0000101w mod reg r/m		6	6	b	h
Immediate to Register/Memory	1000000w mod 001 r/m immediate data		2/7	2/7	b	h
Immediate to Accumulator (Short Form)	0000110w immediate data		2	2		
XOR = Exclusive Or						
Register to Register	001100dw mod reg r/m		2	2		
Register to Memory	0011000w mod reg r/m		7	7	b	h
Memory to Register	0011001w mod reg r/m		6	6	b	h
Immediate to Register/Memory	1000000w mod 110 r/m immediate data		2/7	2/7	b	h
Immediate to Accumulator (Short Form)	0011010w immediate data		2	2		
NOT = Invert Register/Memory	1111011w mod 010 r/m		2/6	2/6	b	h
STRING MANIPULATION						
CMPS = Compare Byte Word	1010011w		10	10	b	h
INS = Input Byte/Word from DX Port	0110110w	†29	15	9*/29**	b	h, m
LODS = Load Byte/Word to AL/AX/EAX	1010110w		5	5	b	h
MOVS = Move Byte Word	1010010w		8	8	b	h
OUTS = Output Byte/Word to DX Port	0110111w	†28	14	8*/28**	b	h, m
SCAS = Scan Byte Word	1010111w		8	8	b	h
STOS = Store Byte/Word from AL/AX/EX	1010101w		5	5	b	h
XLAT = Translate String	11010111		5	5		h
REPEATED STRING MANIPULATION Repeated by Count in CX or ECX						
REPE CMPS = Compare String (Find Non-Match)	11110011 1010011w		5+9n	5+9n	b	h

* If CPL ≤ IOPL ** If CPL > IOPL

Table 8-1. 386™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	Clk Count Virtual 8086 Mode	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
REPEATED STRING MANIPULATION (Continued)						
REPNE CMPS = Compare String						
(Find Match)	11110010 1010011w		5+9n	5+9n	b	h
REP INS = Input String	11110010 0110110w	†28+6n	14+6n	8+6n*/28+6n**	b	h, m
REP LODS = Load String	11110010 1010110w		5+6n	5+6n	b	h
REP MOVS = Move String	11110010 1010010w		8+4n	8+4n	b	h
REP OUTS = Output String	11110010 0110111w	†26+5n	12+5n	6+5n*/26+5n**	b	h, m
REPE SCAS = Scan String						
(Find Non-AL/AX/EAX)	11110011 1010111w		5+8n	5+8n	b	h
REPNE SCAS = Scan String						
(Find AL/AX/EAX)	11110010 1010111w		5+8n	5+8n	b	h
REP STOS = Store String	11110010 1010101w		5+5n	5+5n	b	h
BIT MANIPULATION						
BSF = Scan Bit Forward	00001111 10111100 mod reg r/m		11+3n	11+3n	b	h
BSR = Scan Bit Reverse	00001111 10111101 mod reg r/m		9+3n	9+3n	b	h
BT = Test Bit						
Register/Memory, Immediate	00001111 10111010 mod 100 r/m immed 8-bit data		3/6	3/6	b	h
Register/Memory, Register	00001111 10100011 mod reg r/m		3/12	3/12	b	h
BTC = Test Bit and Complement						
Register/Memory, Immediate	00001111 10111010 mod 111 r/m immed 8-bit data		6/8	6/8	b	h
Register/Memory, Register	00001111 10111011 mod reg r/m		6/13	6/13	b	h
BTR = Test Bit and Reset						
Register/Memory, Immediate	00001111 10111010 mod 110 r/m immed 8-bit data		6/8	6/8	b	h
Register/Memory, Register	00001111 10110011 mod reg r/m		6/13	6/13	b	h
BTS = Test Bit and Set						
Register/Memory, Immediate	00001111 10111010 mod 101 r/m immed 8-bit data		6/8	6/8	b	h
Register/Memory, Register	00001111 10101011 mod reg r/m		6/13	6/13	b	h
CONTROL TRANSFER						
CALL = Call						
Direct Within Segment	11101000 full displacement		7+m	7+m	b	r
Register/Memory						
Indirect Within Segment	11111111 mod 010 r/m		7+m/ 10+m	7+m/ 10+m	b	h, r
Direct Intersegment	10011010 unsigned full offset, selector		17+m	34+m	b	j,k,r

NOTES:
† Clock count shown applies if I/O permission allows I/O to the port in virtual 8086 mode. If I/O bit map denies permission exception 13 fault occurs; refer to clock counts for INT 3 instruction.
* If CPL ≤ IOPL ** If CPL > IOPL

Table 8-1. 386™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONTROL TRANSFER (Continued)					
Protected Mode Only (Direct Intersegment)					
Via Call Gate to Same Privilege Level			52+m		h,j,k,r
Via Call Gate to Different Privilege Level, (No Parameters)			86+m		h,j,k,r
Via Call Gate to Different Privilege Level, (x Parameters)			94+4x+m		h,j,k,r
From 80286 Task to 80286 TSS			273		h,j,k,r
From 80286 Task to 386™ CPU TSS			298		h,j,k,r
From 80286 Task to Virtual 8086 Task (386™ CPU TSS)			218		h,j,k,r
From 386™ CPU Task to 80286 TSS			273		h,j,k,r
From 386™ CPU Task to 386™ CPU TSS			300		h,j,k,r
From 386™ CPU Task to Virtual 8086 Task (386™ CPU TSS)			218		h,j,k,r
Indirect Intersegment	11111111 | mod 011 r/m	22+m	38+m	b	h,j,k,r
Protected Mode Only (Indirect Intersegment)					
Via Call Gate to Same Privilege Level			56+m		h,j,k,r
Via Call Gate to Different Privilege Level, (No Parameters)			90+m		h,j,k,r
Via Call Gate to Different Privilege Level, (x Parameters)			98+4x+m		h,j,k,r
From 80286 Task to 80286 TSS			278		h,j,k,r
From 80286 Task to 386™ CPU TSS			303		h,j,k,r
From 80286 Task to Virtual 8086 Task (386™ CPU TSS)			222		h,j,k,r
From 386™ CPU Task to 80286 TSS			278		h,j,k,r
From 386™ CPU Task to 386™ CPU TSS			305		h,j,k,r
From 386™ CPU Task to Virtual 8086 Task (386™ CPU TSS)			222		h,j,k,r
JMP = Unconditional Jump					
Short	11101011 | 8-bit displacement	7+m	7+m		r
Direct within Segment	11101001 | full displacement	7+m	7+m		r
Register/Memory Indirect within Segment	11111111 | mod 100 r/m	7+m/ 10+m	7+m/ 10+m	b	h,r
Direct Intersegment	11101010 | unsigned full offset, selector	12+m	27+m		j,k,r
Protected Mode Only (Direct Intersegment)					
Via Call Gate to Same Privilege Level			45+m		h,j,k,r
From 80286 Task to 80286 TSS			274		h,j,k,r
From 80286 Task to 386™ CPU TSS			301		h,j,k,r
From 80286 Task to Virtual 8086 Task (386™ CPU TSS)			219		h,j,k,r
From 386™ CPU Task to 80286 TSS			270		h,j,k,r
From 386™ CPU Task to 386™ CPU TSS			303		h,j,k,r
From 386™ CPU Task to Virtual 8086 Task (386™ CPU TSS)			221		h,j,k,r
Indirect Intersegment	11111111 | mod 101 r/m	17+m	31+m	b	h,j,k,r
Protected Mode Only (Indirect Intersegment)					
Via Call Gate to Same Privilege Level			49+m		h,j,k,r
From 80286 Task to 80286 TSS			279		h,j,k,r
From 80286 Task to 386™ CPU TSS			306		h,j,k,r
From 80286 Task to Virtual 8086 Task (386™ CPU TSS)			223		h,j,k,r
From 386™ CPU Task to 80286 TSS			275		h,j,k,r
From 386™ CPU Task to 386™ CPU TSS			308		h,j,k,r
From 386™ CPU Task to Virtual 8086 Task (386™ CPU TSS)			225		h,j,k,r

Table 8-1. 386™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONTROL TRANSFER (Continued)					
RET = Return from CALL:					
Within Segment	11000011	10 + m	10 + m	b	g, h, r
Within Segment Adding Immediate to SP	11000010 16-bit displ	10 + m	10 + m	b	g, h, r
Intersegment	11001011	18 + m	32+m	b	g, h, j, k, r
Intersegment Adding Immediate to SP	11001010 16-bit displ	18 + m	32+m	b	g, h, j, k, r
Protected Mode Only (RET):					
to Different Privilege Level					
Intersegment			69		h, j, k, r
Intersegment Adding Immediate to SP			69		h, j, k, r
CONDITIONAL JUMPS					
NOTE: Times Are Jump "Taken or Not Taken"					
JO = Jump on Overflow					
8-Bit Displacement	01110000 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 10000000 full displacement	7 + m or 3	7 + m or 3		r
JNO = Jump on Not Overflow					
8-Bit Displacement	01110001 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 10000001 full displacement	7 + m or 3	7 + m or 3		r
JB/JNAE = Jump on Below/Not Above or Equal					
8-Bit Displacement	01110010 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 10000010 full displacement	7 + m or 3	7 + m or 3		r
JNB/JAE = Jump on Not Below/Above or Equal					
8-Bit Displacement	01110011 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 10000011 full displacement	7 + m or 3	7 + m or 3		r
JE/JZ = Jump on Equal/Zero					
8-Bit Displacement	01110100 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 10000100 full displacement	7 + m or 3	7 + m or 3		r
JNE/JNZ = Jump on Not Equal/Not Zero					
8-Bit Displacement	01110101 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 10000101 full displacement	7 + m or 3	7 + m or 3		r
JBE/JNA = Jump on Below or Equal/Not Above					
8-Bit Displacement	01110110 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 10000110 full displacement	7 + m or 3	7 + m or 3		r
JNBE/JA = Jump on Not Below or Equal/Above					
8-Bit Displacement	01110111 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 10000111 full displacement	7 + m or 3	7 + m or 3		r
JS = Jump on Sign					
8-Bit Displacement	01111000 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 10001000 full displacement	7 + m or 3	7 + m or 3		r

Table 8-1. 386™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONDITIONAL JUMPS (Continued)					
JNS = Jump on Not Sign					
8-Bit Displacement	01111001 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10001001 | full displacement	7 + m or 3	7 + m or 3		r
JP/JPE = Jump on Parity/Parity Even					
8-Bit Displacement	01111010 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10001010 | full displacement	7 + m or 3	7 + m or 3		r
JNP/JPO = Jump on Not Parity/Parity Odd					
8-Bit Displacement	01111011 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10001011 | full displacement	7 + m or 3	7 + m or 3		r
JL/JNGE = Jump on Less/Not Greater or Equal					
8-Bit Displacement	01111100 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10001100 | full displacement	7 + m or 3	7 + m or 3		r
JNL/JGE = Jump on Not Less/Greater or Equal					
8-Bit Displacement	01111101 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10001101 | full displacement	7 + m or 3	7 + m or 3		r
JLE/JNG = Jump on Less or Equal/Not Greater					
8-Bit Displacement	01111110 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10001110 | full displacement	7 + m or 3	7 + m or 3		r
JNLE/JG = Jump on Not Less or Equal/Greater					
8-Bit Displacement	01111111 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10001111 | full displacement	7 + m or 3	7 + m or 3		r
JCXZ = Jump on CX Zero	11100011 | 8-bit displ	9 + m or 5	9 + m or 5		r
JECXZ = Jump on ECX Zero	11100011 | 8-bit displ	9 + m or 5	9 + m or 5		r
(Address Size Prefix Differentiates JCXZ from JECXZ)					
LOOP = Loop CX Times	11100010 | 8-bit displ	11 + m	11 + m		r
LOOPZ/LOOPE = Loop with Zero/Equal	11100001 | 8-bit displ	11 + m	11 + m		r
LOOPNZ/LOOPNE = Loop While Not Zero	11100000 | 8-bit displ	11 + m	11 + m		r
CONDITIONAL BYTE SET NOTE: Times Are Register/Memory					
SETO = Set Byte on Overflow					
To Register/Memory	00001111 | 10010000 | mod000 r/m	4/5	4/5		h
SETNO = Set Byte on Not Overflow					
To Register/Memory	00001111 | 10010001 | mod000 r/m	4/5	4/5		h
SETB/SETNAE = Set Byte on Below/Not Above or Equal					
To Register/Memory	00001111 | 10010010 | mod000 r/m	4/5	4/5		h

Table 8-1. 386™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONDITIONAL BYTE SET (Continued)					
SETNB = Set Byte on Not Below/Above or Equal					
To Register/Memory	00001111 | 10010011 | mod000 r/m	4/5	4/5		h
SETE/SETZ = Set Byte on Equal/Zero					
To Register/Memory	00001111 | 10010100 | mod000 r/m	4/5	4/5		h
SETNE/SETNZ = Set Byte on Not Equal/Not Zero					
To Register/Memory	00001111 | 10010101 | mod000 r/m	4/5	4/5		h
SETBE/SETNA = Set Byte on Below or Equal/Not Above					
To Register/Memory	00001111 | 10010110 | mod000 r/m	4/5	4/5		h
SETNBE/SETA = Set Byte on Not Below or Equal/Above					
To Register/Memory	00001111 | 10010111 | mod000 r/m	4/5	4/5		h
SETS = Set Byte on Sign					
To Register/Memory	00001111 | 10011000 | mod000 r/m	4/5	4/5		h
SETNS = Set Byte on Not Sign					
To Register/Memory	00001111 | 10011001 | mod000 r/m	4/5	4/5		h
SETP/SETPE = Set Byte on Parity/Parity Even					
To Register/Memory	00001111 | 10011010 | mod000 r/m	4/5	4/5		h
SETNP/SETPO = Set Byte on Not Parity/Parity Odd					
To Register/Memory	00001111 | 10011011 | mod000 r/m	4/5	4/5		h
SETL/SETNGE = Set Byte on Less/Not Greater or Equal					
To Register/Memory	00001111 | 10011100 | mod000 r/m	4/5	4/5		h
SETNL/SETGE = Set Byte on Not Less/Greater or Equal					
To Register/Memory	00001111 | 01111101 | mod000 r/m	4/5	4/5		h
SETLE/SETNG = Set Byte on Less or Equal/Not Greater					
To Register/Memory	00001111 | 10011110 | mod000 r/m	4/5	4/5		h
SETNLE/SETG = Set Byte on Not Less or Equal/Greater					
To Register/Memory	00001111 | 10011111 | mod000 r/m	4/5	4/5		h
ENTER = Enter Procedure	11001000 | 16-bit displacement, 8-bit level				
L = 0		10	10	b	h
L = 1		12	12	b	h
L > 1		15 + 4(n − 1)	15 + 4(n − 1)	b	h
LEAVE = Leave Procedure	11001001	4	4	b	h

Table 8-1. 386™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
INTERRUPT INSTRUCTIONS					
INT = Interrupt:					
Type Specified	11001101 | type	37		b	
Type 3	11001100	33		b	
INTO = Interrupt 4 if Overflow Flag Set	11001110				
If OF = 1		35		b, e	
If OF = 0		3	3	b, e	
Bound = Interrupt 5 if Detect Value Out of Range	01100010 | mod reg r/m				
If Out of Range		44		b, e	e, g, h, j, k, r
If In Range		10	10	b, e	e, g, h, j, k, r
Protected Mode Only (INT)					
INT: Type Specified					
Via Interrupt or Trap Gate to Same Privilege Level			59		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			99		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			282		g, j, k, r
From 80286 Task to 386™ CPU TSS via Task Gate			309		g, j, k, r
From 80286 Task to virt 8086 md via Task Gate			226		g, j, k, r
From 386™ CPU Task to 80286 TSS via Task Gate			284		g, j, k, r
From 386™ CPU Task to 386™ CPU TSS via Task Gate			311		g, j, k, r
From 386™ CPU Task to virt 8086 md via Task Gate			228		g, j, k, r
From virt 8086 md to 80286 TSS via Task Gate			289		g, j, k, r
From virt 8086 md to 386™ CPU TSS via Task Gate			316		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			119		
INT: TYPE 3					
Via Interrupt or Trap Gate to Same Privilege Level			59		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			99		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			278		g, j, k, r
From 80286 Task to 386™ CPU TSS via Task Gate			305		g, j, k, r
From 80286 Task to Virt 8086 md via Task Gate			222		g, j, k, r
From 386™ CPU Task to 80286 TSS via Task Gate			280		g, j, k, r
From 386™ CPU Task to 386™ CPU TSS via Task Gate			307		g, j, k, r
From 386™ CPU Task to Virt 8086 md via Task Gate			224		g, j, k, r
From virt 8086 md to 80286 TSS via Task Gate			285		g, j, k, r
From virt 8086 md to 386™ CPU TSS via Task Gate			312		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			119		
INTO:					
Via Interrupt or Trap Grate to Same Privilege Level			59		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			99		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			280		g, j, k, r
From 80286 Task to 386™ CPU TSS via Task Gate			307		g, j, k, r
From 80286 Task to virt 8086 md via Task Gate			224		g, j, k, r
From 386™ CPU Task to 80286 TSS via Task Gate			282		g, j, k, r
From 386™ CPU Task to 386™ CPU TSS via Task Gate			309		g, j, k, r
From 386™ CPU Task to virt 8086 md via Task Gate			225		g, j, k, r
From virt 8086 md to 80286 TSS via Task Gate			287		g, j, k, r
From virt 8086 md to 386™ CPU TSS via Task Gate			314		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			119		

Table 8-1. 386™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT		NOTES	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
INTERRUPT INSTRUCTIONS (Continued)					
BOUND:					
Via Interrupt or Trap Gate to Same Privilege Level			59		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			99		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			254		g, j, k, r
From 80286 Task to 386™ CPU TSS via Task Gate			284		g, j, k, r
From 80268 Task to virt 8086 Mode via Task Gate			231		g, j, k, r
From 386™ CPU Task to 80286 TSS via Task Gate			264		g, j, k, r
From 386™ CPU Task to 386™ CPU TSS via Task Gate			294		g, j, k, r
From 80368 Task to virt 8086 Mode via Task Gate			243		g, j, k, r,
From virt 8086 Mode to 80286 TSS via Task Gate			264		g, j, k, r
From virt 8086 Mode to 386™ CPU TSS via Task Gate			294		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			119		
INTERRUPT RETURN					
IRET = Interrupt Return	11001111	22			g, h, j, k, r
Protected Mode Only (IRET)					
To the Same Privilege Level (within task)			38		g, h, j, k, r
To Different Privilege Level (within task)			82		g, h, j, k, r
From 80286 Task to 80286 TSS			232		h, j, k, r
From 80286 Task to 386™ CPU TSS			265		h, j, k, r
From 80286 Task to Virtual 8086 Task			213		h, j, k, r
From 80286 Task to Virtual 8086 Mode (within task)			60		
From 386™ CPU Task to 80286 TSS			271		h, j, k, r
From 386™ CPU Task to 386™ CPU TSS			275		h, j, k, r
From 386™ CPU Task to Virtual 8086 Task			223		h, j, k, r
From 386™ CPU Task to Virtual 8086 Mode (within task)			60		
PROCESSOR CONTROL					
HLT = HALT	11110100	5	5		I
MOV = Move to and From Control/Debug/Test Registers					
CR0/CR2/CR3 from register	00001111 00100010 11 eee reg	11/4/5	11/4/5		I
Register From CR0-3	00001111 00100000 11 eee reg	6	6		I
DR0-3 From Register	00001111 00100011 11 eee reg	22	22		I
DR6-7 From Register	00001111 00100011 11 eee reg	16	16		I
Register from DR6-7	00001111 00100001 11 eee reg	14	14		I
Register from DR0-3	00001111 00100001 11 eee reg	22	22		I
TR6-7 from Register	00001111 00100110 11 eee reg	12	12		I
Register from TR6-7	00001111 00100100 11 eee reg	12	12		I
NOP = No Operation	10010000	3	3		
WAIT = Wait until BUSY# pin is negated	10011011	7	7		

Table 8-1. 386™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
PROCESSOR EXTENSION INSTRUCTIONS					
Processor Extension Escape	11011TTT | mod LLL r/m TTT and LLL bits are opcode information for coprocessor.	See 80287/80387 data sheets for clock counts			h
PREFIX BYTES					
Address Size Prefix	01100111	0	0		
LOCK = Bus Lock Prefix	11110000	0	0		m
Operand Size Prefix	01100110	0	0		
Segment Override Prefix					
CS:	00101110	0	0		
DS:	00111110	0	0		
ES:	00100110	0	0		
FS:	01100100	0	0		
GS:	01100101	0	0		
SS:	00110110	0	0		
PROTECTION CONTROL					
ARPL = Adjust Requested Privilege Level					
From Register/Memory	01100011 | mod reg r/m	N/A	20/21	a	h
LAR = Load Access Rights					
From Register/Memory	00001111 | 00000010 | mod reg r/m	N/A	15/16	a	g, h, j, p
LGDT = Load Global Descriptor					
Table Register	00001111 | 00000001 | mod 010 r/m	11	11	b, c	h, l
LIDT = Load Interrupt Descriptor					
Table Register	00001111 | 00000001 | mod 011 r/m	11	11	b, c	h, l
LLDT = Load Local Descriptor					
Table Register to Register/Memory	00001111 | 00000000 | mod 010 r/m	N/A	20/24	a	g, h, j, l
LMSW = Load Machine Status Word					
From Register/Memory	00001111 | 00000001 | mod 110 r/m	11/14	11/14	b, c	h, l
LSL = Load Segment Limit					
From Register/Memory	00001111 | 00000011 | mod reg r/m				
Byte-Granular Limit		N/A	21/22	a	g, h, j, p
Page-Granular Limit		N/A	25/26	a	g, h, j, p
LTR = Load Task Register					
From Register/Memory	00001111 | 00000000 | mod 001 r/m	N/A	23/27	a	g, h, j, l
SGDT = Store Global Descriptor					
Table Register	00001111 | 00000001 | mod 000 r/m	9	9	b, c	h
SIDT = Store Interrupt Descriptor					
Table Register	00001111 | 00000001 | mod 001 r/m	9	9	b, c	h
SLDT = Store Local Descriptor Table Register					
To Register/Memory	00001111 | 00000000 | mod 000 r/m	N/A	2/2	a	h

Table 8-1. 386™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION		FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
SMSW	= Store Machine Status Word	00001111 | 00000001 | mod100 r/m	2/2	2/2	b, c	h, l
STR	= Store Task Register					
	To Register/Memory	00001111 | 00000000 | mod001 r/m	N/A	2/2	a	h
VERR	= Verify Read Accesss					
	Register/Memory	00001111 | 00000000 | mod100 r/m	N/A	10/11	a	g, h, j, p
VERW	= Verify Write Accesss	00001111 | 00000000 | mod101 r/m	N/A	15/16	a	g, h, j, p

INSTRUCTION NOTES FOR TABLE 8-1

Notes a through c apply to 386 Microprocessor Real Address Mode only:

a. This is a Protected Mode instruction. Attempted execution in Real Mode will result in exception 6 (invalid opcode).

b. Exception 13 fault (general protection) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum CS, DS, ES, FS or GS limit, FFFFH. Exception 12 fault (stack segment limit violation or not present) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum SS limit.

c. This instruction may be executed in Real Mode. In Real Mode, its purpose is primarily to initialize the CPU for Protected Mode.

Notes d through g apply to 386 Microprocessor Real Address Mode and 386 Microprocessor Protected Virtual Address Mode:

d. The 386 Microprocessor uses an early-out multiply algorithm. The actual number of clocks depends on the position of the most significant bit in the operand (multiplier).

Clock counts given are minimum to maximum. To calculate actual clocks use the following formula:

Actual Clock = if m < > 0 then max ([$\log_2$ |m|], 3) + b clocks:

if m = 0 then 3+b clocks

In this formula, m is the multiplier, and

b = 9 for register to register,

b = 12 for memory to register,

b = 10 for register with immediate to register,

b = 11 for memory with immediate to register.

e. An exception may occur, depending on the value of the operand.

f. LOCK# is automatically asserted, regardless of the presence or absence of the LOCK# prefix.

g. LOCK# is asserted during descriptor table accesses.

Notes h through r apply to 386 Microprocessor Protected Virtual Address Mode only:

h. Exception 13 fault (general protection violation) will occur if the memory operand in CS, DS, ES, FS or GS cannot be used due to either a segment limit violation or access rights violation. If a stack limit is violated, an exception 12 (stack segment limit violation or not present) occurs.

i. For segment load operations, the CPL, RPL, and DPL must agree with the privilege rules to avoid an exception 13 fault (general protection violation). The segment's descriptor must indicate "present" or exception 11 (CS, DS, ES, FS, GS not present). If the SS register is loaded and a stack segment not present is detected, an exception 12 (stack segment limit violation or not present) occurs.

j. All segment descriptor accesses in the GDT or LDT made by this instruction will automatically assert LOCK# to maintain descriptor integrity in multiprocessor systems.

k. JMP, CALL, INT, RET and IRET instructions referring to another code segment will cause an exception 13 (general protection violation) if an applicable privilege rule is violated.

l. An exception 13 fault occurs if CPL is greater than 0 (0 is the most privileged level).

m. An exception 13 fault occurs if CPL is greater than IOPL.

n. The IF bit of the flag register is not updated if CPL is greater than IOPL. The IOPL and VM fields of the flag register are updated only if CPL = 0.

o. The PE bit of the MSW (CR0) cannot be reset by this instruction. Use MOV into CR0 if desiring to reset the PE bit.

p. Any violation of privilege rules as applied to the selector operand does not cause a protection exception; rather, the zero flag is cleared.

q. If the coprocessor's memory operand violates a segment limit or segment access rights, an exception 13 fault (general protection exception) will occur before the ESC instruction is executed. An exception 12 fault (stack segment limit violation or not present) will occur if the stack limit is violated by the operand's starting address.

r. The destination of a JMP, CALL, INT, RET or IRET must be in the defined limit of a code segment or an exception 13 fault (general protection violation) will occur.

8.2 INSTRUCTION ENCODING

8.2.1 Overview

All instruction encodings are subsets of the general instruction format shown in Figure 8-1. Instructions consist of one or two primary opcode bytes, possibly an address specifier consisting of the "mod r/m" byte and "scaled index" byte, a displacement if required, and an immediate data field if required.

Within the primary opcode or opcodes, smaller encoding fields may be defined. These fields vary according to the class of operation. The fields define such information as direction of the operation, size of the displacements, register encoding, or sign extension.

Almost all instructions referring to an operand in memory have an addressing mode byte following the primary opcode byte(s). This byte, the mod r/m byte, specifies the address mode to be used. Certain encodings of the mod r/m byte indicate a second addressing byte, the scale-index-base byte, follows the mod r/m byte to fully specify the addressing mode.

Addressing modes can include a displacement immediately following the mod r/m byte, or scaled index byte. If a displacement is present, the possible sizes are 8, 16 or 32 bits.

If the instruction specifies an immediate operand, the immediate operand follows any displacement bytes. The immediate operand, if specified, is always the last field of the instruction.

Figure 8-1 illustrates several of the fields that can appear in an instruction, such as the mod field and the r/m field, but the Figure does not show all fields. Several smaller fields also appear in certain instructions, sometimes within the opcode bytes themselves. Table 8-2 is a complete list of all fields appearing in the 386 Microprocessor instruction set. Further ahead, following Table 8-2, are detailed tables for each field.

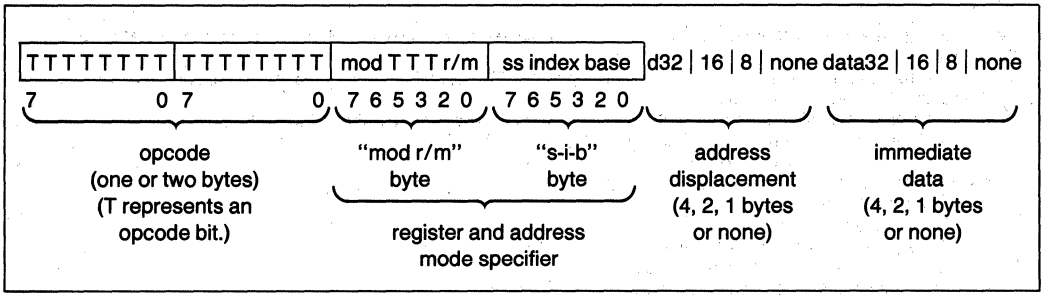


Figure 8-1. General Instruction Format

Table 8-2. Fields within 386™ Microprocessor Instructions

Field Name	Description	Number of Bits
w	Specifies if Data is Byte or Full Size (Full Size is either 16 or 32 Bits	1
d	Specifies Direction of Data Operation	1
s	Specifies if an Immediate Data Field Must be Sign-Extended	1
reg	General Register Specifier	3
mod r/m	Address Mode Specifier (Effective Address can be a General Register)	2 for mod; 3 for r/m
ss	Scale Factor for Scaled Index Address Mode	2
index	General Register to be used as Index Register	3
base	General Register to be used as Base Register	3
sreg2	Segment Register Specifier for CS, SS, DS, ES	2
sreg3	Segment Register Specifier for CS, SS, DS, ES, FS, GS	3
tttn	For Conditional Instructions, Specifies a Condition Asserted or a Condition Negated	4

Note: Table 8-1 shows encoding of individual instructions.

8.2.2 32-Bit Extensions of the Instruction Set

With the 386 Microprocessor, the 8086/80186/80286 instruction set is extended in two orthogonal directions: 32-bit forms of all 16-bit instructions are added to support the 32-bit data types, and 32-bit addressing modes are made available for all instructions referencing memory. This orthogonal instruction set extension is accomplished having a Default (D) bit in the code segment descriptor, and by having 2 prefixes to the instruction set.

Whether the instruction defaults to operations of 16 bits or 32 bits depends on the setting of the D bit in the code segment descriptor, which gives the default length (either 32 bits or 16 bits) for both operands and effective addresses when executing that code segment. In the Real Address Mode or Virtual 8086 Mode, no code segment descriptors are used, but a D value of 0 is assumed internally by the 386 Microprocessor when operating in those modes (for 16-bit default sizes compatible with the 8086/80186/80286).

Two prefixes, the Operand Size Prefix and the Effective Address Size Prefix, allow overriding individually the Default selection of operand size and effective address size. These prefixes may precede any opcode bytes and affect only the instruction they precede. If necessary, one or both of the prefixes may be placed before the opcode bytes. The presence of the Operand Size Prefix and the Effective Address Prefix will toggle the operand size or the effective address size, respectively, to the value "opposite" from the Default setting. For example, if the default operand size is for 32-bit data operations, then presence of the Operand Size Prefix toggles the instruction to 16-bit data operation. As another example, if the default effective address size is 16 bits, presence of the Effective Address Size prefix toggles the instruction to use 32-bit effective address computations.

These 32-bit extensions are available in all 386 Microprocessor modes, including the Real Address Mode or the Virtual 8086 Mode. In these modes the default is always 16 bits, so prefixes are needed to specify 32-bit operands or addresses. For instructions with more than one prefix, the order of prefixes is unimportant.

Unless specified otherwise, instructions with 8-bit and 16-bit operands do not affect the contents of the high-order bits of the extended registers.

8.2.3 Encoding of Instruction Fields

Within the instruction are several fields indicating register selection, addressing mode and so on. The exact encodings of these fields are defined immediately ahead.

8.2.3.1 ENCODING OF OPERAND LENGTH (w) FIELD

For any given instruction performing a data operation, the instruction is executing as a 32-bit operation or a 16-bit operation. Within the constraints of the operation size, the w field encodes the operand size as either one byte or the full operation size, as shown in the table below.

w Field	Operand Size During 16-Bit Data Operations	Operand Size During 32-Bit Data Operations
0	8 Bits	8 Bits
1	16 Bits	32 Bits

8.2.3.2 ENCODING OF THE GENERAL REGISTER (reg) FIELD

The general register is specified by the reg field, which may appear in the primary opcode bytes, or as the reg field of the "mod r/m" byte, or as the r/m field of the "mod r/m" byte.

Encoding of reg Field When w Field is not Present in Instruction

reg Field	Register Selected During 16-Bit Data Operations	Register Selected During 32-Bit Data Operations
000	AX	EAX
001	CX	ECX
010	DX	EDX
011	BX	EBX
100	SP	ESP
101	BP	EBP
101	SI	ESI
101	DI	EDI

Encoding of reg Field When w Field is Present in Instruction

Register Specified by reg Field During 16-Bit Data Operations:		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Register Specified by reg Field During 32-Bit Data Operations		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	EAX
001	CL	ECX
010	DL	EDX
011	BL	EBX
100	AH	ESP
101	CH	EBP
110	DH	ESI
111	BH	EDI

8.2.3.3 ENCODING OF THE SEGMENT REGISTER (sreg) FIELD

The sreg field in certain instructions is a 2-bit field allowing one of the four 80286 segment registers to be specified. The sreg field in other instructions is a 3-bit field, allowing the 386 Microprocessor FS and GS segment registers to be specified.

2-Bit sreg2 Field

2-Bit sreg2 Field	Segment Register Selected
00	ES
01	CS
10	SS
11	DS

3-Bit sreg3 Field

3-Bit sreg3 Field	Segment Register Selected
000	ES
001	CS
010	SS
011	DS
100	FS
101	GS
110	do not use
111	do not use

8.2.3.4 ENCODING OF ADDRESS MODE

Except for special instructions, such as PUSH or POP, where the addressing mode is pre-determined, the addressing mode for the current instruction is specified by addressing bytes following the primary opcode. The primary addressing byte is the "mod r/m" byte, and a second byte of addressing information, the "s-i-b" (scale-index-base) byte, can be specified.

The s-i-b byte (scale-index-base byte) is specified when using 32-bit addressing mode and the "mod r/m" byte has r/m = 100 and mod = 00, 01 or 10. When the sib byte is present, the 32-bit addressing mode is a function of the mod, ss, index, and base fields.

The primary addressing byte, the "mod r/m" byte, also contains three bits (shown as TTT in Figure 8-1) sometimes used as an extension of the primary opcode. The three bits, however, may also be used as a register field (reg).

When calculating an effective address, either 16-bit addressing or 32-bit addressing is used. 16-bit addressing uses 16-bit address components to calculate the effective address while 32-bit addressing uses 32-bit address components to calculate the effective address. When 16-bit addressing is used, the "mod r/m" byte is interpreted as a 16-bit addressing mode specifier. When 32-bit addressing is used, the "mod r/m" byte is interpreted as a 32-bit addressing mode specifier.

Tables on the following three pages define all encodings of all 16-bit addressing modes and 32-bit addressing modes.

Encoding of 16-bit Address Mode with "mod r/m" Byte

mod r/m	Effective Address
00 000	DS:[BX + SI]
00 001	DS:[BX + DI]
00 010	SS:[BP + SI]
00 011	SS:[BP + DI]
00 100	DS:[SI]
00 101	DS:[DI]
00 110	DS:d16
00 111	DS:[BX]
01 000	DS:[BX + SI + d8]
01 001	DS:[BX + DI + d8]
01 010	SS:[BP + SI + d8]
01 011	SS:[BP + DI + d8]
01 100	DS:[SI + d8]
01 101	DS:[DI + d8]
01 110	SS:[BP + d8]
01 111	DS:[BX + d8]

mod r/m	Effective Address
10 000	DS:[BX + SI + d16]
10 001	DS:[BX + DI + d16]
10 010	SS:[BP + SI + d16]
10 011	SS:[BP + DI + d16]
10 100	DS:[SI + d16]
10 101	DS:[DI + d16]
10 110	SS:[BP + d16]
10 111	DS:[BX + d16]
11 000	register—see below
11 001	register—see below
11 010	register—see below
11 011	register—see below
11 100	register—see below
11 101	register—see below
11 110	register—see below
11 111	register—see below

Register Specified by r/m During 16-Bit Data Operations		
mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Register Specified by r/m During 32-Bit Data Operations		
mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Encoding of 32-bit Address Mode with "mod r/m" byte (no "s-i-b" byte present):

mod r/m	Effective Address
00 000	DS:[EAX]
00 001	DS:[ECX]
00 010	DS:[EDX]
00 011	DS:[EBX]
00 100	s-i-b is present
00 101	DS:d32
00 110	DS:[ESI]
00 111	DS:[EDI]
01 000	DS:[EAX + d8]
01 001	DS:[ECX + d8]
01 010	DS:[EDX + d8]
01 011	DS:[EBX + d8]
01 100	s-i-b is present
01 101	SS:[EBP + d8]
01 110	DS:[ESI + d8]
01 111	DS:[EDI + d8]

mod r/m	Effective Address
10 000	DS:[EAX + d32]
10 001	DS:[ECX + d32]
10 010	DS:[EDX + d32]
10 011	DS:[EBX + d32]
10 100	s-i-b is present
10 101	SS:[EBP + d32]
10 110	DS:[ESI + d32]
10 111	DS:[EDI + d32]
11 000	register—see below
11 001	register—see below
11 010	register—see below
11 011	register—see below
11 100	register—see below
11 101	register—see below
11 110	register—see below
11 111	register—see below

Register Specified by reg or r/m during 16-Bit Data Operations:		
mod r/m	function of w field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Register Specified by reg or r/m during 32-Bit Data Operations:		
mod r/m	function of w field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Encoding of 32-bit Address Mode ("mod r/m" byte and "s-i-b" byte present):

mod base	Effective Address
00 000	DS:[EAX + (scaled index)]
00 001	DS:[ECX + (scaled index)]
00 010	DS:[EDX + (scaled index)]
00 011	DS:[EBX + (scaled index)]
00 100	SS:[ESP + (scaled index)]
00 101	DS:[d32 + (scaled index)]
00 110	DS:[ESI + (scaled index)]
00 111	DS:[EDI + (scaled index)]
01 000	DS:[EAX + (scaled index) + d8]
01 001	DS:[ECX + (scaled index) + d8]
01 010	DS:[EDX + (scaled index) + d8]
01 011	DS:[EBX + (scaled index) + d8]
01 100	SS:[ESP + (scaled index) + d8]
01 101	SS:[EBP + (scaled index) + d8]
01 110	DS:[ESI + (scaled index) + d8]
01 111	DS:[EDI + (scaled index) + d8]
10 000	DS:[EAX + (scaled index) + d32]
10 001	DS:[ECX + (scaled index) + d32]
10 010	DS:[EDX + (scaled index) + d32]
10 011	DS:[EBX + (scaled index) + d32]
10 100	SS:[ESP + (scaled index) + d32]
10 101	SS:[EBP + (scaled index) + d32]
10 110	DS:[ESI + (scaled index) + d32]
10 111	DS:[EDI + (scaled index) + d32]

NOTE:
Mod field in "mod r/m" byte; ss, index, base fields in "s-i-b" byte.

ss	Scale Factor
00	x1
01	x2
10	x4
11	x8

Index	Index Register
000	EAX
001	ECX
010	EDX
011	EBX
100	no index reg**
101	EBP
110	ESI
111	EDI

****IMPORTANT NOTE:**
When index field is 100, indicating "no index register," then ss field MUST equal 00. If index is 100 and ss does not equal 00, the effective address is undefined.

8.2.3.5 ENCODING OF OPERATION DIRECTION (d) FIELD

In many two-operand instructions the d field is present to indicate which operand is considered the source and which is the destination.

d	Direction of Operation
0	Register/Memory <- - Register "reg" Field Indicates Source Operand; "mod r/m" or "mod ss index base" Indicates Destination Operand
1	Register <- - Register/Memory "reg" Field Indicates Destination Operand; "mod r/m" or "mod ss index base" Indicates Source Operand

8.2.3.6 ENCODING OF SIGN-EXTEND (s) FIELD

The s field occurs primarily to instructions with immediate data fields. The s field has an effect only if the size of the immediate data is 8 bits and is being placed in a 16-bit or 32-bit destination.

s	Effect on Immediate Data8	Effect on Immediate Data 16|32
0	None	None
1	Sign-Extend Data8 to Fill 16-Bit or 32-Bit Destination	None

8.2.3.7 ENCODING OF CONDITIONAL TEST (tttn) FIELD

For the conditional instructions (conditional jumps and set on condition), tttn is encoded with n indicating to use the condition (n=0) or its negation (n=1), and ttt giving the condition to test.

Mnemonic	Condition	tttn
O	Overflow	0000
NO	No Overflow	0001
B/NAE	Below/Not Above or Equal	0010
NB/AE	Not Below/Above or Equal	0011
E/Z	Equal/Zero	0100
NE/NZ	Not Equal/Not Zero	0101
BE/NA	Below or Equal/Not Above	0110
NBE/A	Not Below or Equal/Above	0111
S	Sign	1000
NS	Not Sign	1001
P/PE	Parity/Parity Even	1010
NP/PO	Not Parity/Parity Odd	1011
L/NGE	Less Than/Not Greater or Equal	1100
NL/GE	Not Less Than/Greater or Equal	1101
LE/NG	Less Than or Equal/Greater Than	1110
NLE/G	Not Less or Equal/Greater Than	1111

8.2.3.8 ENCODING OF CONTROL OR DEBUG OR TEST REGISTER (eee) FIELD

For the loading and storing of the Control, Debug and Test registers.

When Interpreted as Control Register Field

eee Code	Reg Name
000	CR0
010	CR2
011	CR3
Do not use any other encoding	

When Interpreted as Debug Register Field

eee Code	Reg Name
000	DR0
001	DR1
010	DR2
011	DR3
110	DR6
111	DR7
Do not use any other encoding	

When Interpreted as Test Register Field

eee Code	Reg Name
110	TR6
111	TR7
Do not use any other encoding	

9.0 Revision History

This 386 Microprocessor data sheet, version -005, contains updates and improvements to previous versions. A revision summary is listed here for your convenience.

The sections significantly revised since version -001 are:

2.9.6	Sequence of exception checking table added.
2.9.7	Instruction restart revised.
2.11.2	TLB testing revised.
2.12	Debugging support revised.
3.1	LOCK prefix restricted to certain instructions.
4.4.3.3	I/O privilege level and I/O permission bitmap added.
Figures 4-15a, 4-15b	I/O permission bitmap added.
4.6.4	Protection and I/O permission bitmap revised.
4.6.6	Entering and leaving virtual 8086 mode through task switches, trap and interrupt gates, and IRET explained.
5.6	Self-test signature stored in EAX.
5.8	Coprocessor interface description added.
5.8.1	Software testing for coprocessor presence added.
Table 6-3	PGA package thermal characteristics added.
7.6	Designing for ICE-386 revised.
Figures 7-8, 7-9, 7-10	ICE-386 clearance requirements added.
8.2.3.4	Encoding of 32-bit address mode with no "sib" byte corrected.

The sections significantly revised since version -002 are:

Table 2-5	Interrupt vector assignments updated.
Figure 4-15a	Bit_map_offset must be less than or equal to DFFFH.
Figure 5-28	386 Microprocessor outputs remain in their reset state during self-test.
5.7	Component and revision identifier history updated.
7.4	20 MHz D.C. specifications added.
7.5	16 MHz A.C. specifications updated. 20 MHz A.C. specifications added.
Table 8-1	Clock counts updated.

The sections significantly revised since version -003 are:

Table 2-6b	Interrupt priorities 2 and 3 interchanged.
2.9.8	Double page faults do not raise double fault exception.
Figure 4-5	Maximum-sized segments must have segments $Base_{11..0} = 0$.
5.4.3.4	BS16# timing corrected.
Figures 5-16, 5-17, 5-19, 5-22	BS16# timing corrected. BS16# must not be asserted once NA# has been sampled asserted in the current bus cycle.
7.5	16 MHz and 20 MHz A.C. specifications revised. All timing parameters are now guaranteed at 1.5V test levels. The timing parameters have been adjusted to remain compatible with previous 0.8V/2.0V specifications.

The sections significantly revised since version -004 are:

Chapter 4	25 MHz Clock data included.
Table 2-4	Segment Register Selection Rules updated.
5.4.4	Interrupt Acknowledge Cycles discussion corrected.
Table 5-10	Additional Stepping Information added.
Table 7-3	I_{CC} values updated.
7.5.2	Table for 25 MHz A.C. Characteristics added. A.C. Characteristics tables reordered.
Figure 7-5	Output Valid Delay Timing Figure reconfigured. Partial data now provided in additional Figures 7-5a and 7-5b.
Table 8-1	Clock counts updated and formats corrected.

80387
80-BIT CHMOS III
NUMERIC PROCESSOR EXTENSION

- **High Performance 80-Bit Internal Architecture**
- **Implements ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic**
- **Five to Nine Times 8087/80287 Performance**
- **Expands 386™ CPU Data Types to Include 32-, 64-, 80-Bit Floating Point, 32-, 64-Bit Integers and 18-Digit BCD Operands**
- **Directly Extends 386™ CPU Instruction Set to Include Trigonometric, Logarithmic, Exponential and Arithmetic Instructions for All Data Types**
- **Upward Object-Code Compatible from 8087 and 80287**
- **Full-Range Transcendental Operations for SINE, COSINE, TANGENT, ARCTANGENT and LOGARITHM**
- **Built-In Exception Handling**
- **Operates Independently of Real, Protected and Virtual-8086 Modes of the 386™ Microprocessor**
- **Eight 80-Bit Numeric Registers, Usable as Individually Addressable General Registers or as a Register Stack**
- **Available in 68-Pin PGA Package**
  (See Packaging Spec: Order #231369)

The Intel 80387 is a high-performance numerics processor extension that extends the Intel386™ Architecture with floating point, extended integer and BCD data types. The 386™ Microprocessor and the 80387 Coprocessor computing system fully conforms to the ANSI/IEEE floating-point standard. Using a numerics oriented architecture, the 80387 adds over seventy mnemonics to the 386 Microprocessor instruction set, making the 386 Microprocessor and the 80387 Coprocessor a complete solution for high-performance numerics processing. The 80387 is implemented with 1.5 micron, high-speed CHMOS III technology and packaged in a 68-pin ceramic pin grid array (PGA) package. The 386 Microprocessor and the 80387 Coprocessor are upward object-code compatible from the 386 Microprocessor and the 80287 Coprocessor, 80286/80287 and 8086/8087 computing systems.

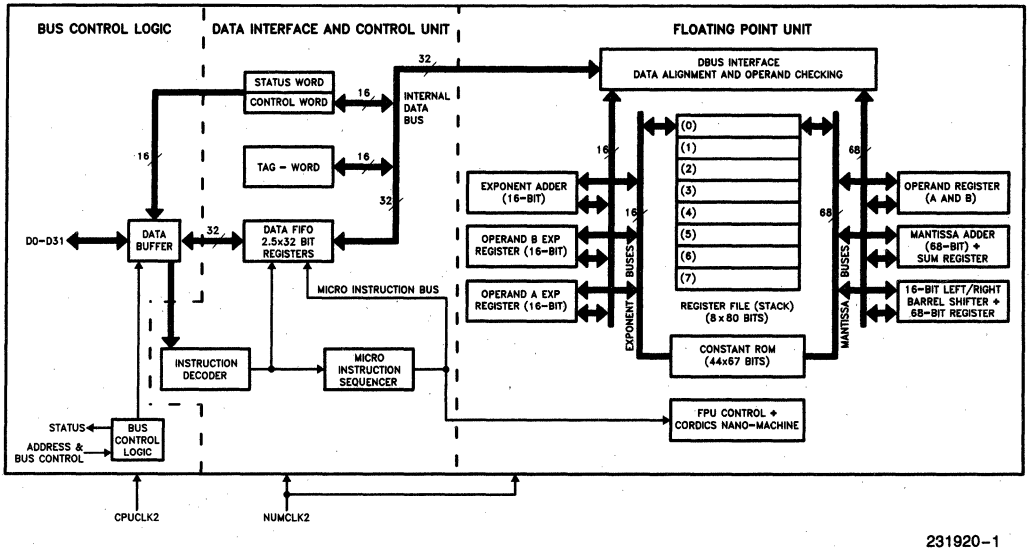


Figure 0.1. 80387 Block Diagram

November 1988
Order Number: 231920-004

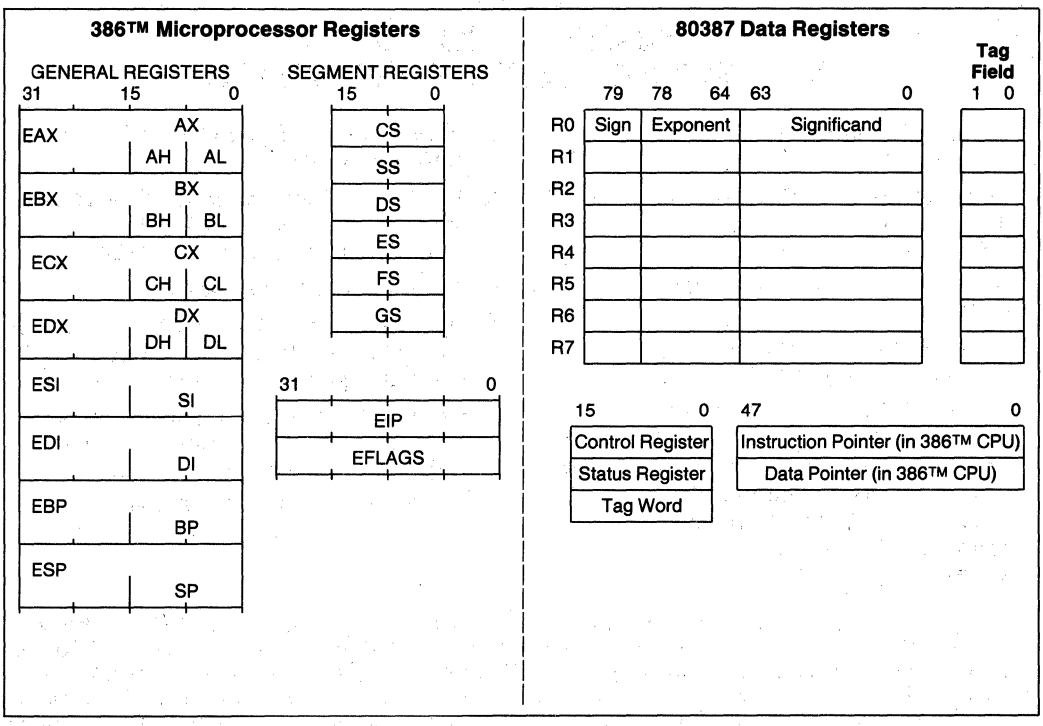


Figure 1.1. 386™ Microprocessor and 80387 Coprocessor Register Set

1.0 FUNCTIONAL DESCRIPTION

The 80387 Numeric Processor Extension (NPX) provides arithmetic instructions for a variety of numeric data types in the 386™ Microprocessor and 80387 Coprocessor systems. It also executes numerous built-in transcendental functions (e.g. tangent, sine, cosine, and log functions). The 80387 effectively extends the register and instruction set of a 386 Microprocessor system for existing data types and adds several new data types as well. Figure 1.1 shows the model of registers visible to the 386 Microprocessor and 80387 Coprocessor programs. Essentially, the 80387 can be treated as an additional resource or an extension to the 386 Microprocessor. The 386 Microprocessor together with an 80387 can be used as a single unified system, the 386 Microprocessor and 80387 Coprocessor.

The 80387 works the same whether the 386 Microprocessor is executing in real-address mode, protected mode, or virtual-8086 mode. All memory access is handled by the 386 Microprocessor; the 80387 merely operates on instructions and values passed to it by the 386 Microprocessor. Therefore, the 80387 is not sensitive to the processing mode of the 386 Microprocessor.

In real-address mode and virtual-8086 mode, the 386 Microprocessor and 80387 Coprocessor is completely upward compatible with software for 8086/8087, 80286/80287 real-address mode, and 386 Microprocessor and 80287 Coprocessor real-address mode systems.

In protected mode, the 386 Microprocessor and 80387 Coprocessor is completely upward compatible with software for 80286/80287 protected mode, and 386 Microprocessor and 80287 Coprocessor protected mode systems.

The only differences of operation that may appear when 8086/8087 programs are ported to a protected-mode 386 Microprocessor and 80387 Coprocessor system (*not* using virtual-8086 mode), is in the format of operands for the administrative instructions FLDENV, FSTENV, FRSTOR and FSAVE. These instructions are normally used only by exception handlers and operating systems, not by applications programs.

The 80387 contains three functional units that can operate in parallel to increase system performance. The 386 Microprocessor can be transferring commands and data to the 80387 *bus control logic* for the next instruction while the 80387 *floating-point unit* is performing the current numeric instruction.

2.0 PROGRAMMING INTERFACE

The 80387 adds to the 386 Microprocessor system additional data types, registers, instructions, and interrupts specifically designed to facilitate high-speed numerics processing. To use the 80387 requires no special programming tools, because all new instructions and data types are directly supported by the 386 CPU assembler and compilers for high-level languages. All 8086/8088 development tools that support the 8087 can also be used to develop software for the 386 Microprocessor and 80387 Coprocessor in real-address mode or virtual-8086 mode. All 80286 development tools that support the 80287 can also be used to develop software for the 386 Microprocessor and 80387 Coprocessor.

All communication between the 386 Microprocessor and the 80387 is transparent to applications software. The CPU automatically controls the 80387 whenever a numerics instruction is executed. All physical memory and virtual memory of the CPU are available for storage of the instructions and operands of programs that use the 80387. All memory addressing modes, including use of displacement, base register, index register, and scaling, are available for addressing numerics operands.

Section 6 at the end of this data sheet lists by class the instructions that the 80387 adds to the instruction set of the 386 Microprocessor system.

2.1 Data Types

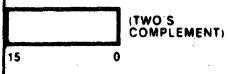
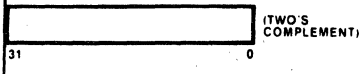
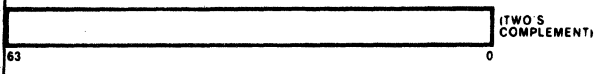
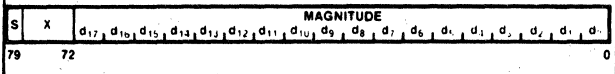
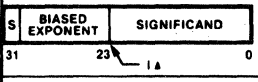
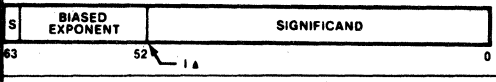
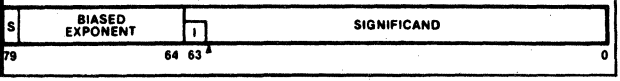
Table 2.1 lists the seven data types that the 80387 supports and presents the format for each type. Operands are stored in memory with the least significant digit at the lowest memory address. Programs retrieve these values by generating the lowest address. For maximum system performance, all operands should start at physical-memory addresses evenly divisible by four (doubleword boundaries); operands may begin at any other addresses, but will require extra memory cycles to access the entire operand.

Internally, the 80387 holds all numbers in the extended-precision real format. Instructions that load operands from memory automatically convert operands represented in memory as 16-, 32-, or 64-bit integers, 32- or 64-bit floating-point numbers, or 18-digit packed BCD numbers into extended-precision real format. Instructions that store operands in memory perform the inverse type conversion.

2.2 Numeric Operands

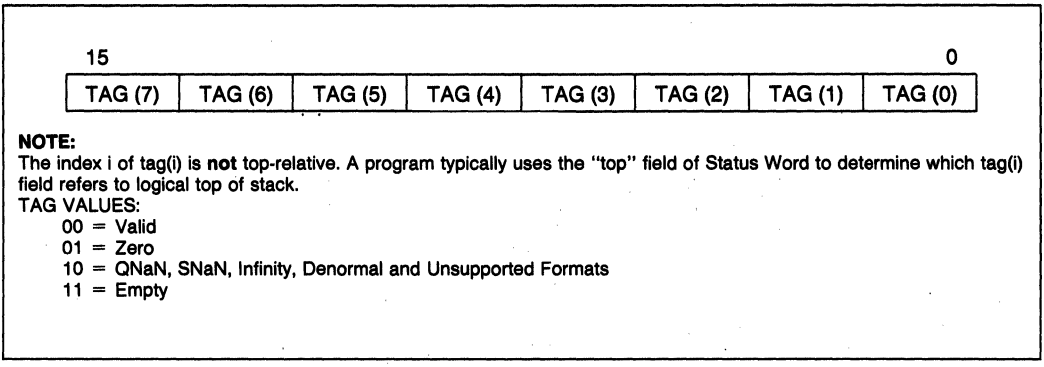
A typical NPX instruction accepts one or two operands and produces a single result. In two-operand instructions, one operand is the contents of an NPX register, while the other may be a memory location. The operands of some instructions are predefined; for example FSQRT always takes the square root of the number in the top stack element.

Table 2.1. 80387 Data Type Representation in Memory

Data Formats	Range	Precision	Most Significant Byte = Highest Addressed Byte
Word Integer	$\pm 10^4$	16 Bits	
Short Integer	$\pm 10^9$	32 Bits	
Long Integer	$\pm 10^{18}$	64 Bits	
Packed BCD	$\pm 10^{\pm 18}$	18 Digits	
Single Precision	$\pm 10^{\pm 38}$	24 Bits	
Double Precision	$\pm 10^{\pm 308}$	53 Bits	
Extended Precision	$\pm 10^{\pm 4932}$	64 Bits	

231920-2

NOTES:
(1) S = Sign bit (0 = positive, 1 = negative)
(2) d_n = Decimal digit (two per byte)
(3) X = Bits have no significance; 80387 ignores when loading, zeros when storing
(4)▲ = Position of implicit binary point
(5) I = Integer bit of significand; stored in temporary real, implicit in single and double precision
(6) Exponent Bias (normalized values):
Single: 127 (7FH)
Double: 1023 (3FFH)
Extended Real: 16383 (3FFFH)
(7) Packed BCD: $(-1)^S$ $(D_{17}...D_0)$
(8) Real: $(-1)^S$ $(2^{E\text{-BIAS}})$ $(F_0\ F_1...)$



NOTE:
The index i of tag(i) is **not** top-relative. A program typically uses the "top" field of Status Word to determine which tag(i) field refers to logical top of stack.
TAG VALUES:
00 = Valid
01 = Zero
10 = QNaN, SNaN, Infinity, Denormal and Unsupported Formats
11 = Empty

Figure 2.1. 80387 Tag Word

2.3 Register Set

Figure 1.1 shows the 80387 register set. When an 80387 is present in a system, programmers may use these registers in addition to the registers normally available on the 386 CPU.

2.3.1 DATA REGISTERS

80387 computations use the 80387's data registers. These eight 80-bit registers provide the equivalent capacity of twenty 32-bit registers. Each of the eight data registers in the 80387 is 80 bits wide and is divided into "fields" corresponding to the NPXs extended-precision real data type.

The 80387 register set can be accessed either as a stack, with instructions operating on the top one or two stack elements, or as a fixed register set, with instructions operating on explicitly designated registers. The TOP field in the status word identifies the current top-of-stack register. A "push" operation decrements TOP by one and loads a value into the new top register. A "pop" operation stores the value from the current top register and then increments TOP by one. Like the 386 Microprocessor stacks in memory, the 80387 register stack grows "down" toward lower-addressed registers.

Instructions may address the data registers either implicitly or explicitly. Many instructions operate on the register at the TOP of the stack. These instructions implicitly address the register at which TOP points. Other instructions allow the programmer to explicitly specify which register to user. This explicit register addressing is also relative to TOP.

2.3.2 TAG WORD

The tag word marks the content of each numeric data register, as Figure 2.1 shows. Each two-bit tag represents one of the eight numerics registers. The principal function of the tag word is to optimize the NPXs performance and stack handling by making it possible to distinguish between empty and nonempty register locations. It also enables exception handlers to check the contents of a stack location without the need to perform complex decoding of the actual data.

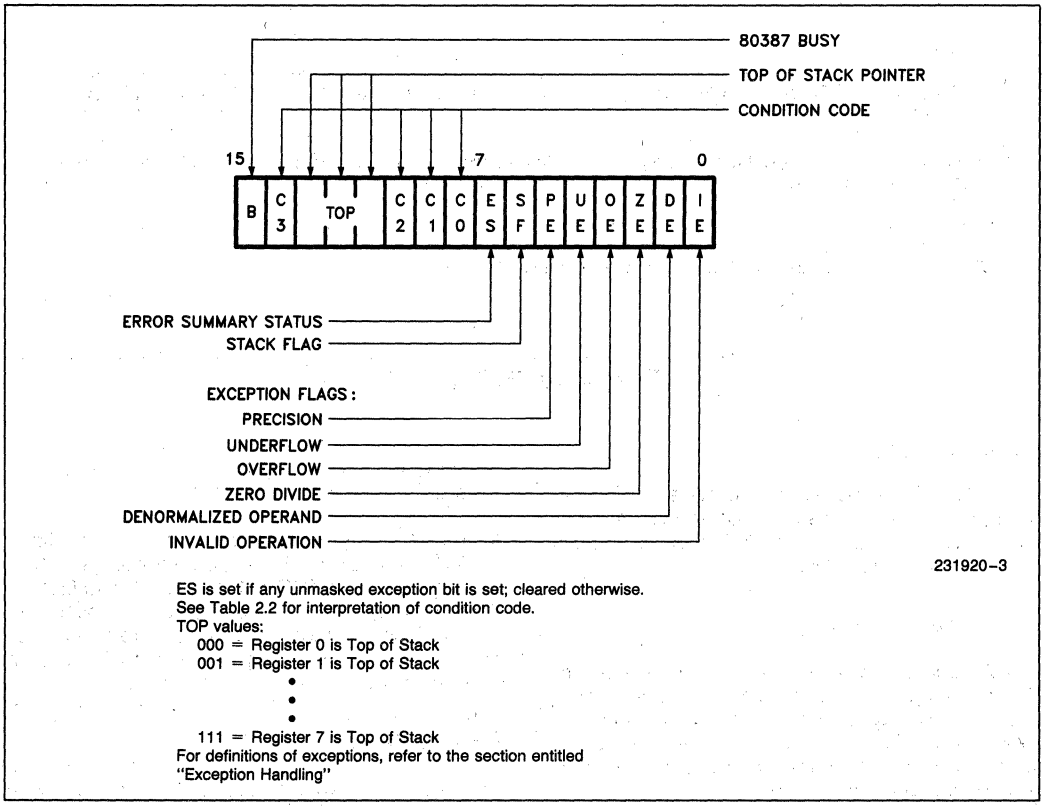


Figure 2.2. 80387 Status Word

2.3.3 STATUS WORD

The 16-bit status word (in the status register) shown in Figure 2.2 reflects the overall state of the 80387. It may be read and inspected by CPU code.

Bit 15, the B-bit (busy bit) is included for 8087 compatibility only. It reflects the contents of the ES bit (bit 7 of the status word), not the status of the BUSY# output of 80387.

Bits 13–11 (TOP) point to the 80387 register that is the current top-of-stack.

The four numeric condition code bits (C_3–C_0) are similar to the flags in a CPU; instructions that perform arithmetic operations update these bits to reflect the outcome. The effects of these instructions on the condition code are summarized in Tables 2.2 through 2.5.

Bit 7 is the error summary (ES) status bit. This bit is set if any unmasked exception bit is set; it is clear otherwise. If this bit is set, the ERROR# signal is asserted.

Bit 6 is the stack flag (SF). This bit is used to distinguish invalid operations due to stack overflow or underflow from other kinds of invalid operations. When SF is set, bit 9 (C_1) distinguishes between stack overflow (C_1 = 1) and underflow (C_1 = 0).

Figure 2.2 shows the six exception flags in bits 5–0 of the status word. Bits 5–0 are set to indicate that the 80387 has detected an exception while executing an instruction. A later section entitled "Exception Handling" explains how they are set and used.

Note that when a new value is loaded into the status word by the FLDENV or FRSTOR instruction, the value of ES (bit 7) and its reflection in the B-bit (bit 15) are not derived from the values loaded from memory but rather are dependent upon the values of the exception flags (bits 5–0) in the status word and their corresponding masks in the control word. If ES is set in such a case, the ERROR# output of the 80387 is activated immediately.

Table 2.2. Condition Code Interpretation

Instruction	C0 (S)	C3 (Z)	C1 (A)	C2 (C)
FPREM, FPREM1 (see Table 2.3)	Three least significant bits of quotient Q2	Q0	Q1 or O/U#	Reduction 0 = complete 1 = incomplete
FCOM, FCOMP, FCOMPP, FTST, FUCOM, FUCOMP, FUCOMPP, FICOM, FICOMP	Result of comparison (see Table 2.4)		Zero or O/U#	Operand is not comparable (Table 2.4)
FXAM	Operand class (see Table 2.5)		Sign or O/U#	Operand class (Table 2.5)
FCHS, FABS, FXCH, FINCTOP, FDECTOP, Constant loads, FXTRACT, FLD, FILD, FBLD, FSTP (ext real)	UNDEFINED		Zero or O/U#	UNDEFINED
FIST, FBSTP, FRNDINT, FST, FSTP, FADD, FMUL, FDIV, FDIVR, FSUB, FSUBR, FSCALE, FSQRT, FPATAN, F2XM1, FYL2X, FYL2XP1	UNDEFINED		Roundup or O/U#	UNDEFINED
FPTAN, FSIN FCOS, FSINCOS	UNDEFINED		Roundup or O/U#, undefined if C2 = 1	Reduction 0 = complete 1 = incomplete
FLDENV, FRSTOR	Each bit loaded from memory			
FLDCW, FSTENV, FSTCW, FSTSW, FCLEX, FINIT, FSAVE	UNDEFINED			

O/U#	When both IE and SF bits of status word are set, indicating a stack exception, this bit distinguishes between stack overflow (C1 = 1) and underflow (C1 = 0).
Reduction	If FPREM or FPREM1 produces a remainder that is less than the modulus, reduction is complete. When reduction is incomplete the value at the top of the stack is a partial remainder, which can be used as input to further reduction. For FPTAN, FSIN, FCOS, and FSINCOS, the reduction bit is set if the operand at the top of the stack is too large. In this case the original operand remains at the top of the stack.
Roundup	When the PE bit of the status word is set, this bit indicates whether the last rounding in the instruction was upward.
UNDEFINED	Do not rely on finding any specific value in these bits.

Table 2.3. Condition Code Interpretation after FPREM and FPREM1 Instructions

Condition Code				Interpretation after FPREM and FPREM1	
C2	C3	C1	C0		
1	X	X	X	Incomplete Reduction: further interation required for complete reduction	
	Q1	Q0	Q2	Q MOD8	
0	0	0	0	0	Complete Reduction: C0, C3, C1 contain three least significant bits of quotient
	0	1	0	1	
	1	0	0	2	
	1	1	0	3	
	0	0	1	4	
	0	1	1	5	
	1	0	1	6	
	1	1	1	7	

Table 2.4. Condition Code Resulting from Comparison

Order	C3	C2	C0
TOP > Operand	0	0	0
TOP < Operand	0	0	1
TOP = Operand	1	0	0
Unordered	1	1	1

Table 2.5. Condition Code Defining Operand Class

C3	C2	C1	C0	Value at TOP
0	0	0	0	+ Unsupported
0	0	0	1	+ NaN
0	0	1	0	− Unsupported
0	0	1	1	− NaN
0	1	0	0	+ Normal
0	1	0	1	+ Infinity
0	1	1	0	− Normal
0	1	1	1	− Infinity
1	0	0	0	+ 0
1	0	0	1	+ Empty
1	0	1	0	− 0
1	0	1	1	− Empty
1	1	0	0	+ Denormal
1	1	1	0	− Denormal

2.3.4 INSTRUCTION AND DATA POINTERS

Because the NPX operates in parallel with the CPU, any errors detected by the NPX may be reported after the CPU has executed the ESC instruction which caused it. To allow identification of the failing numeric instruction, the 386 Microprocessor and 80387 Coprocessor contains two pointer registers that supply the address of the failing numeric instruction and the address of its numeric memory operand (if appropriate).

The instruction and data pointers are provided for user-written error handlers. These registers are actually located in the 386 CPU, but appear to be located in the 80387 because they are accessed by the ESC instructions FLDENV, FSTENV, FSAVE, and FRSTOR. (In the 8086/8087 and 80286/80287, these registers are located in the NPX.) Whenever the 386 CPU decodes a new ESC instruction, it saves the address of the instruction (including any prefixes that may be present), the address of the operand (if present), and the opcode.

The instruction and data pointers appear in one of four formats depending on the operating mode of the 386 Microprocessor (protected mode or real-address mode) and depending on the operand-size attribute in effect (32-bit operand or 16-bit operand). When the 386 Microprocessor is in virtual-8086 mode, the real-address mode formats are used. (See Figures 2.3 through 2.6.) The ESC instructions FLDENV, FSTENV, FSAVE, and FRSTOR are used to transfer these values between the 386 Microprocessor registers and memory. Note that the value of the data pointer is *undefined* if the prior ESC instruction did not have a memory operand.

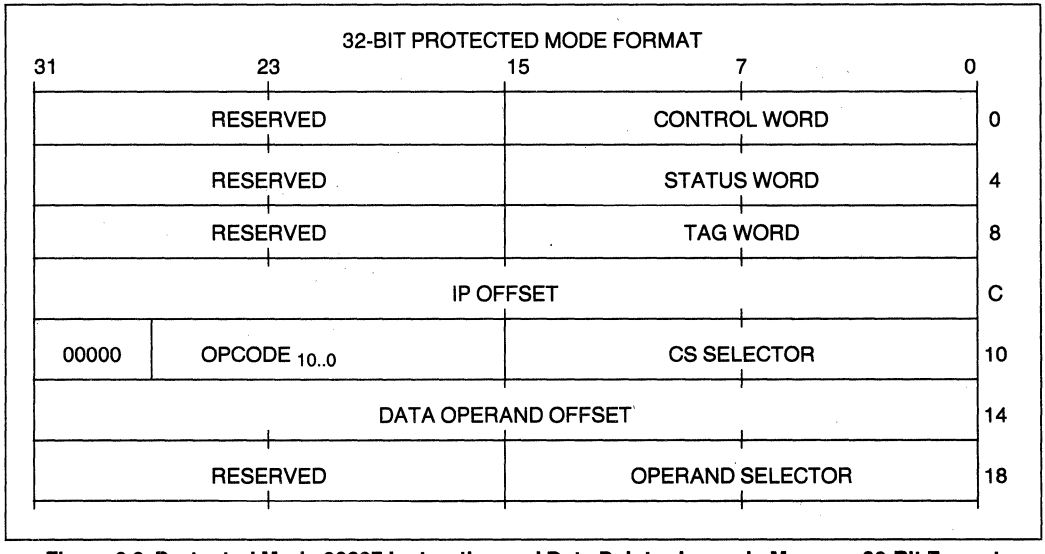


Figure 2.3. Protected Mode 80387 Instruction and Data Pointer Image in Memory, 32-Bit Format

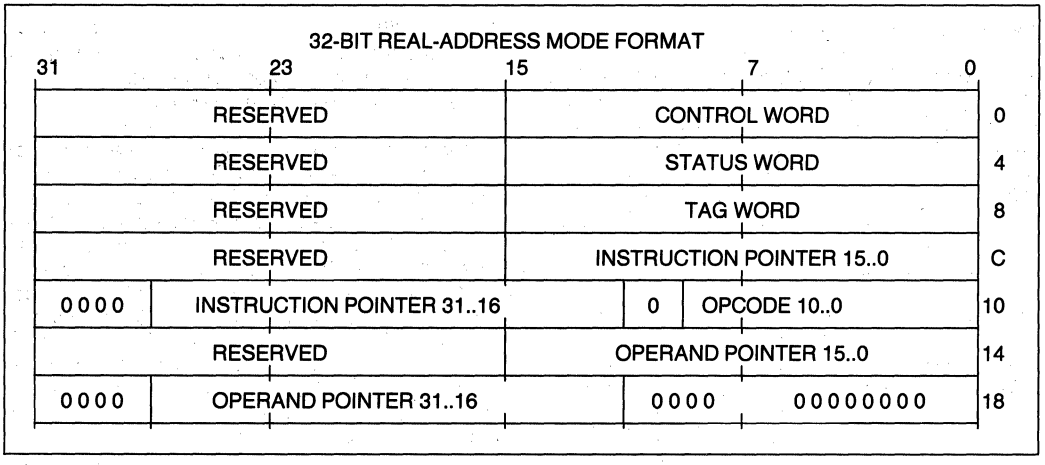


Figure 2.4. Real Mode 80387 Instruction and Data Pointer Image in Memory, 32-Bit Format

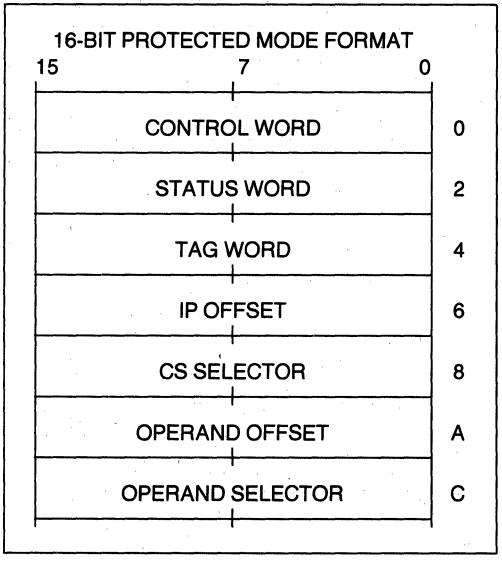


Figure 2.5. Protected Mode 80387 Instruction and Data Pointer Image in Memory, 16-Bit Format

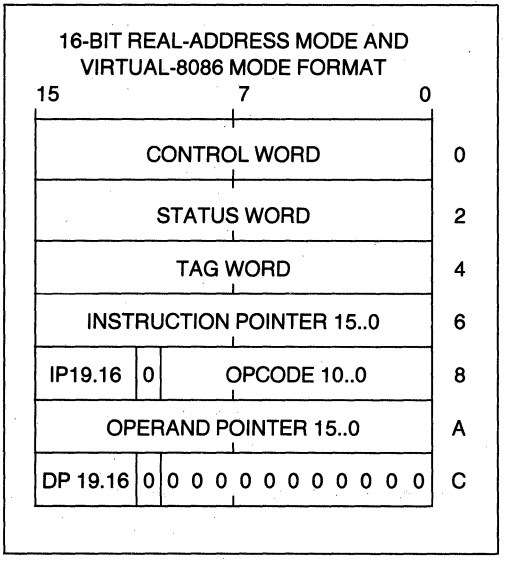


Figure 2.6. Real Mode 80387 Instruction and Data Pointer Image in Memory, 16-Bit Format

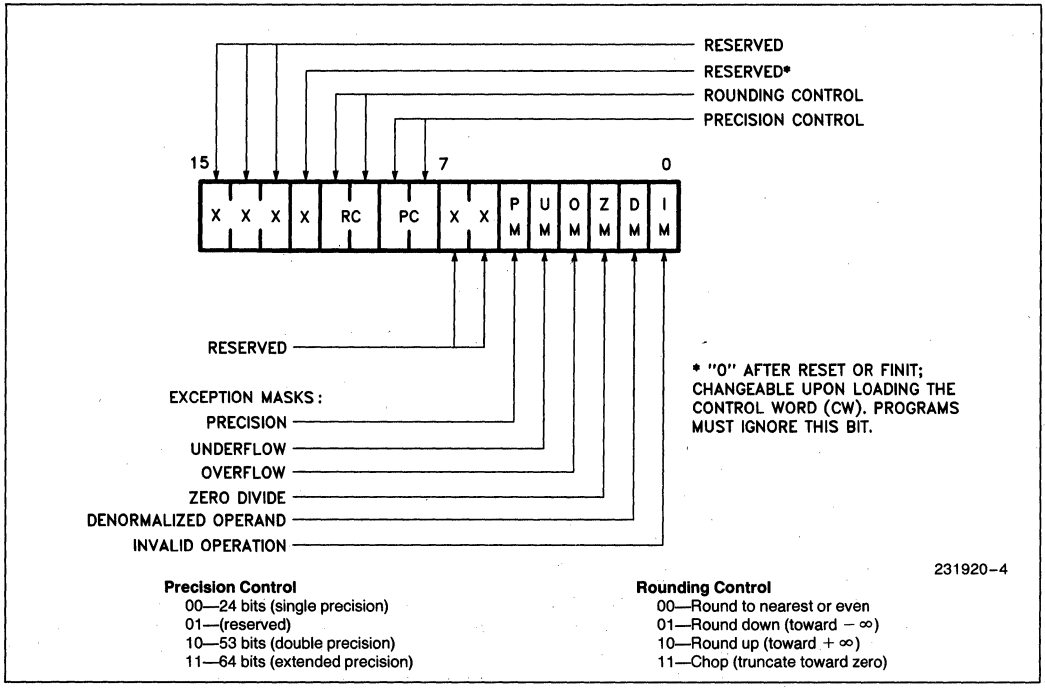


Figure 2.7. 80387 Control Word

2.3.5 CONTROL WORD

The NPX provides several processing options that are selected by loading a control word from memory into the control register. Figure 2.7 shows the format and encoding of fields in the control word.

The low-order byte of this control word configures the 80387 error and exception masking. Bits 5–0 of the control word contain individual masks for each of the six exceptions that the 80387 recognizes.

The high-order byte of the control word configures the 80387 operating mode, including precision and rounding.

- Bit 12 no longer defines infinity control and is a reserved bit. Only affine closure is supported for infinity arithmetic. The bit is initialized to zero after RESET or FINIT and is changeable upon loading the CW. Programs must ignore this bit.
- The rounding control (RC) bits (bits 11–10) provide for directed rounding and true chop, as well as the unbiased round to nearest even mode specified in the IEEE standard. Rounding control affects only those instructions that perform rounding at the end of the operation (and thus can generate a precision exception); namely, FST, FSTP, FIST, all arithmetic instructions (except FPREM, FPREM1, FXTRACT, FABS, and FCHS), and all transcendental instructions.
- The precision control (PC) bits (bits 9–8) can be used to set the 80387 internal operating precision of the significand at less than the default of 64 bits (extended precision). This can be useful in providing compatibility with early generation arithmetic processors of smaller precision. PC affects only the instructions ADD, SUB, DIV, MUL, and SQRT. For all other instructions, either the precision is determined by the opcode or extended precision is used.

2.4 Interrupt Description

Several interrupts of the 386 CPU are used to report exceptional conditions while executing numeric programs in either real or protected mode. Table 2.6 shows these interrupts and their causes.

Table 2.6. 386™ Microprocessor Interrupt Vectors Reserved for NPX

Interrupt Number	Cause of Interrupt
7	An ESC instruction was encountered when EM or TS of the 386™ CPU control register zero (CR0) was set. EM = 1 indicates that software emulation of the instruction is required. When TS is set, either an ESC or WAIT instruction causes interrupt 7. This indicates that the current NPX context may not belong to the current task.
9	An operand of a coprocessor instruction wrapped around an addressing limit (0FFFFH for small segments, 0FFFFFFFFH for big segments, zero for expand-down segments) and spanned inaccessible addresses[a]. The failing numerics instruction is not restartable. The address of the failing numerics instruction and data operand may be lost; an FSTENV does not return reliable addresses. As with the 80286/80287, the segment overrun exception should be handled by executing an FNINIT instruction (i.e. an FINIT without a preceding WAIT). The return address on the stack does not necessarily point to the failing instruction nor to the following instruction. The interrupt can be avoided by never allowing numeric data to start within 108 bytes of the end of a segment.
13	The first word or doubleword of a numeric operand is not entirely within the limit of its segment. The return address pushed onto the stack of the exception handler points at the ESC instruction that caused the exception, including any prefixes. The 80387 has not executed this instruction; the instruction pointer and data pointer register refer to a previous, correctly executed instruction.
16	The previous numerics instruction caused an unmasked exception. The address of the faulty instruction and the address of its operand are stored in the instruction pointer and data pointer registers. Only ESC and WAIT instructions can cause this interrupt. The 386™ CPU return address pushed onto the stack of the exception handler points to a WAIT or ESC instruction (including prefixes). This instruction can be restarted after clearing the exception condition in the NPX. FNINIT, FNCLEX, FNSTSW, FNSTENV, and FNSAVE cannot cause this interrupt.

a. An operand may wrap around an addressing limit when the segment limit is near an addressing limit and the operand is near the largest valid address in the segment. Because of the wrap-around, the beginning and ending addresses of such an operand will be at opposite ends of the segment. There are two ways that such an operand may also span inaccessible addresses: 1) if the segment limit is not equal to the addressing limit (e.g. addressing limit is FFFFH and segment limit is FFFDH) the operand will span addresses that are not within the segment (e.g. an 8-byte operand that starts at valid offset FFFC will span addresses FFFC–FFFF and 0000-0003; however addresses FFFE and FFFF are not valid, because they exceed the limit); 2) if the operand begins and ends in present and accessible pages but intermediate bytes of the operand fall in a not-present page or a page to which the procedure does not have access rights.

2.5 Exception Handling

The 80387 detects six different exception conditions that can occur during instruction execution. Table 2.7 lists the exception conditions in order of precedence, showing for each the cause and the default action taken by the 80387 if the exception is masked by its corresponding mask bit in the control word.

Any exception that is not masked by the control word sets the corresponding exception flag of the status word, sets the ES bit of the status word, and asserts the ERROR# signal. When the CPU attempts to execute another ESC instruction or WAIT, exception 7 occurs. The exception condition must be resolved via an interrupt service routine. The 386 Microprocessor and 80387 Coprocessor save the address of the floating-point instruction that caused the exception and the address of any memory operand required by that instruction.

2.6 Initialization

80387 initialization software must execute an FNINIT instruction (i.e. an FINIT without a preceding WAIT) to clear ERROR#. After a hardwareRESET, the ERROR# output is asserted to indicate that an 80387 is present. To accomplish this, the IE and ES bits of the status word are set, and the IM bit in the control word is reset. After FNINIT, the status word and the control word have the same values as in an 80287 after RESET.

2.7 8087 and 80287 Compatibility

This section summarizes the differences between the 80387 and the 80287. Any migration from the 8087 directly to the 80387 must also take into account the differences between the 8087 and the 80287 as listed in Appendix A.

Many changes have been designed into the 80387 to directly support the IEEE standard in hardware. These changes result in increased performance by eliminating the need for software that supports the standard.

2.7.1 GENERAL DIFFERENCES

The 80387 supports only affine closure for infinity arithmetic, not projective closure. Bit 12 of the Control Word (CW) no longer defines infinity control. It is a reserved bit; but it is initialized to zero after RESET or FINIT and is changeable upon loading the CW. Programs must ignore this bit.

Operands for FSCALE and FPATAN are no longer restricted in range (except for $\pm\infty$); F2XM1 and FPTAN accept a wider range of operands.

The results of transcendental operations may be slightly different from those computed by 80287.

In the case of FPTAN, the 80387 supplies a true tangent result in ST(1), and (always) a floating point 1 in ST.

Rounding control is in effect for FLD *constant*.

Software cannot change entries of the tag word to values (other than empty) that do not reflect the actual register contents.

After reset, FINIT, and incomplete FPREM, the 80387 resets to zero the condition code bits C_3–C_0 of the status word.

In conformance with the IEEE standard, the 80387 does not support the special data formats: pseudozero, pseudo-NaN, pseudoinfinity, and unnormal.

Table 2.7. Exceptions

Exception	Cause	Default Action (if exception is masked)
Invalid Operation	Operation on a signaling NaN, unsupported format, indeterminate form ($0*\infty$, 0/0, $(+\infty) + (-\infty)$, etc.), or stack overflow/underflow (SF is also set).	Result is a quiet NaN, integer indefinite, or BCD indefinite
Denormalized Operand	At least one of the operands is denormalized, i.e. it has the smallest exponent but a nonzero significand.	Normal processing continues
Zero Divisor	The divisor is zero while the dividend is a noninfinite, nonzero number.	Result is ∞
Overflow	The result is too large in magnitude to fit in the specified format.	Result is largest finite value or ∞
Underflow	The true result is nonzero but too small to be represented in the specified format, and, if underflow exception is masked, denormalization causes loss of accuracy.	Result is denormalized or zero
Inexact Result (Precision)	The true result is not exactly representable in the specified format (e.g. 1/3); the result is rounded according to the rounding mode.	Normal processing continues

2.7.2 EXCEPTIONS

A number of differences exist due to changes in the IEEE standard and to functional improvements to the architecture of the 80387:

1. When the overflow or underflow exception is masked, the 80387 differs from the 80287 in rounding when overflow or underflow occurs. The 80387 produces results that are consistent with the rounding mode.
2. When the underflow exception is masked, the 80387 sets its underflow flag only if there is also a loss of accuracy during denormalization.
3. Fewer invalid-operation exceptions due to denormal operands, because the instructions FSQRT, FDIV, FPREM, and conversions to BCD or to integer normalize denormal operands before proceeding.
4. The FSQRT, FBSTP, and FPREM instructions may cause underflow, because they support denormal operands.
5. The denormal exception can occur during the transcendental instructions and the FXTRACT instruction.
6. The denormal exception no longer takes precedence over all other exceptions.
7. When the denormal exception is masked, the 80387 automatically normalizes denormal operands. The 8087/80287 performs unnormal arithmetic, which might produce an unnormal result.
8. When the operand is zero, the FXTRACT instruction reports a zero-divide exception and leaves $-\infty$ in ST(1).
9. The status word has a new bit (SF) that signals when invalid-operation exceptions are due to stack underflow or overflow.
10. FLD *extended precision* no longer reports denormal exceptions, because the instruction is not numeric.
11. FLD *single/double precision* when the operand is denormal converts the number to extended precision and signals the denormalized operand exception. When loading a signaling NaN, FLD *single/double precision* signals an invalid-operand exception.
12. The 80387 only generates quiet NaNs (as on the 80287); however, the 80387 distinguishes between quiet NaNs and signaling NaNs. Signaling NaNs trigger exceptions when they are used as operands; quiet NaNs do not (except for FCOM, FIST, and FBSTP which also raise IE for quiet NaNs).
13. When stack overflow occurs during FPTAN and overflow is masked, both ST(0) and ST(1) contain quiet NaNs. The 80287/8087 leaves the original operand in ST(1) intact.
14. When the scaling factor is $\pm\infty$, the FSCALE (ST(0), ST(1)) instruction behaves as follows (ST(0) and ST(1) contain the scaled and scaling operands respectively):
    - FSCALE(0, ∞) generates the invalid operation exception.
    - FSCALE(finite, $-\infty$) generates zero with the same sign as the scaled operand.
    - FSCALE(finite, $+\infty$) generates ∞ with the same sign as the scaled operand.

    The 8087/80287 returns zero in the first case and raises the invalid-operation exception in the other cases.
15. The 80387 returns signed infinity/zero as the unmasked response to massive overflow/underflow. The 8087 and 80287 support a limited range for the scaling factor; within this range either massive overflow/underflow do not occur or undefined results are produced.

3.0 HARDWARE INTERFACE

In the following description of hardware interface, the # symbol at the end of a signal name indicates that the active or asserted state occurs when the signal is at a low voltage. When no # is present after the signal name, the signal is asserted when at the high voltage level.

3.1 Signal Description

In the following signal descriptions, the 80387 pins are grouped by function as follows:

1. Execution control—CPUCLK2, NUMCLK2, CKM, RESETIN
2. NPX handshake—PEREQ, BUSY#, ERROR#
3. Bus interface pins—D31–D0, W/R#, ADS#, READY#, READYO#
4. Chip/Port Select—STEN, NPS1#, NPS2, CMD0#
5. Power supplies—V_{CC}, V_{SS}

Table 3.1 lists every pin by its identifier, gives a brief description of its function, and lists some of its characteristics. All output signals are tristate; they leave floating state only when STEN is active. The output buffers of the bidirectional data pins D31–D0 are also tristate; they leave floating state only in read cycles when the 80387 is selected (i.e. when STEN, NPS1#, and NPS2 are all active).

Figure 3.1 and Table 3.2 together show the location of every pin in the pin grid array.

Table 3.1. 80387 Pin Summary

Pin Name	Function	Active State	Input/ Output	Referenced To
CPUCLK2	386™ CPU CLocK 2		I	
NUMCLK2	80387 CLocK 2		I	
CKM	80387 CLocKing Mode		I	
RESETIN	System reset	High	I	CPUCLK2
PEREQ	Processor Extension REQuest	High	O	CPUCLK2/STEN
BUSY#	Busy status	Low	O	CPUCLK2/STEN
ERROR#	Error status	Low	O	NUMCLK2/STEN
D31–D0	Data pins	High	I/O	CPUCLK2
W/R#	Write/Read bus cycle	Hi/Lo	I	CPUCLK2
ADS#	ADdress Strobe	Low	I	CPUCLK2
READY#	Bus ready input	Low	I	CPUCLK2
READYO#	Ready output	Low	O	CPUCLK2/STEN
STEN	STatus ENable	High	I	CPUCLK2
NPS1#	NPX select #1	Low	I	CPUCLK2
NPS2	NPX select #2	High	I	CPUCLK2
CMD0#	CoMmanD	Low	I	CPUCLK2
V_{CC}			I	
V_{SS}			I	

NOTE:
STEN is referenced to only when getting the output pins into or out of tristate mode.

Table 3.2. 80387 Pin Cross-Reference

A2	—	D9	C11	—	V_{SS}	J10	—	V_{SS}
A3	—	D11	D1	—	D5	J11	—	CKM
A4	—	D12	D2	—	D4	K1	—	PEREQ
A5	—	D14	D10	—	D24	K2	—	BUSY#
A6	—	V_{CC}	D11	—	D25	K3	—	Tie High
A7	—	D16	E1	—	V_{CC}	K4	—	W/R#
A8	—	D18	E2	—	V_{SS}	K5	—	V_{CC}
A9	—	V_{CC}	E10	—	D26	K6	—	NPS2
A10	—	D21	E11	—	D27	K7	—	ADS#
B1	—	D8	F1	—	V_{CC}	K8	—	READY#
B2	—	V_{SS}	F2	—	V_{SS}	K9	—	No Connect
B3	—	D10	F10	—	V_{CC}	K10	—	CPUCLK2
B4	—	V_{CC}	F11	—	V_{SS}	K11	—	NUMCLK2
B5	—	D13	G1	—	D3	L2	—	ERROR#
B6	—	D15	G2	—	D2	L3	—	READYO#
B7	—	V_{SS}	G10	—	D28	L4	—	STEN
B8	—	D17	G11	—	D29	L5	—	V_{SS}
B9	—	D19	H1	—	D1	L6	—	NPS1#
B10	—	D20	H2	—	D0	L7	—	V_{CC}
B11	—	D22	H10	—	D30	L8	—	CMD0#
C1	—	D7	H11	—	D31	L9	—	Tie High
C2	—	D6	J1	—	V_{SS}	L10	—	RESETIN
C10	—	D23	J2	—	V_{CC}			

3.1.1 386™ CPU CLOCK 2 (CPUCLK2)

This input uses the 386 CPU CLK2 signal to time the bus control logic. Several other 80387 signals are referenced to the rising edge of this signal. When CKM = 1 (synchronous mode) this pin also clocks the data interface and control unit and the floating-point unit of the 80387. This pin requires MOS-level input. The signal on this pin is divided by two to produce the internal clock signal CLK.

3.1.2 80387 CLOCK 2 (NUMCLK2)

When CKM = 0 (asynchronous mode) this pin provides the clock for the data interface and control unit and the floating-point unit of the 80387. In this case, the ratio of the frequency of NUMCLK2 to the frequency of CPUCLK2 must lie within the range 10:16 to 14:10. When CKM = 1 (synchronous mode) this pin is ignored; CPUCLK2 is used instead for the data interface and control unit and the floating-point unit. This pin requires TTL-level input.

3.1.3 80387 CLOCKING MODE (CKM)

This pin is a strapping option. When it is strapped to V_{CC}, the 80387 operates in synchronous mode; when strapped to V_{SS}, the 80387 operates in asynchronous mode. These modes relate to clocking of the data interface and control unit and the floating-point unit only; the bus control logic always operates synchronously with respect to the 386 Microprocessor.

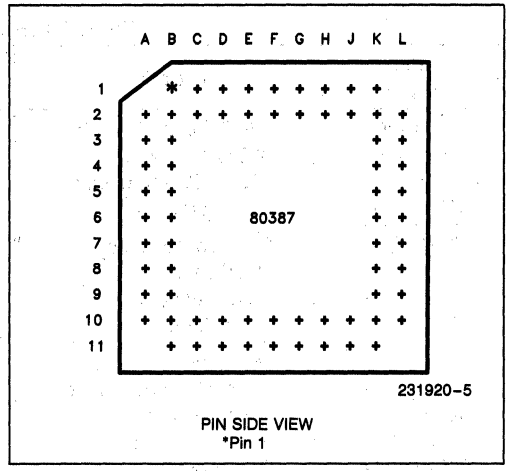


Figure 3.1. 80387 Pin Configuration

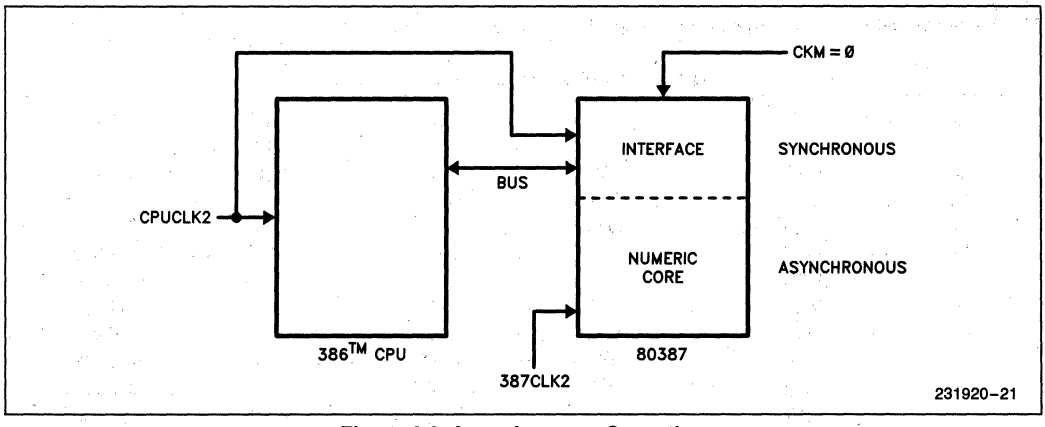


Figure 3.2. Asynchronous Operation

3.1.4 SYSTEM RESET (RESETIN)

A LOW to HIGH transition on this pin causes the 80387 to terminate its present activity and to enter a dormant state. RESETIN must remain HIGH for at least 40 NUMCLK2 periods. The HIGH to LOW transitions of RESETIN must be synchronous with CPUCLK2, so that the phase of the internal clock of the bus control logic (which is the CPUCLK2 divided by 2) is the same as the phase of the internal clock of the 386 CPU. After RESETIN goes LOW, at least 50 NUMCLK2 periods must pass before the first NPX instruction is written into the 80387. This pin should be connected to the 386 CPU RESET pin. Table 3.3 shows the status of other pins after a reset.

Table 3.3. Output Pin Status During Reset

Pin Value	Pin Name
HIGH	READYO#, BUSY#
LOW	PEREQ, ERROR#
Tri-State OFF	D31–D0

3.1.5 PROCESSOR EXTENSION REQUEST (PEREQ)

When active, this pin signals to the 386 CPU that the 80387 is ready for data transfer to/from its data FIFO. When all data is written to or read from the data FIFO, PEREQ is deactivated. This signal always goes inactive before BUSY# goes inactive. This signal is referenced to CPUCLK2. It should be connected to the 386 CPU PEREQ input. Refer to Figure 3.8 for the timing relationships between this and the BUSY# and ERROR# pins.

3.1.6 BUSY STATUS (BUSY#)

When active, this pin signals to the 386 CPU that the 80387 is currently executing an instruction. This signal is referenced to CPUCLK2. It should be connected to the 386 CPU BUSY# pin. Refer to Figure 3.8 for the timing relationships between this and the PEREQ and ERROR# pins.

3.1.7 ERROR STATUS (ERROR#)

This pin reflects the ES bits of the status register. When active, it indicates that an unmasked exception has occurred (except that, immediately after a reset, it indicates to the 386 Microprocessor that an 80387 is present in the system). This signal can be changed to inactive state only by the following instructions (without a preceding WAIT): FNINIT, FNCLEX, FNSTENV, and FNSAVE. This signal is referenced to NUMCLK2. It should be connected to the 386 CPU ERROR# pin. Refer to Figure 3.8 for the timing relationships between this and the PEREQ and BUSY# pins.

3.1.8 DATA PINS (D31–D0)

These bidirectional pins are used to transfer data and opcodes between the 386 CPU and 80387. They are normally connected directly to the corresponding 386 CPU data pins. HIGH state indicates a value of one. D0 is the least significant data bit. Timings are referenced to CPUCLK2.

3.1.9 WRITE/READ BUS CYCLE (W/R#)

This signal indicates to the 80387 whether the 386 CPU bus cycle in progress is a read or a write cycle. This pin should be connected directly to the 386 CPU W/R# pin. HIGH indicates a write cycle; LOW, a read cycle. This input is ignored if any of the signals STEN, NPS1#, or NPS2 is inactive. Setup and hold times are referenced to CPUCLK2.

3.1.10 ADDRESS STROBE (ADS#)

This input, in conjunction with the READY# input indicates when the 80387 bus-control logic may sample W/R# and the chip-select signals. Setup and hold times are referenced to CPUCLK2. This pin should be connected to the 386 CPU ADS# pin.

3.1.11 BUS READY INPUT (READY#)

This input indicates to the 80387 when a 386 CPU bus cycle is to be terminated. It is used by the bus-control logic to trace bus activities. Bus cycles can be extended indefinitely until terminated by READY#. This input should be connected to the same signal that drives the 386 CPU READY# input. Setup and hold times are referenced to CPUCLK2.

3.1.12 READY OUTPUT (READYO#)

This pin is activated at such a time that write cycles are terminated after two clocks and read cycles after three clocks. In configurations where no extra wait states are required, this pin must directly or indirectly drive the 386 CPU READY# input. Refer to section 3.4 "Bus Operation" for details. This pin is activated only during bus cycles that select the 80387. This signal is referenced to CPUCLK2.

3.1.13 STATUS ENABLE (STEN)

This pin serves as a chip select for the 80387. When inactive, this pin forces BUSY#, PEREQ, ERROR#, and READYO# outputs into floating state. D31–D0 are normally floating and leave floating state only if STEN is active and additional conditions are met. STEN also causes the chip to recognize its other chip-select inputs. STEN makes it easier to do on-board testing (using the overdrive method) of other chips in systems containing the 80387. STEN should be pulled up with a resistor so that it can be pulled down when testing. In boards that do not use on-board testing, STEN should be connected to V_{CC}. Setup and hold times are relative to CPUCLK2. Note that STEN must maintain the same setup and hold times as NPS1#, NPS2, and CMD0# (i.e. if STEN changes state during an 80387 bus cycle, it should change state during the same CLK period as the NPS1#, NPS2, and CMD0# signals).

3.1.14 NPX Select #1 (NPS1#)

When active (along with STEN and NPS2) in the first period of a 386 CPU bus cycle, this signal indicates that the purpose of the bus cycle is to communicate with the 80387. This pin should be connected directly to the 386 CPU M/IO# pin, so that the 80387 is selected only when the 386 CPU performs I/O cycles. Setup and hold times are referenced to CPUCLK2.

3.1.15 NPX SELECT #2 (NPS2)

When active (along with STEN and NPS1#) in the first period of an 386 CPU bus cycle, this signal indicates that the purpose of the bus cycle is to communicate with the 80387. This pin should be connected directly to the 386 CPU A31 pin, so that the 80387 is selected only when the 386 CPU uses one of the I/O addresses reserved for the 80387 (800000F8 or 800000FC). Setup and hold times are referenced to CPUCLK2.

3.1.16 COMMAND (CMD0#)

During a write cycle, this signal indicates whether an opcode (CMD0# active) or data (CMD0# inactive) is being sent to the 80387. During a read cycle, it indicates whether the control or status register (CMD0# active) or a data register (CMD0# inactive) is being read. CMD0# should be connected directly to the A2 output of the 386 Microprocessor. Setup and hold times are referenced to CPUCLK2.

3.2 Processor Architecture

As shown by the block diagram on the front page, the NPX is internally divided into three sections: the bus control logic (BCL), the data interface and control unit, and the floating point unit (FPU). The FPU (with the support of the control unit which contains the sequencer and other support units) executes all numerics instructions. The data interface and control unit is responsible for the data flow to and from the FPU and the control registers, for receiving the instructions, decoding them, and sequencing the microinstructions, and for handling some of the administrative instructions. The BCL is responsible for the 386 CPU bus tracking and interface. The BCL is the only unit in the 80387 that must run synchronously with the 386 CPU; the rest of the 80387 can run asynchronously with respect to the 386 Microprocessor.

3.2.1 BUS CONTROL LOGIC

The BCL communicates solely with the CPU using I/O bus cycles. The BCL appears to the CPU as a special peripheral device. It is special in two respects: the CPU initiates I/O automatically when it encounters ESC instructions, and the CPU uses reserved I/O addresses to communicate with the BCL. The BCL does not communicate directly with memory. The CPU performs all memory access, transferring input operands from memory to the 80387 and transferring outputs from the 80387 to memory.

3.2.2 DATA INTERFACE AND CONTROL UNIT

The data interface and control unit latches the data and, subject to BCL control, directs the data to the FIFO or the instruction decoder. The instruction decoder decodes the ESC instructions sent to it by the CPU and generates controls that direct the data flow in the FIFO. It also triggers the microinstruction sequencer that controls execution of each instruction. If the ESC instruction is FINIT, FCLEX, FSTSW, FSTSW AX, or FSTCW, the control executes it independently of the FPU and the sequencer. The data interface and control unit is the one that generates the BUSY#, PEREQ and ERROR# signals that synchronize 80387 activities with the 386 CPU. It also supports the FPU in all operations that it cannot perform alone (e.g. exceptions handling, transcendental operations, etc.).

3.2.3 FLOATING POINT UNIT

The FPU executes all instructions that involve the register stack, including arithmetic, logical, transcendental, constant, and data transfer instructions. The data path in the FPU is 84 bits wide (68 significant bits, 15 exponent bits, and a sign bit) which allows internal operand transfers to be performed at very high speeds.

3.3 System Configuration

As an extension to the 386 Microprocessor, the 80387 can be connected to the CPU as shown by Figure 3.3. A dedicated communication protocol

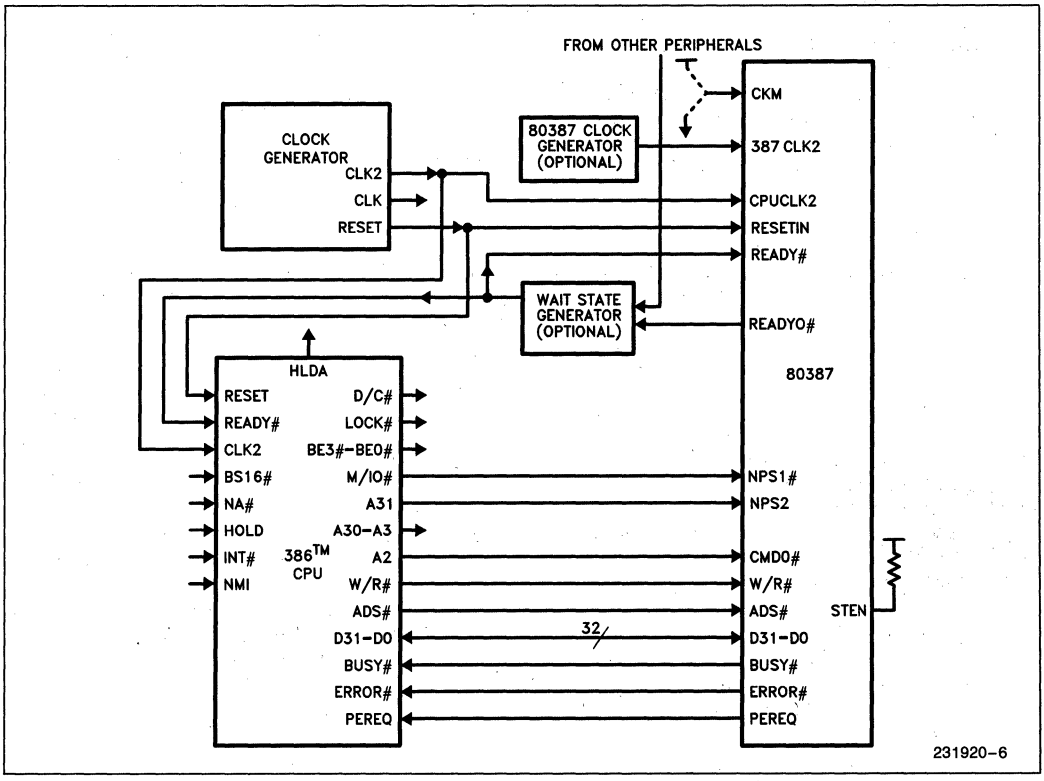


Figure 3.3. 386™ Microprocessor and 80387 Coprocessor System Configuration

Table 3.4. Bus Cycles Definition

STEN	NPS1#	NPS2	CMD0#	W/R#	Bus Cycle Type
0	x	x	x	x	80387 not selected and all outputs in floating state
1	1	x	x	x	80387 not selected
1	x	0	x	x	80387 not selected
1	0	1	0	0	CW or SW read from 80387
1	0	1	0	1	Opcode write to 80387
1	0	1	1	0	Data read from 80387
1	0	1	1	1	Data write to 80387

makes possible high-speed transfer of opcodes and operands between the 386 CPU and 80387. The 80387 is designed so that no additional components are required for interface with the 386 CPU. The 80387 shares the 32-bit wide local bus of the 386 CPU and most control pins of the 80387 are connected directly to pins of the 386 Microprocessor.

3.3.1 BUS CYCLE TRACKING

The ADS# and READY# signals allow the 80387 to track the beginning and end of the 386 CPU bus cycles, respectively. When ADS# is asserted at the same time as the 80387 chip-select inputs, the bus cycle is intended for the 80387. To signal the end of a bus cycle for the 80387, READY# may be asserted directly or indirectly by the 80387 or by other bus-control logic. Refer to Table 3.4 for definition of the types of 80387 bus cycles.

3.3.2 80387 ADDRESSING

The NPS1#, NPS2 and STEN signals allow the NPX to identify which bus cycles are intended for the NPX. The NPX responds only to I/O cycles when bit 31 of the I/O address is set. In other words, the NPX acts as an I/O device in a reserved I/O address space.

Because A_{31} is used to select the 80387 for data transfers, it is not possible for a program running on the 386 CPU to address the 80387 with an I/O instruction. Only ESC instructions cause the 386 Microprocessor to communicate with the 80387. The 386 CPU BS16# input must be inactive during I/O cycles when A_{31} is active.

3.3.3 FUNCTION SELECT

The CMD0# and W/R# signals identify the four kinds of bus cycle: control or status register read, data read, opcode write, data write.

3.3.4 CPU/NPX Synchronization

The pin pairs BUSY#, PEREQ, and ERROR# are used for various aspects of synchronization between the CPU and the NPX.

BUSY# is used to synchronize instruction transfer from the 386 CPU to the 80387. When the 80387 recognizes an ESC instruction, it asserts BUSY#. For most ESC instructions, the 386 CPU waits for the 80387 to deassert BUSY# before sending the new opcode.

The NPX uses the PEREQ pin of the 386 CPU to signal that the NPX is ready for data transfer to or from its data FIFO. The NPX does not directly access memory; rather, the 386 Microprocessor provides memory access services for the NPX. Thus, memory access on behalf of the NPX always obeys the rules applicable to the mode of the 386 CPU, whether the 386 CPU be in real-address mode or protected mode.

Once the 386 CPU initiates an 80387 instruction that has operands, the 386 CPU waits for PEREQ signals that indicate when the 80387 is ready for operand transfer. Once all operands have been transferred (or if the instruction has no operands) the 386 CPU continues program execution while the 80387 executes the ESC instruction.

In 8086/8087 systems, WAIT instructions may be required to achieve synchronization of both commands and operands. In 80286/80287, 386 Microprocessor and 80387 Coprocessor systems, WAIT instructions are required only for operand synchronization; namely, after NPX stores to memory (except FSTSW and FSTCW) or loads from memory. Used this way, WAIT ensures that the value has already been written or read by the NPX before the CPU reads or changes the value.

Once it has started to execute a numerics instruction and has transferred the operands from the 386 CPU, the 80387 can process the instruction in parallel with and independent of the host CPU. When the NPX detects an exception, it asserts the ERROR# signal, which causes a 386 CPU interrupt.

3.3.5 SYNCHRONOUS OR ASYNCHRONOUS MODES

The internal logic of the 80387 (the FPU) can either operate directly from the CPU clock (synchronous mode) or from a separate clock (asynchronous mode). The two configurations are distinguished by the CKM pin. In either case, the bus control logic (BCL) of the 80387 is synchronized with the CPU clock. Use of asynchronous mode allows the 386 CPU and the FPU section of the 80387 to run at different speeds. In this case, the ratio of the frequency of NUMCLK2 to the frequency of CPUCLK2 must lie within the range 10:16 to 14:10. Use of synchronous mode eliminates one clock generator from the board design.

3.3.6 AUTOMATIC BUS CYCLE TERMINATION

In configurations where no extra wait states are required, READYO# can be used to drive the 386 CPU READY# input. If this pin is used, it should be connected to the logic that ORs all READY outputs from peripherals on the 386 CPU bus. READYO# is asserted by the 80387 only during I/O cycles that select the 80387. Refer to section 3.4 "Bus Operation" for details.

3.4 Bus Operation

With respect to the bus interface, the 80387 is fully synchronous with the 386 Microprocessor. Both operate at the same rate, because each generates its internal CLK signal by dividing CPUCLK2 by two.

The 386 CPU initiates a new bus cycle by activating ADS#. The 80387 recognizes a bus cycle, if, during the cycle in which ADS# is activated, STEN, NPS1#, and NPS2 are all activated. Proper operation is achieved if NPS1# is connected to the M/IO# output of the 386 CPU, and NPS2 to the A31 output. The 386 CPU's A31 output is guaranteed to be inactive in all bus cycles that do not address the 80387 (i.e. I/O cycles to other devices, interrupt acknowledge, and reserved types of bus cycles). System logic must not signal a 16-bit bus cycle via the 386 CPU BS16# input during I/O cycles when A31 is active.

During the CLK period in which ADS# is activated, the 80387 also examines the W/R# input signal to determine whether the cycle is a read or a write cycle and examines the CMD0# input to determine whether an opcode, operand, or control/status register transfer is to occur.

The 80387 supports both pipelined and nonpipelined bus cycles. A nonpipelined cycle is one for which the 386 CPU asserts ADS# when no other 80387 bus cycle is in progress. A pipelined bus cycle is one for which the 386 CPU asserts ADS# and provides valid next-address and control signals as soon as in the second CLK period after the ADS# assertion for the previous 386 CPU bus cycle. Pipelining increases the availability of the bus by at least one CLK period. The 80387 supports pipelined bus cycles in order to optimize address pipelining by the 386 CPU for memory cycles.

Bus operation is described in terms of an abstract *state machine*. Figure 3.4 illustrates the states and state transitions for 80387 bus cycles:

- T_I is the idle state. This is the state of the bus logic after RESET, the state to which bus logic returns after evey nonpipelined bus cycle, and the state to which bus logic returns after a series of pipelined cycles.
- T_{RS} is the READY# sensitive state. Different types of bus cycle may require a minimum of one or two successive T_{RS} states. The bus logic remains in T_{RS} state until READY# is sensed, at which point the bus cycle terminates. Any number of wait states may be implemented by delaying READY#, thereby causing additional successive T_{RS} states.
- T_P is the first state for every pipelined bus cycle.

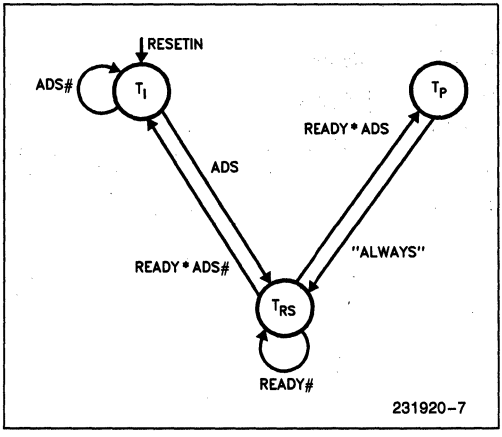


Figure 3.4. Bus State Diagram

The READYO# output of the 80387 indicates when a bus cycle for the 80387 may be terminated if no extra wait states are required. For all write cycles (except those for the instructions FLDENV and FRSTOR), READYO# is always asserted in the first T_{RS} state, regardless of the number of wait states. For all read cycles and write cycles for FLDENV and FRSTOR, READYO# is always asserted in the second T_{RS} state, regardless of the number of wait states. These rules apply to both pipelined and nonpipelined cycles. Systems designers must use READYO# in one of the following ways:

1. Connect it (directly or through logic that ORs READY signals from other devices) to the READY# inputs of the 386 CPU and 80387.
2. Use it as one input to a wait-state generator.

The following sections illustrate different types of 80387 bus cycles.

Because different instructions have different amounts of overhead before, between, and after operand transfer cycles, it is not possible to represent in a few diagrams all of the combinations of successive operand transfer cycles. The following bus-cycle diagrams show memory cycles between 80387 operand-transfer cycles. Note however that, during the instructions FLDENV, FSTENV, FSAVE, and FRSTOR, some consecutive accesses to the NPX do not have intervening memory accesses. For the timing relationship between operand transfer cycles and opcode write or other overhead activities, see Figure 3.8.

3.4.1 NONPIPELINED BUS CYCLES

Figure 3.5 illustrates bus activity for consecutive nonpipelined bus cycles.

3.4.1.1 Write Cycle

At the second clock of the bus cycle, the 80387 enters the T_{RS} (READY#-sensitive) state. During this state, the 80387 samples the READY# input and stays in this state as long as READY# is inactive.

In write cycles, the 80387 drives the READYO# signal for one CLK period beginning with the second CLK of the bus cycle; therefore, the fastest write cycle takes two CLK cycles (see cycle 2 of Figure 3.5). For the instructions FLDENV and FRSTOR, however, the 80387 forces a wait state by delaying the activation of READYO# to the second T_{RS} cycle (not shown in Figure 3.5).

When READY# is asserted the 80387 returns to the idle state, in which ADS# could be asserted again by the 386 Microprocessor for the next cycle.

3.4.1.2 Read Cycle

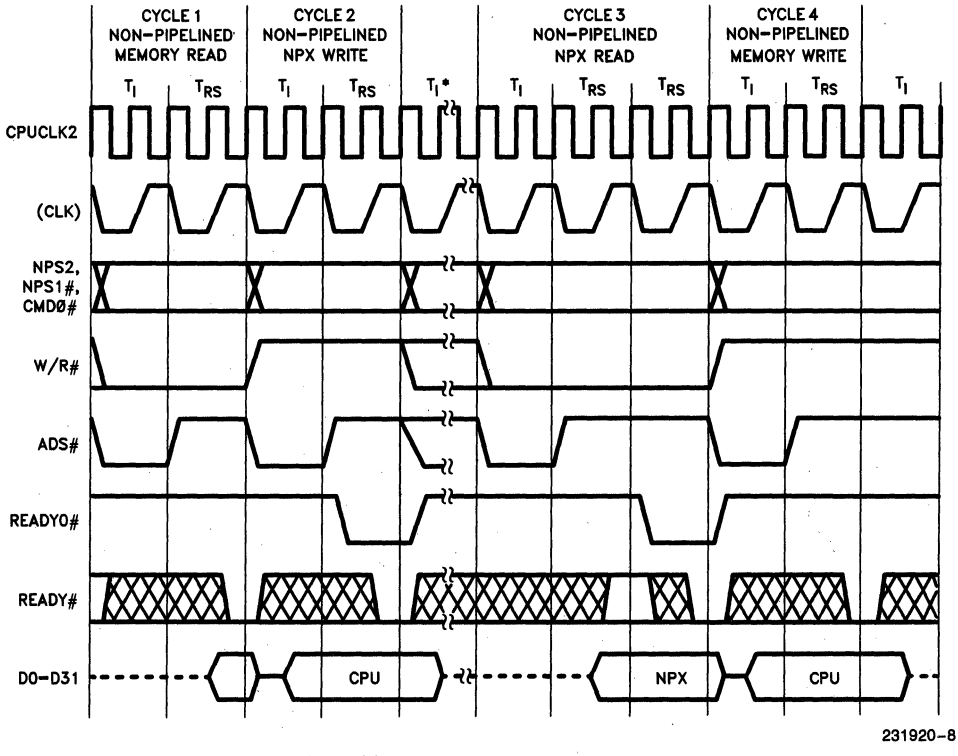
At the second clock of the bus cycle, the 80387 enters the T_{RS} state. See Figure 3.5. In this state, the 80387 samples the READY# input and stays in this state as long as READY# is inactive.

At the rising edge of CLK in the second clock period of the cycle, the 80387 starts to drive the D31–D0 outputs and continues to drive them as long as it stays in T_{RS} state.

In read cycles that address the 80387, at least one wait state must be inserted to insure that the 386 CPU latches the correct data. Since the 80387 starts driving the system data bus only at the rising edge of CLK in the second clock period of the bus cycle, not enough time is left for the data signals to propagate and be latched by the 386 CPU at the falling edge of the same clock period. The 80387 drives the READYO# signal for one CLK period in the third CLK of the bus cycle. Therefore, if the READYO# output is used to drive the 386 CPU READY# input, one wait state is inserted automatically.

Because one wait state is required for 80387 reads, the minimum is three CLK cycles per read, as cycle 3 of Figure 3.5 shows.

When READY# is asserted the 80387 returns to the idle state, in which ADS# could be asserted again by the 386 CPU for the next cycle. The transition from T_{RS} state to idle state causes the 80387 to put the tristate D31–D0 outputs into the floating state, allowing another device to drive the system data bus.



Cycles 1 & 2 represent part of the operand transfer cycle for instructions involving either 4-byte or 8-byte operand loads.
Cycles 3 & 4 represent part of the operand transfer cycle for a store operation.
*Cycles 1 & 2 could repeat here or T_I states for various non-operand transfer cycles and overhead.

Figure 3.5. Nonpipelined Read and Write Cycles

3.4.2 PIPELINED BUS CYCLES

Because all the activities of the 80387 bus interface occur either during the T_{RS} state or during the transitions to or from that state, the only difference between a pipelined and a nonpipelined cycle is the manner of changing from one state to another. The exact activities in each state are detailed in the previous section "Nonpipelined Bus Cycles".

When the 386 CPU asserts ADS# before the end of a bus cycle, both ADS# and READY# are active during a T_{RS} state. This condition causes the 80387 to change to a different state named T_P. The 80387 activities in the transition from a T_{RS} state to a T_P state are exactly the same as those in the transition from a T_{RS} state to a T_I state in nonpipelined cycles. T_P state is metastable; therefore, one clock period later the 80387 returns to T_{RS} state. In consecutive pipelined cycles, the 80387 bus logic uses only T_{RS} and T_P states.

Figure 3.6 shows the fastest transition into and out of the pipelined bus cycles. Cycle 1 in this figure represents a nonpipelined cycle. (Nonpipelined write cycles with only one T_{RS} state (i.e. no wait states) are always followed by another nonpipelined cycle, because READY# is asserted before the earliest possible assertion of ADS# for the next cycle.)

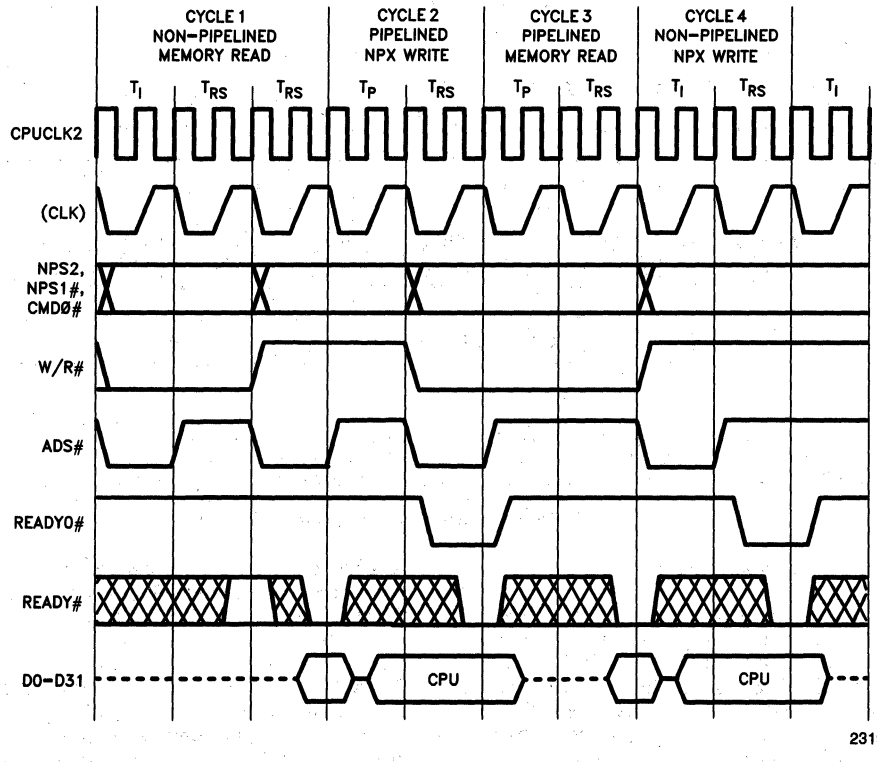
Figure 3.7 shows the pipelined write and read cycles with one additional T_{RS} states beyond the minimum required. To delay the assertion of READY# requires external logic.

3.4.3 BUS CYCLES OF MIXED TYPE

When the 80387 bus logic is in the T_{RS} state, it distinguishes between nonpipelined and pipelined cycles according to the behavior of ADS# and READY#. In a nonpipelined cycle, only READY# is activated, and the transition is from T_{RS} to idle state. In a pipelined cycle, both READY# and ADS# are active and the transition is first from T_{RS} state to T_P state then, after one clock period, back to T_{RS} state.

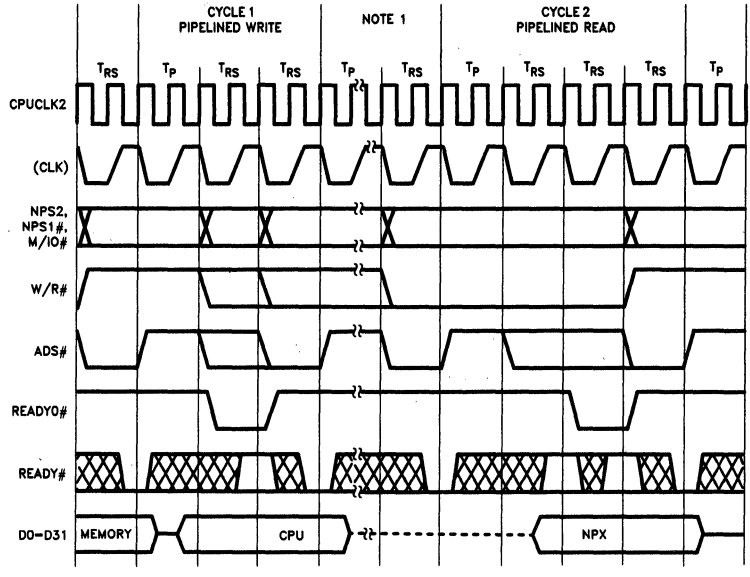
3.4.4 BUSY# AND PEREQ TIMING RELATIONSHIP

Figure 3.8 shows the activation of BUSY# at the beginning of instruction execution and its deactivation after execution of the instruction is complete. PEREQ is activated in this interval. If ERROR# (not shown in the diagram) is ever asserted, it would occur at least six CPUCLK2 periods after the deactivation of PEREQ and at least six CPUCLK2 periods before the deactivation of BUSY#. Figure 3.8 shows also that STEN is activated at the beginning of a bus cycle.



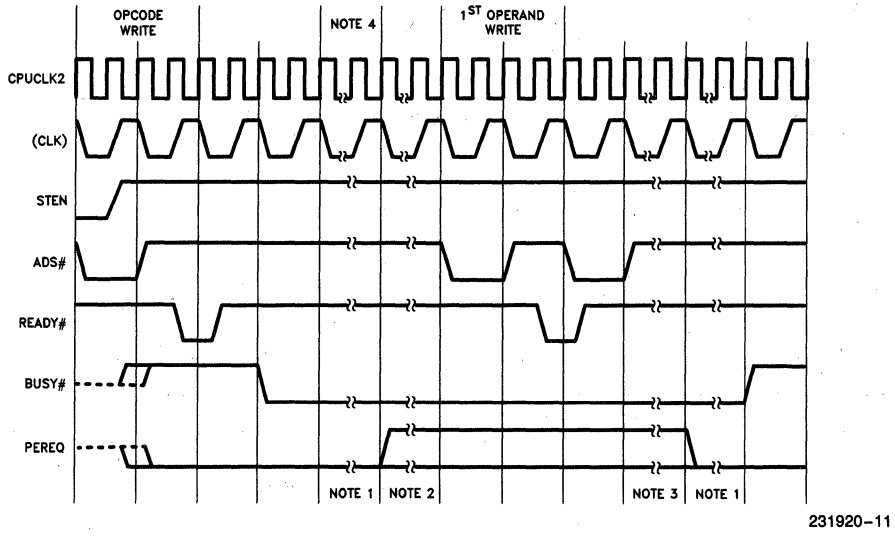
Cycle 1–Cycle 4 represent the operand transfer cycle for an instruction involving a transfer of two 32-bit loads in total. The opcode write cycles and other overhead are not shown.
Note that the next cycle will be a pipelined cycle if both READY# and ADS# are sampled active at the end of a T_{RS} state of the current cycle.

Figure 3.6. Fastest Transitions to and from Pipelined Cycles



NOTE:
1. Cycles between operand write to the NPX and storing result.

Figure 3.7. Pipelined Cycles with Wait States

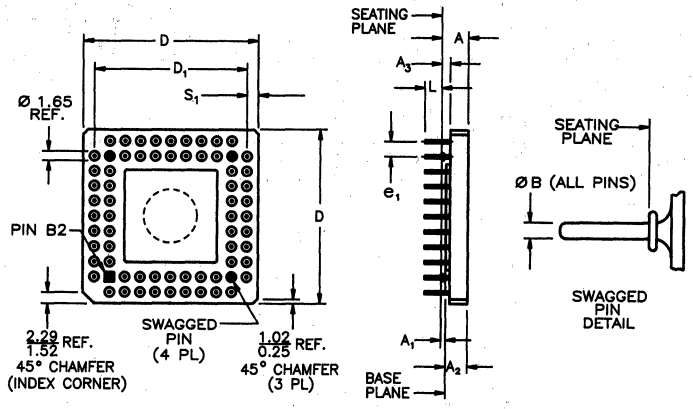


NOTES:
1. Instruction dependent.
2. PEREQ is an asynchronous input to the 386™ Microprocessor; it may not be asserted (instruction dependent).
3. More operand transfers.
4. Memory read (operand) cycle is not shown.

Figure 3.8. STEN, BUSY# and PEREQ Timing Relationship

4.0 MECHANICAL DATA

68 LEAD CERAMIC PIN GRID ARRAY PACKAGE INTEL TYPE A



231920-12

Family: Ceramic Pin Grid Array Package						
Symbol	**Millimeters**			**Inches**		
	Min	**Max**	**Notes**	**Min**	**Max**	**Notes**
A	3.56	4.57		0.140	0.180	
A_1	0.76	1.27	Solid Lid	0.030	0.050	Solid Lid
A_1		0.41	EPROM Lid		0.016	EPROM Lid
A_2	2.72	3.43	Solid Lid	0.107	0.135	Solid Lid
A_2	3.43	4.32	EPROM Lid	0.135	0.170	EPROM Lid
A_3	1.14	1.40		0.045	0.055	
B	0.43	0.51		0.017	0.020	
D	28.83	29.59		1.135	1.165	
D_1	25.27	25.53		0.995	1.005	
e_1	2.29	2.79		0.090	0.110	
L	2.29	3.30		0.090	0.130	
N	68			68		
S_1	1.27	2.54		0.050	0.100	
ISSUE	IWS REV 7 3/26/86					

Figure 4.1. Package Description

5.0 ELECTRICAL DATA

5.1 Absolute Maximum Ratings*

Case Temperature T_C
Under Bias ..... −65°C to +110°C
Storage Temperature ..... −65°C to +150°C
Voltage on Any Pin with
Respect to Ground ..... −0.5 to V_{CC} +0.5V
Power Dissipation 1.5W

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

5.2 D.C. Characteristics

Table 5.1. DC Specifications T_C = 0° to 80°C, V_{CC} = 5V ± 5%

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input LO Voltage	−0.3	+0.8	V	(Note 1)
V_{IH}	Input HI Voltage	2.0	V_{CC} + 0.3	V	(Note 1)
V_{CL}	CPUCLK2 Input LO Voltage	−0.3	+0.8	V	
V_{CH}	CPUCLK2 Input HI Voltage	3.7	V_{CC} +0.3	V	
V_{OL}	Output LO Voltage		0.45	V	(Note 2)
V_{OH}	Output HI Voltage	2.4		V	(Note 3)
I_{CC}	Supply Current				
	NUMCLK2 = 32 MHz(4)		250	mA	I_{CC} typ. = 150 mA
	NUMCLK2 = 40 MHz(4)		310	mA	I_{CC} typ. = 190 mA
	NUMCLK2 = 50 MHz(4)		390	mA	I_{CC} typ. = 250 mA
I_{LI}	Input Leakage Current		±15	μA	$0V \le V_{IN} \le V_{CC}$
I_{LO}	I/O Leakage Current		±15	μA	$0.45V \le V_O \le V_{CC}$
C_{IN}	Input Capacitance		10	pF	fc = 1 MHz
C_O	I/O or Output Capacitance		12	pF	fc = 1 MHz
C_{CLK}	Clock Capacitance		20	pF	fc = 1 MHz

NOTES:
1. This parameter is for all inputs, including NUMCLK2 but excluding 386CLK2.
2. This parameter is measured at I_{OL} as follows:
   data = 4.0 mA
   READYO# = 2.5 mA
   ERROR#, BUSY#, PEREQ = 2.5 mA
3. This parameter is measured at I_{OH} as follows:
   data = 1.0 mA
   READYO# = 0.6 mA
   ERROR#, BUSY#, PEREQ = 0.6 mA
4. I_{CC} is measured at steady state, maximum capacitive loading on the outputs, and worst-case DC level at the inputs; 386CLK2 at the same frequency as NUMCLK2.

5.3 A.C. Characteristics

Table 5.2a. Combinations of Bus Interface and Execution Speeds

Speed Combinations			
Functional Block	**80387-16**	**80387-20**	**80387-25**
Bus Interface Unit (MHz)	16	20	25
Execution Unit (MHz)	16	20	25

Table 5.2b. Timing Requirements of the Execution Unit
T_C = 0°C to +80°C, V_{CC} = 5V ±5%

Pin	Symbol	Parameter	16 MHz 1.5V		20 MHz 1.5V		25 MHz 1.5V Preliminary		Test Conditions	Figure Reference
			Min (ns)	Max (ns)	Min (ns)	Max (ns)	Min (ns)	Max (ns)		
NUMCLK2	t1	Period	31.25	125	25	125	20	125	2.0V	5.1
NUMCLK2	t2a	High Time	9		8		7		2.0V	
NUMCLK2	t2b	High Time	5		5		4		3.7V	
NUMCLK2	t3a	Low Time	9		8		7		2.0V	
NUMCLK2	t3b	Low Time	7		6		5		0.8V	
NUMCLK2	t4	Fall Time		8		8		7	3.7V to 0.8V	
NUMCLK2	t5	Rise Time		8		8		7	0.8V to 3.7V	

Table 5.2c. Timing Requirements of the Bus Interface Unit
T_C = 0°C to +80°C, V_{CC} = 5V ±5%

Pin	Symbol	Parameter	16 MHz 1.5V		20 MHz 1.5V		25 MHz 1.5V Preliminary		Test Conditions	Figure Reference
			Min (ns)	Max (ns)	Min (ns)	Max (ns)	Min (ns)	Max (ns)		
CPUCLK2	t1	Period	31.25	125	25	125	20	125	2.0V	5.1
CPUCLK2	t2a	High Time	9		8		7		2.0V	
CPUCLK2	t2b	High Time	5		5		4		3.7V	
CPUCLK2	t3a	Low Time	9		8		7		2.0V	
CPUCLK2	t3b	Low Time	7		6		5		0.8V	
CPUCLK2	t4	Fall Time		8		8		7	3.7V to 0.8V	
CPUCLK2	t5	Rise Time		8		8		7	0.8V to 3.7V	
CPUCLK2/ NUMCLK2		Ratio	10/16	14/10	10/16	14/10	10/16	14/10		
READYO#	t7	Out Delay	3	34	3	31	3	24	C_L = 75 pF†	5.2
READYO#	t7	Out Delay	4	31	3	27	3	21	C_L = 25 pF	
PEREQ (1)	t7	Out Delay	5	34	5	34	4	33	C_L = 75 pF†	
BUSY# (1)	t7	Out Delay	5	34	5	29	4	29	C_L = 75 pF†	
BUSY# (1, 2)	t7	Out Delay	N/A	N/A	N/A	N/A	4	27	C_L = 25 pF	
ERROR# (1)	t7	Out Delay	5	34	5	34	4	33	C_L = 75 pF†	
D31-D0	t8	Out Delay	1	54	1	54	0	50	C_L = 120 pF†	5.3
D31-D0	t10	Setup Time	11		11		11			
D31-D0	t11	Hold Time	11		11		11			
D31-D0 (3)	t12*	Float Time	6	33	6	27	5	24	C_L = 120 pF†	
PEREQ (3)	t13*	Float Time	1	60	1	50	1	40	C_L = 75 pF†	5.5
BUSY# (3)	t13*	Float Time	1	60	1	50	1	40	C_L = 75 pF†	
ERROR# (3)	t13*	Float Time	1	60	1	50	1	40	C_L = 75 pF†	
READYO# (3)	t13*	Float Time	1	60	1	50	1	40	C_L = 75 pF†	

*Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not tested.
†For 25 MHz, C_L = 50 pF.

Table 5.2c. Timing Requirements of the Bus Interface Unit (Continued)
$T_C = 0°C$ to $+80°C$, $V_{CC} = 5V \pm 5\%$

Pin	Symbol	Parameter	16 MHz 1.5V		20 MHz 1.5V		25 MHz 1.5V Preliminary		Test Conditions	Figure Reference
			Min (ns)	Max (ns)	Min (ns)	Max (ns)	Min (ns)	Max (ns)		
ADS#	t14	Setup Time	26		21		16			5.3
ADS#	t15	Hold Time	5		5		4			
W/R#	t14	Setup Time	26		21		16			
W/R#	t15	Hold Time	5		5		4			
READY# (4)	t16	Setup Time	21		12		9			
READY#	t17	Hold Time	4		4		4			
CMD0#	t16	Setup Time	21		19		16			
CMD0#	t17	Hold Time	2		2		4			
NPS1# NPS2	t16	Setup Time	21		19		16			
NPS1# NPS2	t17	Hold Time	2		2		4			
STEN	t16	Setup Time	21		21		15			
STEN	t17	Hold Time	2		2		2			
RESETIN	t18	Setup Time	13		12		10			5.4
RESETIN	t19	Hold Time	4		4		3			

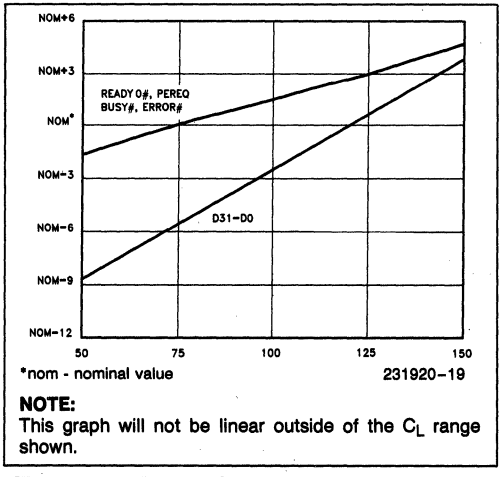
NOTES:
1.

	Min (ns)	
PEREQ, BUSY#, ERROR# Out Delay	4	@ $T_C = 0°C$
	4	@ $T_C = 85°C$

2. Not tested at 25 pF.
3. Float delay is not tested. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude.
4. Min (ns)

t_{16} Ready Setup $t_{16} = 2CLK2_{min} - t_{18}(82385)_{max} - LOGIC_{max}$

where:
$CLK2_{min}$ = 20 ns (period of 386 CLK2 input at 25 MHz)
$t_{18}(82385)$ = 82385 READYO# Valid Delay
$LOGIC_{max}$ = 10 ns (D-PAL)

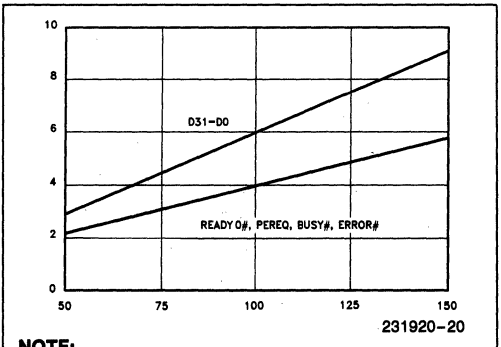


*nom - nominal value

231920-19

NOTE:
This graph will not be linear outside of the C_L range shown.

Figure 5.0a. Typical Output Valid Delay vs Load Capacitance at Max Operating Temperature



231920-20

NOTE:
This graph will not be linear outside of the C_L range shown.

Figure 5.0b. Typical Output Rise Time vs Load Capacitance at Max Operating Temperature

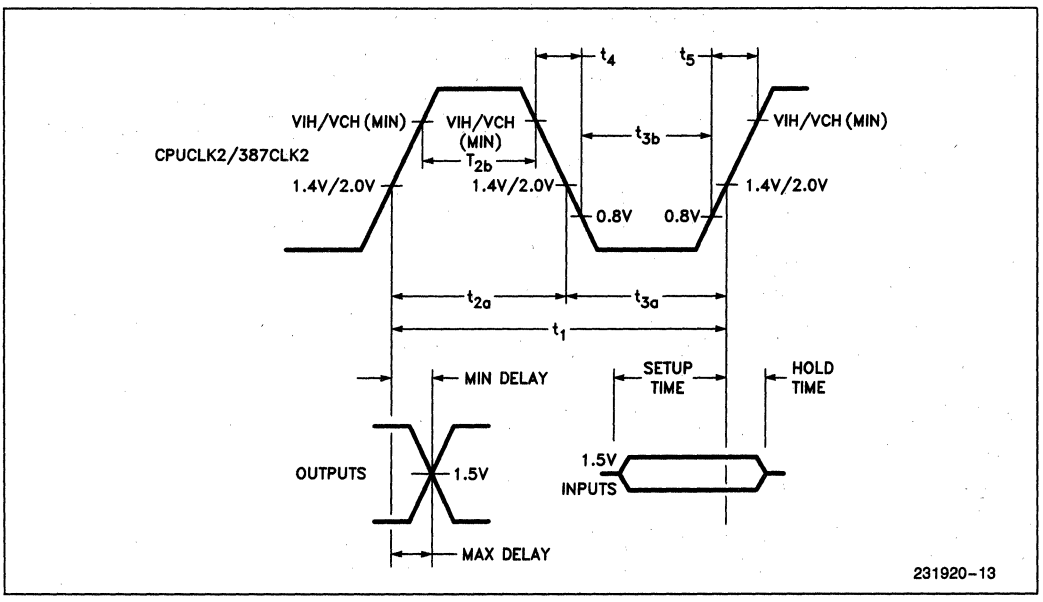


Figure 5.1. CPUCLK2/387CLK2 Waveform and Measurement Points for Input/Output A.C. Specifications

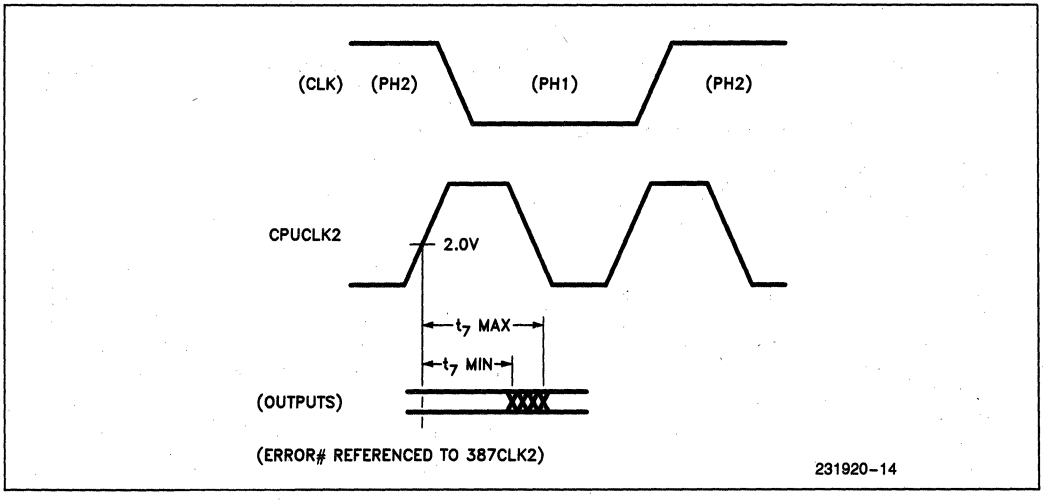


Figure 5.2. Output Signals

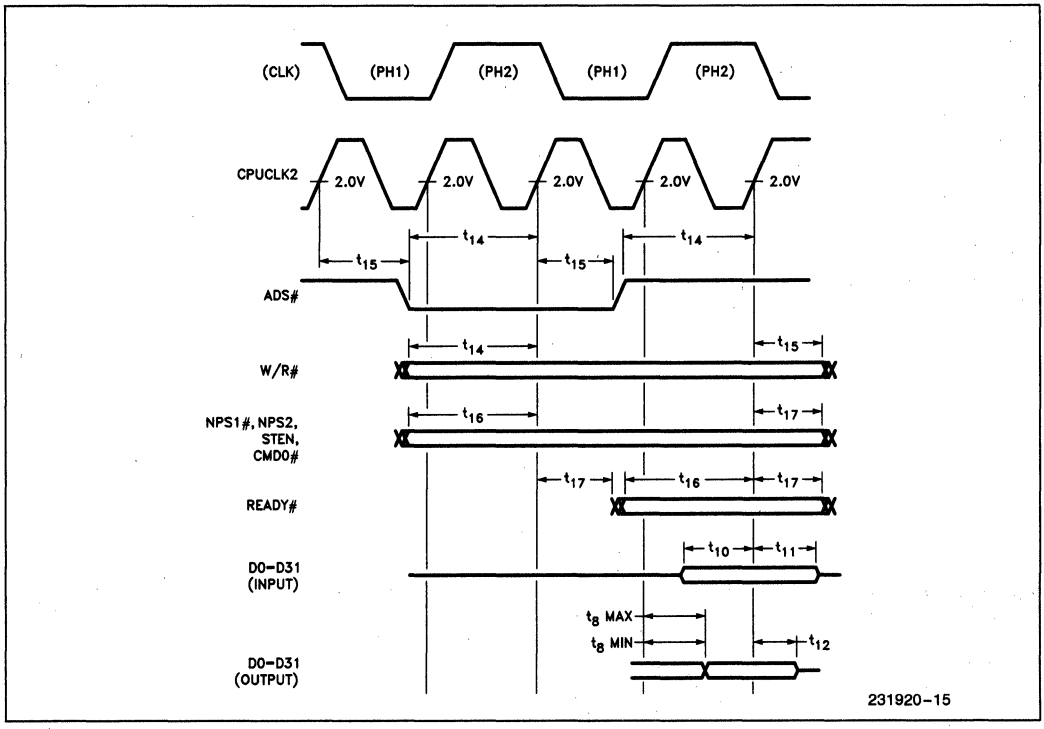
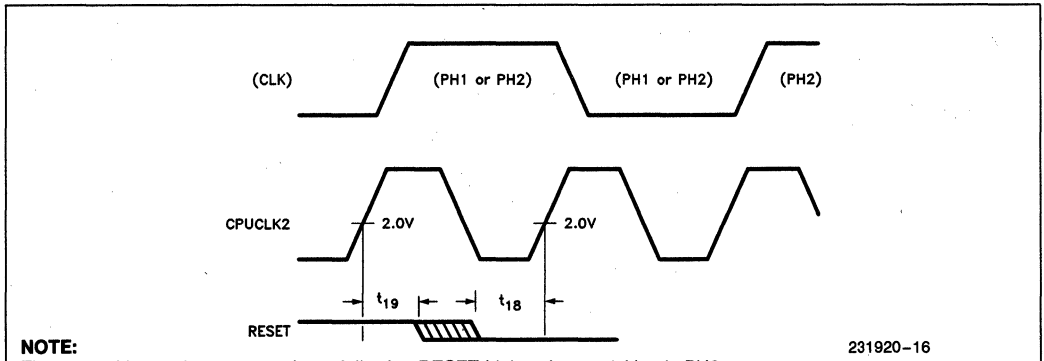


Figure 5.3. Input and I/O Signals



NOTE:
The second internal processor phase following RESET high to low transition is PH2.

Figure 5.4. RESET Signal

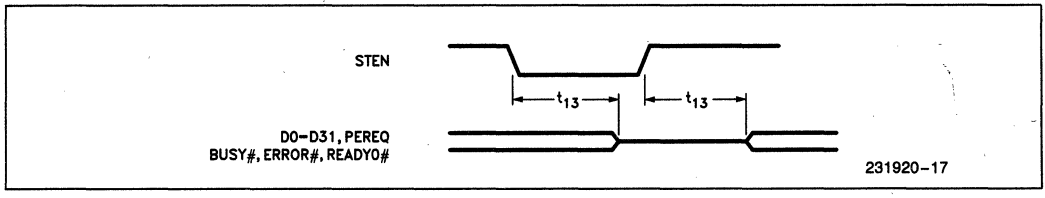
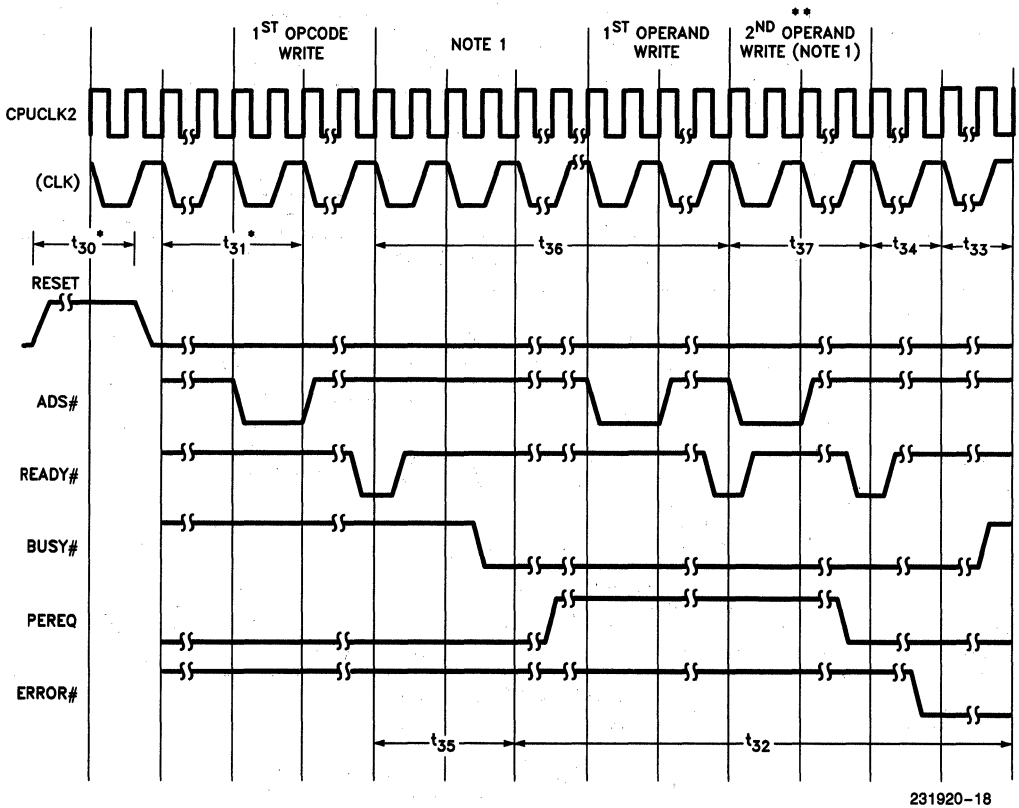


Figure 5.5. Float from STEN

Table 5.3. Other Parameters

Pin	Symbol	Parameter	Min	Max	Units
RESETIN	t30	Duration	40		NUMCLK2
RESETIN	t31	RESETIN Inactive to 1st Opcode Write	50		NUMCLK2
BUSY#	t32	Duration	6		CPUCLK2
BUSY#, ERROR#	t33	ERROR# (In) Active to BUSY# Inactive	6		CPUCLK2
PEREQ, ERROR#	t34	PEREQ Inactive to ERROR# Active	6		CPUCLK2
READY#, BUSY#	t35	READY# Active to BUSY# Active	4	4	CPUCLK2
READY#	t36	Minimum Time from Opcode Write to Opcode/Operand Write	6		CPUCLK2
READY#	t37	Minimum Time from Operand Write to Operand Write	8		CPUCLK2



* In NUMCLK2's
** or last operand

NOTE:
1. Memory read (operand) cycle is not shown.

Figure 5.6. Other Parameters

	Instruction								Optional Fields	
	First Byte				Second Byte					
1	11011	OPA		1	MOD		1	OPB R/M	SIB	DISP
2	11011	MF		OPA	MOD		OPB	R/M	SIB	DISP
3	11011	d	P	OPA	1	1	OPB	ST(i)		
4	11011	0	0	1	1	1	1	OP		
5	11011	0	1	1	1	1	1	OP		
	15–11	10	9	8	7	6	5	4 3 2 1 0		

6.0 80387 EXTENSIONS TO THE 386™ CPU INSTRUCTION SET

Instructions for the 80387 assume one of the five forms shown in the following table. In all cases, instructions are at least two bytes long and begin with the bit pattern 11011B, which identifies the ESCAPE class of instruction. Instructions that refer to memory operands specify addresses using the 386 CPU addressing modes.

OP = Instruction opcode, possible split into two fields OPA and OPB

MF = Memory Format
- 00—32-bit real
- 01—32-bit integer
- 10—64-bit real
- 11—16-bit integer

P = Pop
- 0—Do not pop stack
- 1—Pop stack after operation

ESC = 11011

d = Destination
- 0—Destination is ST(0)
- 1—Destination is ST(i)

R XOR d = 0—Destination (op) Source
R XOR d = 1—Source (op) Destination

ST(i) = Register stack element *i*
- 000 = Stack top
- 001 = Second stack element
- •
- •
- •
- 111 = Eighth stack element

MOD (Mode field) and R/M (Register/Memory specifier) have the same interpretation as the corresponding fields of the 386 Microprocessor instructions (refer to *386™ Microprocessor Programmer's Reference Manual*).

SIB (Scale Index Base) byte and DISP (displacement) are optionally present in instructions that have MOD and R/M fields. Their presence depends on the values of MOD and R/M, as for 386 Microprocessor instructions.

The instruction summaries that follow assume that the instruction has been prefetched, decoded, and is ready for execution; that bus cycles do not require wait states; that there are no local bus HOLD request delaying processor access to the bus; and that no exceptions are detected during instruction execution. If the instruction has MOD and R/M fields that call for both base and index registers, add one clock.

80387 Extensions to the 386™ CPU Instruction Set

Instruction	Encoding			Clock Count Range			
	Byte 0	Byte 1	Optional Bytes 2–6	32-Bit Real	32-Bit Integer	64-Bit Real	16-Bit Integer
DATA TRANSFER							
FLD = Load[a]							
Integer/real memory to ST(0)	ESC MF 1	MOD 000 R/M	SIB/DISP	20	45–52	25	61–65
Long integer memory to ST(0)	ESC 111	MOD 101 R/M	SIB/DISP		56–67		
Extended real memory to ST(0)	ESC 011	MOD 101 R/M	SIB/DISP		44		
BCD memory to ST(0)	ESC 111	MOD 100 R/M	SIB/DISP		266–275		
ST(i) to ST(0)	ESC 001	11000 ST(i)			14		
FST = Store							
ST(0) to integer/real memory	ESC MF 1	MOD 010 R/M	SIB/DISP	44	79–93	45	82–95
ST(0) to ST(i)	ESC 101	11010 ST(i)			11		
FSTP = Store and Pop							
ST(0) to integer/real memory	ESC MF 1	MOD 011 R/M	SIB/DISP	44	79–93	45	82–95
ST(0) to long integer memory	ESC 111	MOD 111 R/M	SIB/DISP		80–97		
ST(0) to extended real	ESC 011	MOD 111 R/M	SIB/DISP		53		
ST(0) to BCD memory	ESC 111	MOD 110 R/M	SIB/DISP		512–534		
ST(0) to ST(i)	ESC 101	11001 ST (i)			12		
FXCH = Exchange							
ST(i) and ST(0)	ESC 001	11001 ST(i)			18		
COMPARISON							
FCOM = Compare							
Integer/real memory to ST(0)	ESC MF 0	MOD 010 R/M	SIB/DISP	26	56–63	31	71–75
ST(i) to ST(0)	ESC 000	11010 ST(i)			24		
FCOMP = Compare and pop							
Integer/real memory to ST	ESC MF 0	MOD 011 R/M	SIB/DISP	26	56–63	31	71–75
ST(i) to ST(0)	ESC 000	11011 ST(i)			26		
FCOMPP = Compare and pop twice							
ST(1) to ST(0)	ESC 110	1101 1001			26		
FTST = Test ST(0)	ESC 001	1110 0100			28		
FUCOM = Unordered compare	ESC 101	11100 ST(i)			24		
FUCOMP = Unordered compare and pop	ESC 101	11101 ST(i)			26		
FUCOMPP = Unordered compare and pop twice	ESC 010	1110 1001			26		
FXAM = Examine ST(0)	ESC 001	11100101			30-38		
CONSTANTS							
FLDZ = Load +0.0 into ST(0)	ESC 001	1110 1110			20		
FLD1 = Load +1.0 into ST(0)	ESC 001	1110 1000			24		
FLDPI = Load pi into ST(0)	ESC 001	1110 1011			40		
FLDL2T = Load $\log_2(10)$ into ST(0)	ESC 001	1110 1001			40		

Shaded areas indicate instructions not available in 8087/80287.

NOTE:
a. When loading single- or double-precision zero from memory, add 5 clocks.

80387 Extensions to the 386™ CPU Instruction Set (Continued)

Instruction	Encoding			Clock Count Range			
	Byte 0	Byte 1	Optional Bytes 2–6	32-Bit Real	32-Bit Integer	64-Bit Real	16-Bit Integer
CONSTANTS (Continued)							
FLDL2E = Load $\log_2(e)$ into ST(0)	ESC 001	1110 1010			40		
FLDLG2 = Load $\log_{10}(2)$ into ST(0)	ESC 001	1110 1100			41		
FLDLN2 = Load $\log_e(2)$ into ST(0)	ESC 001	1110 1101			41		
ARITHMETIC							
FADD = Add							
Integer/real memory with ST(0)	ESC MF 0	MOD 000 R/M	SIB/DISP	24–32	57–72	29–37	71–85
ST(i) and ST(0)	ESC d P 0	11000 ST(i)			23–31[b]		
FSUB = Subtract							
Integer/real memory with ST(0)	ESC MF 0	MOD 10 R R/M	SIB/DISP	24–32	57–82	28–36	71–83[c]
ST(i) and ST(0)	ESC d P 0	1110 R R/M			26–34[d]		
FMUL = Multiply							
Integer/real memory with ST(0)	ESC MF 0	MOD 001 R/M	SIB/DISP	27–35	61–82	32–57	76–87
ST(i) and ST(0)	ESC d P 0	1100 1 R/M			29–57[e]		
FDIV = Divide							
Integer/real memory with ST(0)	ESC MF 0	MOD 11 R R/M	SIB/DISP	89	120–127[f]	94	136–140[g]
ST(i) and ST(0)	ESC d P 0	1111 R R/M			88[h]		
FSQRT[i] = Square root	ESC 001	1111 1010			122–129		
FSCALE = Scale ST(0) by ST(1)	ESC 001	1111 1101			67–86		
FPREM = Partial remainder	ESC 001	1111 1000			74–155		
FPREM1 = Partial remainder (IEEE)	ESC 001	1111 0101			95–185		
FRNDINT = Round ST(0) to integer	ESC 001	1111 1100			66–80		
FXTRACT = Extract components of ST(0)	ESC 001	1111 0100			70–76		
FABS = Absolute value of ST(0)	ESC 001	1110 0001			22		
FCHS = Change sign of ST(0)	ESC 001	1110 0000			24–25		

Shaded areas indicate instructions not available in 8087/80287.

NOTES:
b. Add 3 clocks to the range when d = 1.
c. Add 1 clock to **each** range when R = 1.
d. Add 3 clocks to the range when d = 0.
e. typical = 52 (When d = 0, 46–54, typical = 49).
f. Add 1 clock to the range when R = 1.
g. 135–141 when R = 1.
h. Add 3 clocks to the range when d = 1.
i. $-0 \leq ST(0) \leq +\infty$.

80387 Extensions to the 386™ CPU Instruction Set (Continued)

Instruction	Encoding			Clock Count Range
	Byte 0	Byte 1	Optional Bytes 2-6	
TRANSCENDENTAL				
FCOS[k] = Cosine of ST(0)	ESC 001	1111 1111		123-772[j]
FPTAN[k] = Partial tangent of ST(0)	ESC 001	1111 0010		191-497[j]
FPATAN = Partial arctangent	ESC 001	1111 0011		314-487
FSIN[k] = Sine of ST(0)	ESC 001	1111 1110		122-771[j]
FSINCOS[k] = Sine and cosine of ST(0)	ESC 001	1111 1011		194-809[j]
F2XM1[l] = $2^{ST(0)} - 1$	ESC 001	1111 0000		211-476
FYL2X[m] = ST(1) * $\log_2(ST(0))$	ESC 001	1111 0001		120-538
FYL2XP1[n] = ST(1) * $\log_2(ST(0) + 1.0)$	ESC 001	1111 1001		257-547
PROCESSOR CONTROL				
FINIT = Initialize NPX	ESC 011	1110 0011		33
FSTSW AX = Store status word	ESC 111	1110 0000		13
FLDCW = Load control word	ESC 001	MOD 101 R/M	SIB/DISP	19
FSTCW = Store control word	ESC 101	MOD 111 R/M	SIB/DISP	15
FSTSW = Store status word	ESC 101	MOD 111 R/M	SIB/DISP	15
FCLEX = Clear exceptions	ESC 011	1110 0010		11
FSTENV = Store environment	ESC 001	MOD 110 R/M	SIB/DISP	103-104
FLDENV = Load environment	ESC 001	MOD 100 R/M	SIB/DISP	71
FSAVE = Save state	ESC 101	MOD 110 R/M	SIB/DISP	375-376
FRSTOR = Restore state	ESC 101	MOD 100 R/M	SIB/DISP	308
FINCSTP = Increment stack pointer	ESC 001	1111 0111		21
FDECSTP = Decrement stack pointer	ESC 001	1111 0110		22
FFREE = Free ST(i)	ESC 101	1100 0 ST(i)		18
FNOP = No operations	ESC 001	1101 0000		12

Shaded areas indicate instructions not available in 8087/80287.

NOTES:
j. These timings hold for operands in the range $|x| < \pi/4$. For operands not in this range, up to 76 additional clocks may be needed to reduce the operand.
k. $0 \leq |ST(0)| < 2^{63}$.
l. $-1.0 \leq ST(0) \leq 1.0$.
m. $0 \leq ST(0) < \infty$, $-\infty < ST(1) < +\infty$.
n. $0 \leq |ST(0)| < (2 - SQRT(2))/2$, $-\infty < ST(1) < +\infty$.

APPENDIX A
COMPATIBILITY BETWEEN THE 80287 AND THE 8087

The 80286/80287 operating in Real-Address mode will execute 8086/8087 programs without major modification. However, because of differences in the handling of numeric exceptions by the 80287 NPX and the 8087 NPX, exception-handling routines *may* need to be changed.

This appendix summarizes the differences between the 80287 NPX and the 8087 NPX, and provides details showing how 8086/8087 programs can be ported to the 80286/80287.

1. The NPX signals exceptions through a dedicated $\overline{\text{ERROR}}$ line to the 80286. The NPX error signal does not pass through an interrupt controller (the 8087 INT signal does). Therefore, any interrupt-controller-oriented instructions in numeric exception handlers for the 8086/8087 should be deleted.
2. The 8087 instructions FENI/FNENI and FDISI/FNDISI perform no useful function in the 80287. If the 80287 encounters one of these opcodes in its instruction stream, the instruction will effectively be ignored—none of the 80287 internal states will be updated. While 8086/8087 containing these instructions may be executed on the 80286/80287, it is unlikely that the exception-handling routines containing these instructions will be completely portable to the 80287.
3. Interrupt vector 16 must point to the numeric exception handling routine.
4. The ESC instruction address saved in the 80287 includes any leading prefixes before the ESC opcode. The corresponding address saved in the 8087 does not include leading prefixes.
5. In Protected-Address mode, the format of the 80287's saved instruction and address pointers is different than for the 8087. The instruction opcode is not saved in Protected mode—exception handlers will have to retrieve the opcode from memory if needed.
6. Interrupt 7 will occur in the 80286 when executing ESC instructions with either TS (task switched) or EM (emulation) of the 80286 MSW set (TS = 1 or EM = 1). If TS is set, then a WAIT instruction will also cause interrupt 7. An exception handler should be included in 80286/80287 code to handle these situations.
7. Interrupt 9 will occur if the second or subsequent words of a floating-point operand fall outside a segment's size. Interrupt 13 will occur if the starting address of a numeric operand falls outside a segment's size. An exception handler should be included in 80286/80287 code to report these programming errors.
8. Except for the processor control instructions, all of the 80287 numeric instructions are automatically synchronized by the 80286 CPU—the 80286 automatically tests the $\overline{\text{BUSY}}$ line from the 80287 to ensure that the 80287 has completed its previous instruction before executing the next ESC instruction. No explicit WAIT instructions are required to assure this synchronization. For the 8087 used with 8086 and 8088 processors, explicit WAITs are required before each numeric instruction to ensure synchronization. Although 8086/8087 programs having explicit WAIT instructions will execute perfectly on the 80286/80287 without reassembly, these WAIT instructions are unnecessary.
9. Since the 80287 does not require WAIT instructions before each numeric instruction, the ASM286 assembler does not automatically generate these WAIT instructions. The ASM86 assembler, however, automatically precedes every ESC instruction with a WAIT instruction. Although numeric routines generated using the ASM86 assembler will generally execute correctly on the 80286/80287, reassembly using ASM286 may result in a more compact code image.

   The processor control instructions for the 80287 may be coded using either a WAIT or No-WAIT form of mnemonic. The WAIT forms of these instructions cause ASM286 to precede the ESC instruction with a CPU WAIT instruction, in the identical manner as does ASM86.

DATA SHEET REVISION REVIEW

The following list represents the key differences between this and the -003 versions of the 80387 Data Sheet. Please review this summary carefully.

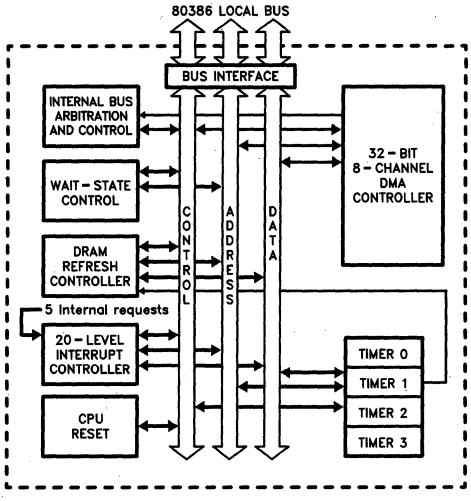
1. On the front page, the high side of the relative performance increase of the 80387 over the 8087/80287 was changed from seven to nine times to reflect the higher performance from a 25 MHz 80387.
2. Data type representation ranges were inaccurate, and are revised in Table 2.1.
3. The ratio of the 386 CLK2 frequency to the NUMCLK2 frequency must lie within the range 10:16 to 14:10 instead of the former 10:16 to 16:10 specification.
4. Figure 3.2 was added to illustrate the 80387 operating in an asynchronous mode.
5. The 80387 READYO# output must be included in the system READY# logic for a proper READY# signal to be generated.
6. Figure 3.5 contains a corrected W/R# signal.
7. The 80387 25 MHz A.C. and D.C. specifications are now available.

82380
HIGH PERFORMANCE 32-BIT DMA CONTROLLER WITH INTEGRATED SYSTEM SUPPORT PERIPHERALS

- **High Performance 32-Bit DMA Controller**
  - **— 50 MBytes/sec Maximum Data Transfer Rate at 25 MHz**
  - **— 8 Independently Programmable Channels**
- **20-Source Interrupt Controller**
  - **— Individually Programmable Interrupt Vectors**
  - **— 15 External, 5 Internal Interrupts**
  - **— 82C59A Superset**
- **Four 16-Bit Programmable Interval Timers**
  - **— 82C54 Compatible**
- **Programmable Wait State Generator**
  - **— 0 to 15 Wait States Pipelined**
  - **— 1 to 16 Wait States Non-Pipelined**
- **DRAM Refresh Controller**
- **80386 Shutdown Detect and Reset Control**
  - **— Software/Hardware Reset**
- **High Speed CHMOS III Technology**
- **132-Pin PGA Package**
- **Optimized for use with the 80386 Microprocessor**
  - **— Resides on Local Bus for Maximum Bus Bandwidth**

The 82380 is a multi-function support peripheral that integrates system functions necessary in an 80386 environment. It has eight channels of high performance 32-bit DMA with the most efficient transfer rates possible on the 80386 bus. System support peripherals integrated into the 82380 provide Interrupt Control, Timers, Wait State generation, DRAM Refresh Control, and System Reset logic.

The 82380's DMA Controller can transfer data between devices of different data path widths using a single channel. Each DMA channel operates independently in any of several modes. Each channel has a temporary data storage register for handling non-aligned data without the need for external alignment logic.



290128-1

82380 Internal Block Diagram

October 1988
Order Number: 290128-003

1.0 FUNCTIONAL OVERVIEW

The 82380 contains several independent functional modules. The following is a brief discussion of the components and features of the 82380. Each module has a corresponding detailed section later in this data sheet. Those sections should be referred to for design and programming information.

1.1 82380 Architecture

The 82380 is comprised of several computer system functions that are normally found in separate LSI and VLSI components. These include: a high-performance, eight-channel, 32-bit Direct Memory Access Controller; a 20-level Programmable Interrupt Controller which is a superset of the 82C59A; four 16-bit Programmable Interval Timers which are functionally equivalent to the 82C54 timers; a DRAM Refresh Controller; a Programmable Wait State Generator; and system reset logic. The interface to the 82380 is optimized for high-performance operation with the 80386 microprocessor.

The 82380 operates directly on the 80386 bus. In the Slave mode, it monitors the state of the processor at all times and acts or idles according to the commands of the host. It monitors the address pipeline status and generates the programmed number of wait states for the device being accessed. The 82380 also has logic to reset the 80386 via hardware or software reset requests and processor shutdown status.

After a system reset, the 82380 is in the Slave mode. It appears to the system as an I/O device. It becomes a bus master when it is performing DMA transfers.

To maintain compatibility with existing software, the registers within the 82380 are accessed as bytes. If the internal logic of the 82380 requires a delay before another access by the processor, wait states are automatically inserted into the access cycle. This allows the programmer to write initialization routines, etc. without regard to hardware recovery times.

Figure 1-1 shows the basic architectural components of the 82380. The following sections briefly discuss the architecture and function of each of the distinct sections of the 82380.

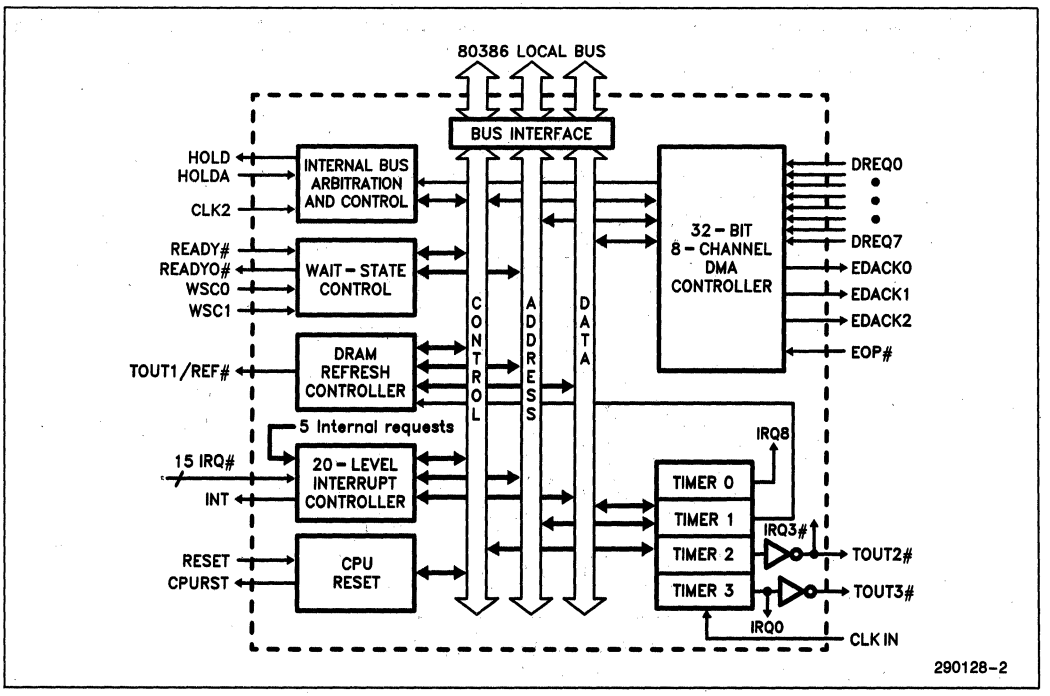


Figure 1-1. Architecture of the 82380

1.1.1 DMA CONTROLLER

The 82380 contains a high-performance, 8-channel, 32-bit DMA controller. It is capable of transferring any combination of bytes, words, and double words. The addresses of both source and distination can be independently incremented, decremented or held constant, and cover the entire 32-bit physical address space of the 80386. It can disassemble and assemble misaligned data via a 32-bit internal temporary data storage register. Data transferred between devices of different data path widths can also be assembled and disassembled using the internal temporary data storage register. The DMA Controller can also transfer aligned data between I/O and memory on the fly, allowing data transfer rates up to 32 megabytes per second for an 82380 operating at 16 MHz. Figure 1-2 illustrates the functional components of the DMA Controller.

There are twenty-four general status and command registers in the 82380 DMA Controller. Through these registers any of the channels may be programmed into any of the possible modes. The operating modes of any one channel are independent of the operation of the other channels.

Each channel has three programmable registers which determine the location and amount of data to be transferred:

Byte Count Register—Number of bytes to transfer. (24-bits)

Requester Register—Address of memory or peripheral which is requesting DMA service. (32-bits)

Target Register—Address of peripheral or memory which will be accessed. (32-bits)

There are also port addresses which, when accessed, cause the 82380 to perform specific functions. The actual data written does not matter, the act of writing to the specific address causes the command to be executed. The commands which operate in this mode are: Master Clear, Clear Terminal Count Interrupt Request, Clear Mask Register, and Clear Byte Pointer Flip-Flop.

DMA transfers can be done between all combinations of memory and I/O; memory-to-memory, memory-to-I/O, I/O-to-memory, and I/O-to-I/O. DMA service can be requested through software and/or hardware. Hardware DMA acknowledge signals are available for all channels (except channel 4) through an encoded 3-bit DMA acknowledge bus (EDACK0–2).

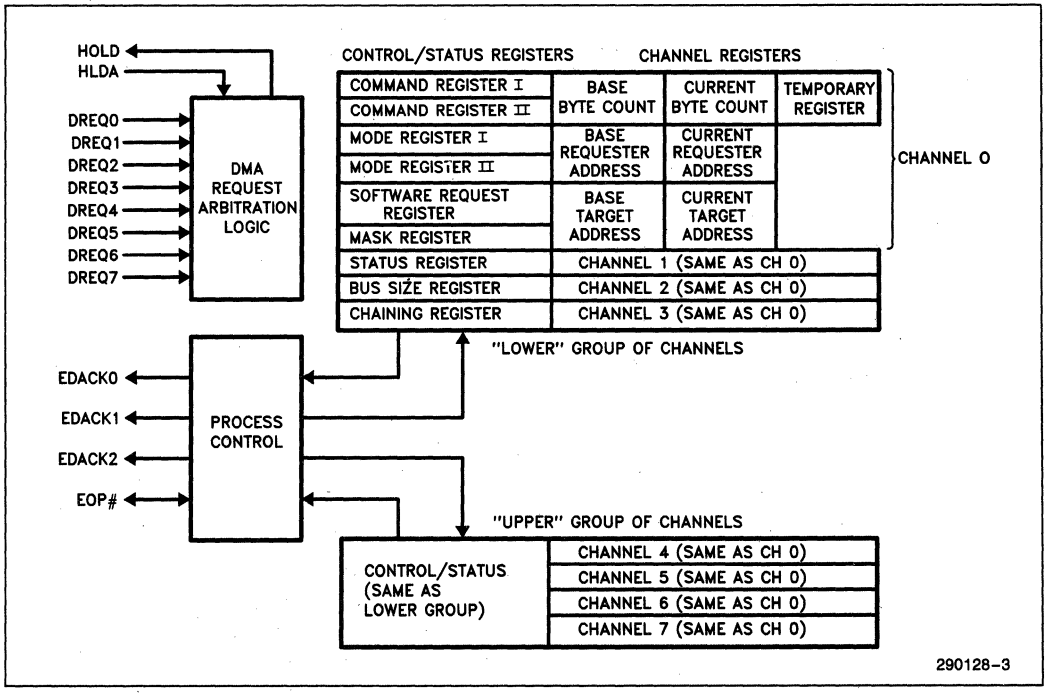


Figure 1-2. 82380 DMA Controller

The 82380 DMA controller transfers blocks of data (buffers) in three modes: Single Buffer, Buffer Auto-Initialize, and Buffer Chaining. In the Single Buffer Process, the 82380 DMA Controller is programmed to transfer one particular block of data. Successive transfers then require reprogramming of the DMA channel. Single Buffer transfers are useful in systems where it is known at the time the transfer begins what quantity of data is to be transferred, and there is a contiguous block of data area available.

The Buffer Auto-Initialize Process allows the same data area to be used for successive DMA transfers without having to reprogram the channel.

The Buffer Chaining Process allows a program to specify a list of buffer transfers to be executed. The 82380 DMA Controller, through interrupt routines, is reprogrammed from the list. The channel is reprogrammed for a new buffer before the current buffer transfer is complete. This pipelining of the channel programming process allows the system to allocate non-contiguous blocks of data storage space, and transfer all of the data with one DMA process. The buffers that make up the chain do not have to be in contiguous locations.

Channel priority can be fixed or rotating. Fixed priority allows the programmer to define the priority of DMA channels based on hardware or other fixed parameters. Rotating priority is used to provide peripherals access to the bus on a shared basis.

With fixed priority, the programmer can set any channel to have the current lowest priority. This allows the user to reset or manually rotate the priority schedule without reprogramming the command registers.

1.1.2 PROGRAMMABLE INTERVAL TIMERS

Four 16-bit programmable interval timers reside within the 82380. These timers are identical in function to the timers in the 82C54 Programmable Interval Timer. All four of the timers share a common clock input which can be independent of the system clock. The timers are capable of operating in six different modes. In all of the modes, the current count can be latched and read by the 80386 at any time, making these very versatile event timers. Figure 1-3 shows the functional components of the Programmable Interval Timers.

The outputs of the timers are directed to key system functions, making system design simpler. Timer 0 is routed directly to an interrupt input and is not available externally. This timer would typically be used to generate time-keeping interrupts.

Timers 1 and 2 have outputs which are available for general timer/counter purposes as well as special functions. Timer 1 is routed to the refresh control logic to provide refresh timing. Timer 2 is connected to an interrupt request input to provide other timer functions. Timer 3 is a general purpose timer/counter whose output is available to external hardware. It is also connected internally to the interrupt request which defaults to the highest priority (IRQ0).

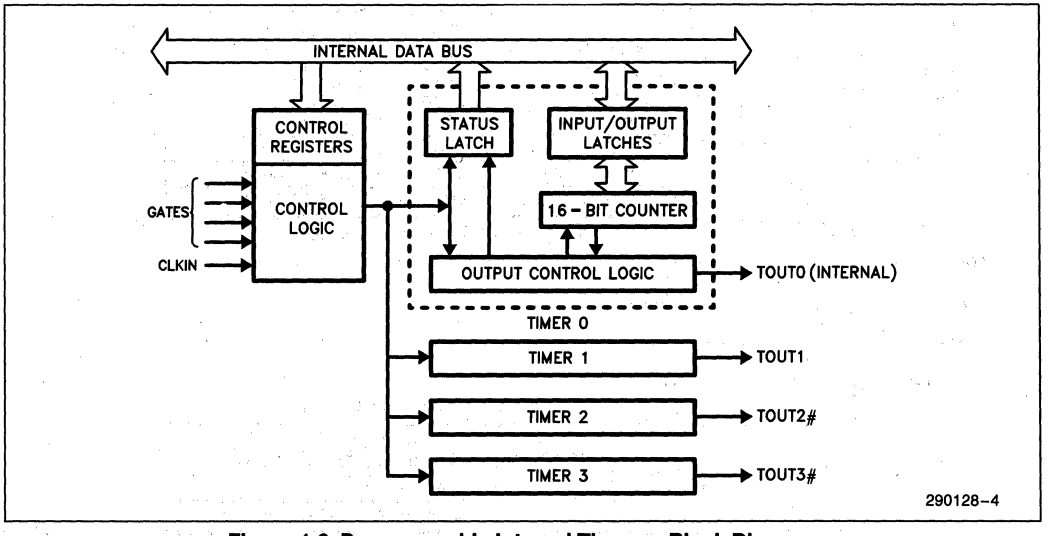


Figure 1-3. Programmable Interval Timers—Block Diagram

1.1.3 INTERRUPT CONTROLLER

The 82380 has the equivalent of three enhanced 82C59A Programmable Interrupt Controllers. These controllers can all be operated in the Master mode, but the priority is always as if they were cascaded. There are 15 interrupt request inputs provided for the user, all of which can be inputs from external slave interrupt controllers. Cascading 82C59As to these request inputs allows a possible total of 120 external interrupt requests. Figure 1-4 is a block diagram of the 82380 Interrupt Controller.

Each of the interrupt request inputs can be individually programmed with its own interrupt vector, allowing more flexibility in interrupt vector mapping than was available with the 82C59A. An interrupt is provided to alert the system that an attempt is being made to program the vectors in the method of the 82C59A. This provides compatibility of existing software that used the 82C59A or 8259A with new designs using the 82380.

In the event of an unrequested or otherwise erroneous interrupt acknowledge cycle, the 82380 Interrupt Controller issues a default vector. This vector, programmed by the system software, will alert the system of unsolicited interrupts of the 80386.

The functions of the 82380 Interrupt Controller are identical to the 82C59A, except in regards to programming the interrupt vectors as mentioned above. Interrupt request inputs are programmable as either edge or level triggered and are software maskable. Priority can be either fixed or rotating and interrupt requests can be nested.

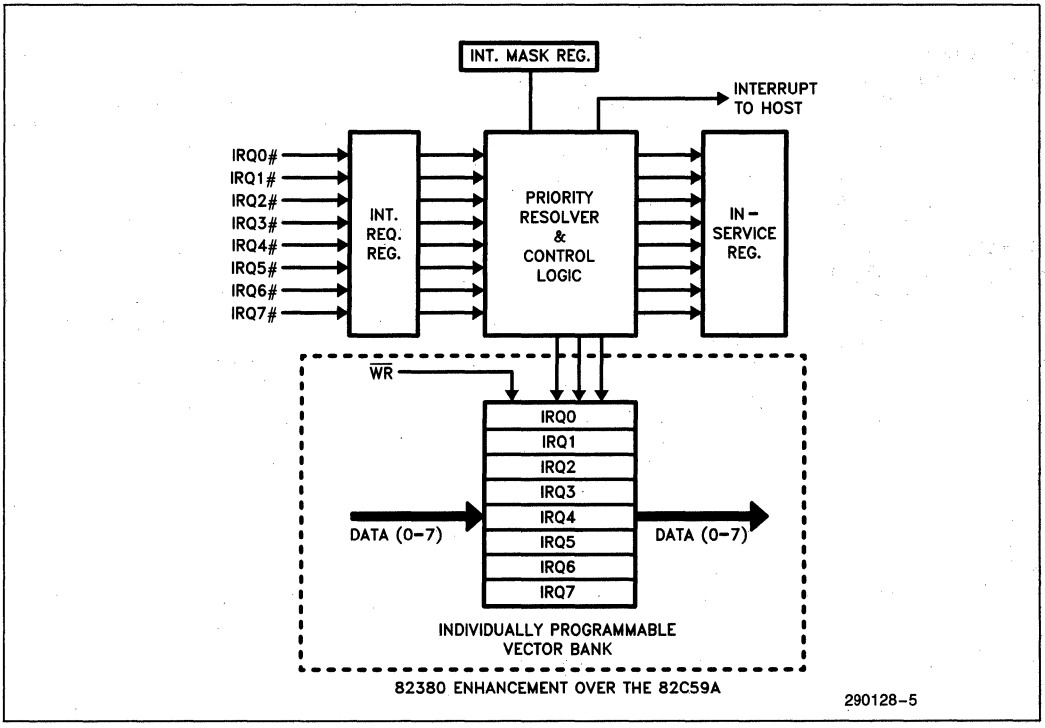


Figure 1-4. 82380 Interrupt Controller—Block Diagram

Enhancements are added to the 82380 for cascading external interrupt controllers. Master to Slave handshaking takes place on the data bus, instead of dedicated cascade lines.

1.1.4 WAIT STATE GENERATOR

The Wait State Generator is a programmable READY generation circuit for the 80386 bus. A peripheral requiring wait states can request the Wait State Generator to hold the processor's READY input inactive for a predetermined number of bus states. Six different wait state counts can be programmed into the Wait State Generator by software; three for memory accesses and three for I/O accesses. A block diagram of the 82380 Wait State Generator is shown in Figure 1-5.

The peripheral being accessed selects the required wait state count by placing a code on a 2-bit wait state select bus. This code along with the M/IO# signal from the bus master is used to select one of six internal 4-bit wait state registers which has been programmed with the desired number of wait states. From zero to fifteen wait states can be programmed into the wait state registers. The Wait State Generator tracks the state of the processor or current bus master at all times, regardless of which device is the current bus master and regardless of whether or not the Wait State Generator is currently active.

The 82380 Wait State Generator is disabled by making the select inputs both high. This allows hardware which is intelligent enough to generate its own ready signal to be accessed without penalty. As previously mentioned, deselecting the Wait State Generator does not disable its ability to determine the proper number of wait states due to pipeline status in subsequent bus cycles.

The number of wait states inserted into a pipelined bus cycle is the value in the selected wait state register. If the bus master is operating in the non-pipelined mode, the Wait State Generator will increase the number of wait states inserted into the bus cycle by one.

On reset, the Wait State Generator's registers are loaded with the value FFH, giving the maximum number of wait states for any access in which the wait state select inputs are active.

1.1.5 DRAM REFRESH CONTROLLER

The 82380 DRAM Refresh Controller consists of a 24-bit refresh address counter and bus arbitration logic. The output of Timer 1 is used to periodically request a refresh cycle. When the controller receives the request, it requests access to the system bus through the HOLD signal. When bus control is acknowledged by the processor or current bus master, the refresh controller executes a memory read operation at the address currently in the Refresh Address Register. At the same time, it activates a refresh signal (REF#) that the memory uses to force a refresh instead of a normal read. Control of the bus is transferred to the processor at the completion of this cycle. Typically a refresh cycle will take six clock cycles to execute on an 80386 bus.

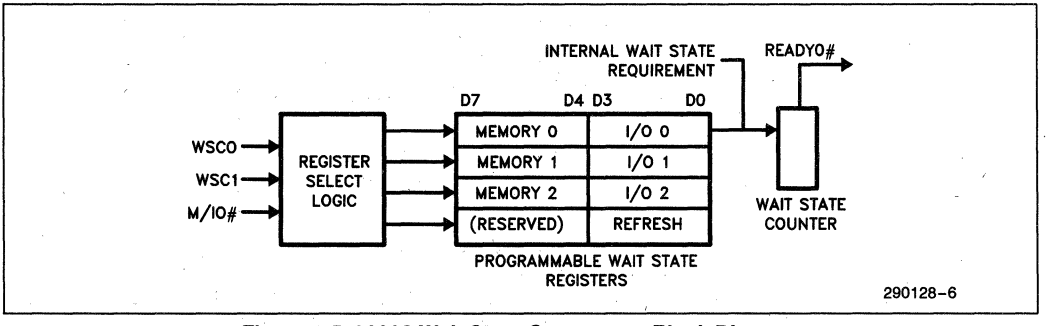


Figure 1-5. 82380 Wait State Generator—Block Diagram

The 82380 DRAM Refresh Controller has the highest priority when requesting bus access and will interrupt any active DMA process. This allows large blocks of data to be moved by the DMA controller without affecting the refresh function. Also the DMA controller is not required to completely relinquish the bus, the refresh controller simply steals a bus cycle between DMA accesses.

The amount by which the refresh address is incremented is programmable to allow for different bus widths and memory bank arrangements.

1.1.6 CPU RESET FUNCTION

The 82380 contains a special reset function which can respond to hardware reset signals from the 82384, as well as a software reset command. The circuit will hold the 80386's RESET line active while an external hardware reset signal is present at its RESET input. It can also reset the 80386 processor as the result of a software command. The software reset command causes the 82380 to hold the processor's RESET line active for a minimum of 62 CLK2 cycles; enough time to allow an 80386 to re-initialize.

The 82380 can be programmed to sense the shutdown detect code on the status lines from the 80386. If the Shutdown Detect function is enabled, the 82380 will automatically reset the processor. A diagnostic register is available which can be used to determine the cause of reset.

1.1.7 REGISTER MAP RELOCATION

After a hardware reset, the internal registers of the 82380 are located in I/O space beginning at port address 0000H. The map of the 82380's registers is relocatable via a software command. The default mapping places the 82380 between I/O addresses 0000H and 00DBH. The relocation register allows this map to be moved to any even 256-byte boundary in the processor's 16-bit I/O address space or any even 16-Mbyte boundary in the 32-bit memory address space.

1.2 Host Interface

The 82380 is designed to operate efficiently on the local bus of an 80386 microprocessor. The control signals of the 82380 are identical in function to those of the 80386. As a slave, the 82380 operates with all of the features available on the 80386 bus. When the 82380 is in the Master mode, it looks identical to the 80386 to the connected devices.

The 82380 monitors the bus at all times, and determines whether the current bus cycle is a pipelined or non-pipelined access. All of the status signals of the processor are monitored.

The control, status, and data registers within the 82380 are located at fixed addresses relative to each other, but the group can be relocated to either memory or I/O space and to different locations within those spaces.

As a Slave device, the 82380 monitors the control/status lines of the CPU. The 82380 will generate all of the wait states it needs whenever it is accessed. This allows the programmer the freedom of accessing 82380 registers without having to insert NOPs in the program to wait for slower 82380 internal registers.

The 82380 can determine if a current bus cycle is a pipelined or a non-pipelined cycle. It does this by monitoring the ADS# and READY# signals and thereby keeping track of the current state of the 80386.

As a bus master, the 82380 looks like an 80386 to the rest of the system. This enables the designer greater flexibility in systems which include the 82380. The designer does not have to alter the interfaces of any peripherals designed to operate with the 80386 to accommodate the 82380. The 82380 will access any peripherals on the bus in the same manner as the 80386, including recognizing pipelined bus cycles.

The 82380 is accessed as an 8-bit peripheral. This is done to maintain compatibility with existing system architectures and software. The 80386 places the data of all 8-bit accesses either on D (0–7) or D (8–15). The 82380 will only accept data on these lines when in the Slave mode. When in the Master mode, the 82380 is a full 32-bit machine, sending and receiving data in the same manner as the 80386.

1.3 IBM PC* System Compatibility

The 82380 is an 80386 companion device designed to provide an enhancement of the system functions common to most small computer systems. It is modeled after and is a superset of the Intel peripheral products found in the IBM PC, PC-AT, and other popular small computers.

2.0 80386 HOST INTERFACE

The 82380 contains a set of interface signals to operate efficiently with the 80386 host processor. These signals were designed so that minimal hardware is needed to connect the 82380 to the 80386.

Figure 2-1 depicts a typical system configuration with the 80386 processor. As shown in the diagram, the 82380 is designed to interface directly with the 80386 bus.

*IBM PC and IBM PC-AT are registered trademarks of International Business Machines Inc.

Since the 82380 is residing on the opposite side of the data bus transceiver (with respect to the rest of the peripherals in the system), it is important to note that the transceiver should be controlled so that contention between the data bus transceiver and the 82380 will not occur. In order to do this, port address decoding logic should be included in the direction and enable control logic of the transceiver. When any of the 82380 internal registers is read, the data bus transceiver should be disabled so that only the 82380 will drive the local bus.

This section describes the basic bus functions of the 82380 to show how this device interacts with the 80386 processor. Other signals which are not directly related to the host interface will be discussed in their associated functional block description.

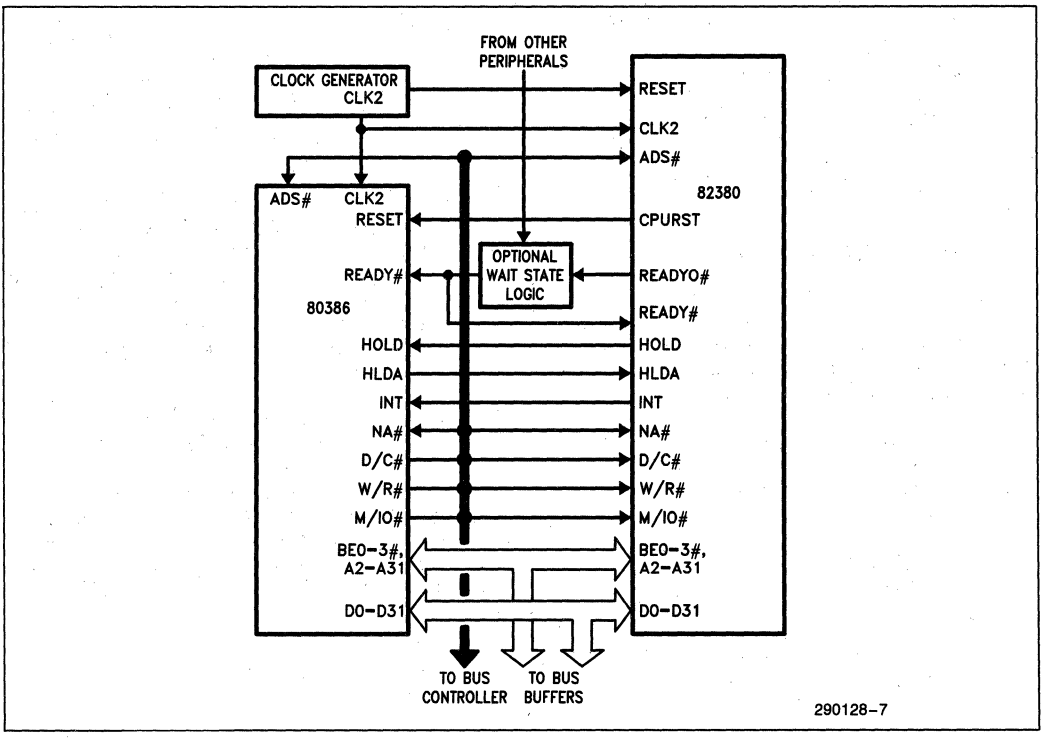


Figure 2-1. 80386/82380 System Configuration

2.1 Master and Slave Modes

At any time, the 82380 acts as either a Slave device or a Master device in the system. Upon reset, the 82380 will be in the Slave Mode. In this mode, the 80386 processor can read/write into the 82380 internal registers. Initialization information may be programmed into the 82380 during Slave Mode.

When DMA service (including DRAM Refresh Cycles generated by the 82380) is requested, the 82380 will request and subsequently get control of the 80386 local bus. This is done through the HOLD and HLDA (Hold Acknowledge) signals. When the 80386 processor responds by asserting the HLDA signal, the 82380 will switch into Master Mode and perform DMA transfers. In this mode, the 82380 is the bus master of the system. It can read/write data from/to memory and peripheral devices. The 82380 will return to the Slave Mode upon completion of DMA transfers, or when HLDA is negated.

2.2 80386 Interface Signals

As mentioned in the Architecture section, the Bus Interface module of the 82380 (see Figure 1-1) contains signals that are directly connected to the 80386 host processor. This module has separate 32-bit Data and Address busses. Also, it has additional control signals to support different bus operations on the system. By residing on the 80386 local bus, the 82380 shares the same address, data and control lines with the processor. The following subsections discuss the signals which interface to the 80386 host processor.

2.2.1 CLOCK (CLK2)

The CLK2 input provides fundamental timing for the 82380. It is divided by two internally to generate the 82380 internal clock. Therefore, CLK2 should be driven with twice the 80386's frequency. In order to maintain synchronization with the 80386 host processor, the 82380 and the 80386 should share a common clock source.

The internal clock consists of two phases: PHI1 and PHI2. Each CLK2 period is a phase of the internal clock. PHI2 is usually used to sample input and set up internal signals and PHI1 is for latching internal data. Figure 2-2 illustrates the relationship of CLK2 and the 82380 internal clock signals. The CPURST signal generated by the 82380 guarantees that the 80386 will wake up in phase with PHI1.

2.2.2 DATA BUS (D0–D31)

This 32-bit three-state bidirectional bus provides a general purpose data path between the 82380 and the system. These pins are tied directly to the corresponding Data Bus pins of the 80386 local bus. The Data Bus is also used for interrupt vectors generated by the 82380 in the Interrupt Acknowledge cycle.

During Slave I/O operations, the 82380 expects a single byte to be written or read. When the 80386 host processor writes into the 82380, either D0–D7 or D8–D15 will be latched into the 82380, depending upon how the Byte Enable (BE0#–BE#3) signals are driven. The 82380 does not need to look at D16–D31 since the 80386 duplicates the single byte

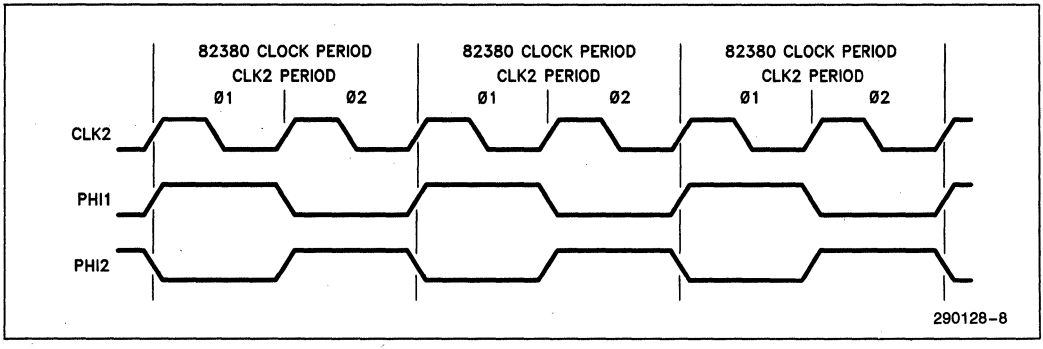


Figure 2-2. CLK2 and 82380 Internal Clock

data on both halves of the bus. When the 80386 host processor reads from the 82380, the single byte data will be duplicated four times on the Data Bus; i.e., on D0–D7, D8–D15, D16–D23 and D24–D31.

During Master Mode, the 82380 can transfer 32-, 16-, and 8-bit data between memory (or I/O devices) and I/O devices (or memory) via the Data Bus.

2.2.3 ADDRESS BUS (A31–A2)

These three-state bidirectional signals are connected directly to the 80386 Address Bus. In the Slave Mode, they are used as input signals so that the processor can address the 82380 internal ports/registers. In the Master Mode, they are used as output signals by the 82380 to address memory and peripheral devices. The Address Bus is capable of addressing 4 G-bytes of physical memory space (00000000H to FFFFFFFFH), and 64 K-bytes of I/O addresses (00000000H to 0000FFFFH).

2.2.4 BYTE ENABLE (BE3#–BE0#)

These bidirectional pins select specific byte(s) in the double word addressed by A31–A2. Similar to the Address Bus function, these signals are used as inputs to address internal 82380 registers during Slave Mode operation. During Master Mode operation, they are used as outputs by the 82380 to address memory and I/O locations.

In addition to the above function, BE3# is used to enable a production test mode and must be LOW during reset. The 80386 processor will automatically hold BE3# LOW during RESET.

The definitions of the Byte Enable signals depend upon whether the 82380 is in the Master or Slave Mode. These definitions are depicted in Table 2-1.

Table 2-1. Byte Enable Signals

As INPUTS (Slave Mode):

BE3#–BE0#	Implied A1, A0	Data Bits Written to 82380*
XXX0	00	D0–D7
XX01	01	D8–D15
X011	10	D0–D7
X111	11	D8–D15

X–DON'T CARE
*During READ, data will be duplicated on D0–D7, D8–D15, D16–D23, and D24–D31.
During WRITE, the 80386 host processor duplicates data on D0–D15, and D16–D31, so that the 82380 is concerned only with the lower half of the Data Bus.

As OUTPUTS (Master Mode):

BE3#–BE0#	Byte to be Accessed Relative to A31–A2	Logical Byte Presented On Data Bus During WRITE Only*			
		D24–31	D16–23	D8–15	D0–7
1110	0	U	U	U	A
1101	1	U	U	A	A
1011	2	U	A	U	A
0111	3	A	U	A	A
1001	1, 2	U	B	A	A
1100	0, 1	U	U	B	A
0011	2, 3	B	A	B	A
1000	0, 1, 2	U	C	B	A
0001	1, 2, 3	C	B	A	A
0000	0, 1, 2, 3	D	C	B	A

U = Undefined
A = Logical D0–D7
B = Logical D8–D15
C = Logical D16–D23
D = Logical D24–D31
*Actual number of bytes accessed depends upon the programmed data path width.

2.2.5 BUS CYCLE DEFINITION SIGNALS (D/C#, W/R#, M/IO#)

These three-state bidirectional signals define the type of bus cycle being performed. W/R# distinguishes between write and read cycles. D/C# distinguishes between processor data and control cycles. M/IO# distinguishes between memory and I/O cycles.

During Slave Mode, these signals are driven by the 80386 host processor; during Master Mode, they are driven by the 82380. In either mode, these signals will be valid when the Address Status (ADS#) is driven LOW. Exact bus cycle definitions are given in Table 2-2. Note that some combinations are recognized as inputs, but not generated as outputs. In the Master Mode, D/C# is always HIGH.

2.2.6 ADDRESS STATUS (ADS#)

This bidirectional signal indicates that a valid address (A2–A31, BE0#–BE3#) and bus cycle definition (W/R#, D/C#, M/IO#) is being driven on the bus. In the Master Mode, it is driven by the 82380 as an output. In the Slave Mode, this signal is monitored as an input by the 82380. By the current and past status of ADS# and the READY# input, the 82380 is able to determine, during Slave Mode, if the next bus cycle is a pipelined address cycle. ADS# is asserted during T1 and T2P bus states (see Bus State Definition).

Note that during the idle states at the beginning and the end of a DMA process, neither the 80386 nor the 82380 is driving the ADS# signal; i.e., the signal is left floated. Therefore, it is important to use a pull-up resistor (approximately 10 KΩ) on the ADS# signal.

2.2.7 TRANSFER ACKNOWLEDGE (READY#)

This input indicates that the current bus cycle is complete. In the Master Mode, assertion of this signal indicates the end of a DMA bus cycle. In the Slave Mode, the 82380 monitors this input and ADS# to detect a pipelined address cycles. This signal should be tied directly to the READY# input of the 80386 host processor.

2.2.8 NEXT ADDRESS REQUEST (NA#)

This input is used to indicate to the 82380 in the Master Mode that the system is requesting address pipelining. When driven LOW by either memory or peripheral devices during Master Mode, it indicates that the system is prepared to accept a new address and bus cycle definition signals from the 82380 before the end of the current bus cycle. If this input is active when sampled by the 82380, the next address is driven onto the bus, provided a bus request is already pending internally.

This input pin is monitored only in the Master Mode. In the Slave Mode, the 82380 uses the ADS# and READY# signals to determine address pipelining cycles, and NA# will be ignored.

2.2.9 RESET (RESET, CPURST)

RESET

This synchronous input suspends any operation in progress and places the 82380 in a known initial state. Upon reset, the 82380 will be in the Slave Mode waiting to be initialized by the 80386 host processor. The 82380 is reset by asserting RESET for 15 or more CLK2 periods. When RESET is asserted, all other input pins are ignored, and all other bus pins are driven to an idle bus state as shown in Table 2-3. The 82380 will determine the phase of its internal clock following RESET going inactive.

Table 2-2. Bus Cycle Definition

M/IO#	D/C#	W/R#	As INPUTS	As OUTPUTS
0	0	0	Interrupt Acknowledge	NOT GENERATED
0	0	1	UNDEFINED	NOT GENERATED
0	1	0	I/O Read	I/O Read
0	1	1	I/O Write	I/O Write
1	0	0	UNDEFINED	NOT GENERATED
1	0	1	HALT if BE(3–0) # = X011 SHUTDOWN if BE (3–0)# = XXX0	NOT GENERATED
1	1	0	Memory Read	Memory Read
1	1	1	Memory Write	Memory Write

Table 2-3. Output Signals Following RESET

Signal	Level
A2–A31, D0–D31, BE0#–BE3#	Float
D/C#, W/R#, M/IO#, ADS#	Float
READYO#	'1'
EOP#	'1' (Weak Pull-UP)
EDACK2–EDACK0	'100'
HOLD	'0'
INT	UNDEFINED*
TOUT1/REF#, TOUT2#/IRQ3#, TOUT3#	UNDEFINED*
CPURST	'0'

*The Interrupt Controller and Programmable Interval Timer are initialized by software commands.

RESET is level-sensitive and must be synchronous to the CLK2 signal. Therefore, this RESET input should be tied to the RESET output of the Clock Generator. The RESET setup and hold time requirements are shown in Figure 2.3.

CPURST

This output signal is used to reset the 80386 host processor. It will go active (HIGH) whenever one of the following events occurs: a) 82380's RESET input is active; b) a software RESET command is issued to the 82380; or c) when the 82380 detects a processor Shutdown cycle and when this detection feature is enabled (see CPU Reset and Shutdown Detect). When activated, CPURST will be held active for 62 CLK2 periods. The timing of CPURST is such that the 80386 processor will be in synchronization with the 82380. This timing is shown in Figure 2-4.

2.2.10 INTERRUPT OUT (INT)

This output pin is used to signal the 80386 host processor that one or more interrupt requests (either internal or external) are pending. The processor is expected to respond with an Interrupt Acknowledge cycle. This signal should be connected directly to the Maskable Interrupt Request (INTR) input of the 80386 host processor.

2.3 82380 Bus Timing

The 82380 internally divides the CLK2 signal by two to generate its internal clock. Figure 2-2 shows the relationship of CLK2 and the internal clock. The internal clock consists of two phases: PHI1 and PHI2. Each CLK2 period is a phase of the internal clock. In Figure 2-2, both PHI1 and PHI2 of the 82380 internal clock are shown.

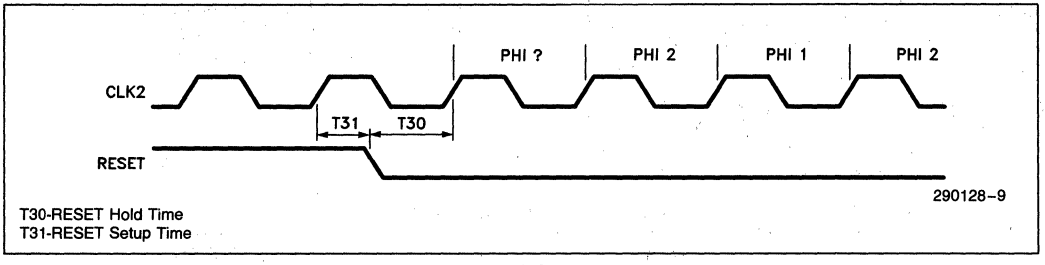


Figure 2-3. RESET Timing

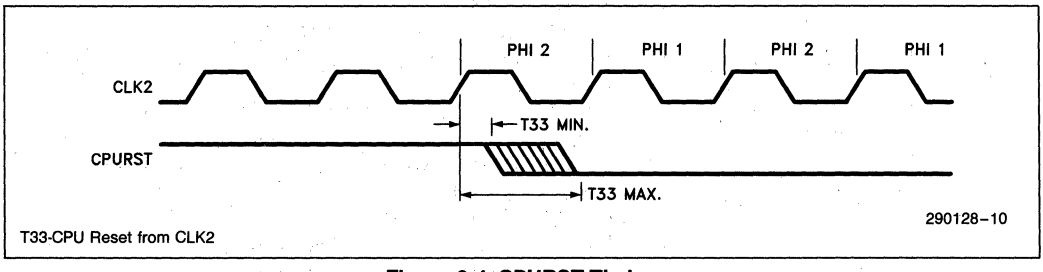


Figure 2-4. CPURST Timing

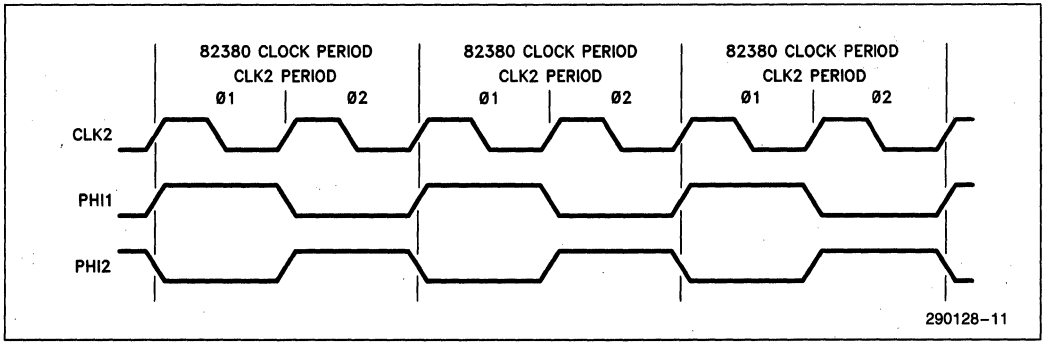


Figure 2-2. CLK2 and 82380 Internal Clock

In the 82380, whether it is in the Master or Slave Mode, the shortest time unit of bus activity is a bus state. A bus state, which is also referred as a 'T-state', is defined as one 82380 PHI2 clock period (i.e., two CLK2 periods). Recall in Table 2-2, there are six different types of bus cycles in the 82380 as defined by the M/IO#, D/C# and W/R# signals. Each of these bus cycles is composed of two or more bus states. The length of a bus cycle depends on when the READY# input is asserted (i.e., driven LOW).

2.3.1 ADDRESS PIPELINING

The 82380 supports Address Pipelining as an option in both the Master and Slave Mode. This feature typically allows a memory or peripheral device to operate with one less wait state than would otherwise be required. This is possible because during a pipelined cycle, the address and bus cycle definition of the next cycle will be generated by the bus master while waiting for the end of the current cycle to be acknowledged. The pipelined bus is especially well suited for interleaved memory environment. For 16 MHz interleaved memory designs with 100 ns access time DRAMs, zero wait state memory accesses can be achieved when pipelined addressing is selected.

In the Master Mode, the 82380 is capable of initiating, on a cycle-by-cycle basis, either a pipelined or non-pipelined access depending upon the state of the NA# input. If a pipelined cycle is requested (indicated by NA# being driven LOW), the 82380 will drive the address and bus cycle definition of the next cycle as soon as there is an internal bus request pending.

In the Slave Mode, the 82380 is constantly monitoring the ADS# and READY# signals on the processor local bus to determine if the current bus cycle is a pipelined cycle. If a pipelined cycle is detected, the 82380 will request one less wait state from the processor if the Wait State Generator feature is selected. On the other hand, during an 82380 internal register access in a pipelined cycle, it will make use of the advance address and bus cycle information. In all cases, Address Pipelining will result in a savings of one wait state.

2.3.2 MASTER MODE BUS TIMING

When the 82380 is in the Master Mode, it will be in one of six bus states. Figure 2-5 shows the complete bus state diagram of the Master Mode, including pipelined address states. As seen in the figure, the 82380 state diagram is very similar to that of the 80386. The major difference is that in the 82380, there is no Hold state. Also, in the 82380, the conditions for some state transitions depend upon whether it is the end of a DMA process*.

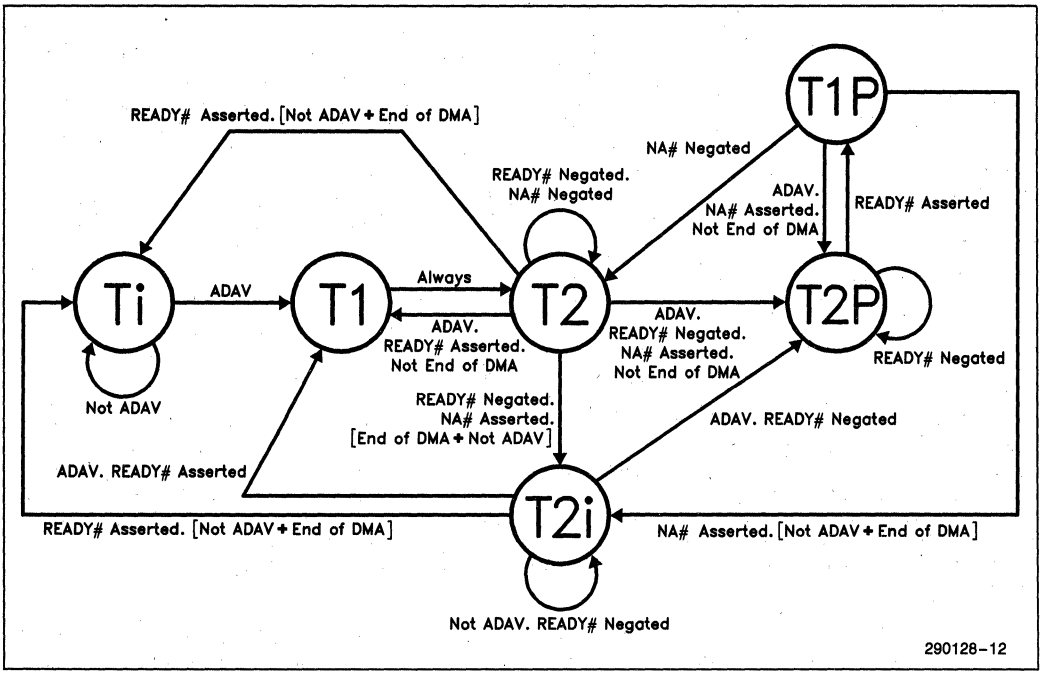
NOTE:

*The term 'end of a DMA process' is loosely defined here. It depends on the DMA modes of operation as well as the state of the EOP# and DREQ inputs. This is explained in detail in section 3—DMA Controller.

The 82380 will enter the idle state, Ti, upon RESET and whenever the internal address is not available at the end of a DMA cycle or at the end of a DMA process. When address pipelining is not used (NA# is not asserted), a new bus cycle always begins with state T1. During T1, address and bus cycle definition signals will be driven on the bus. T1 is always followed by T2.

If a bus cycle is not acknowledged (with READY#) during T2 and NA# is negated, T2 will be repeated. When the end of the bus cycle is acknowledged during T2, the following state will be T1 of the next bus cycle (if the internal address latch is loaded and if this is not the end of the DMA process). Otherwise, the Ti state will be entered. Therefore, if the memory or peripheral accessed is fast enough to respond within the first T2, the fastest non-pipelined cycle will take one T1 and one T2 state.

Use of the address pipelining feature allows the 82380 to enter three additional bus states: T1P, T2P, and T2i. T1P is the first bus state of a pipelined bus cycle. T2P follows T1P (or T2) if NA# is asserted when sampled. The 82380 will drive the bus with the address and bus cycle definition signals of the next cycle during T2P. From the state diagram, it can be seen that after an idle state Ti, the first bus cycle must begin with T1, and is therefore a non-pipelined bus cycle. The next bus cycle can be pipelined if NA# is asserted and the previous bus cycle ended in a T2P state. Once the 82380 is in a pipelined cycle and provided that NA# is asserted in subsequent cycles, the 82380 will be switching between T1P and T2P states. If the end of the current bus cycle is not acknowledged by the READY# input, the 82380 will extend the cycle by adding T2P states. The fastest pipelined cycle will consist of one T1P and one T2P state.



NOTE:
ADAV—Internal Address Available

Figure 2-5. Master Mode State Diagram

The 82380 will enter state T2i when NA# is asserted and when one of the following two conditions occurs. The first condition is when the 82380 is in state T2. T2i will be entered if READY# is not asserted and there is no next address available. This situation is similar to a wait state. The 82380 will stay in T2i for as long as this condition exists. The second condition which will cause the 82380 enter T2i is when the 82380 is in state T1P. Before going to state T2P, the 82380 needs to wait in state T2i until the next address is available. Also, in both cases, if the DMA process is complete, the 82380 will enter the T2i state in order to finish the current DMA cycle.

Figure 2-6 is a timing diagram showing non-pipelined bus accesses in the Master Mode. Figure 2-7 shows the timing of pipelined accesses in the Master Mode.

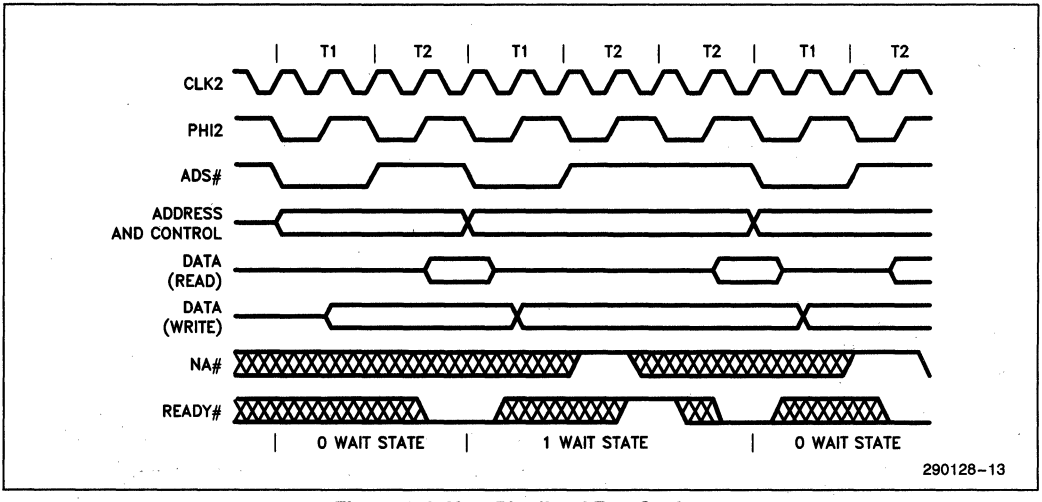


Figure 2-6. Non-Pipelined Bus Cycles

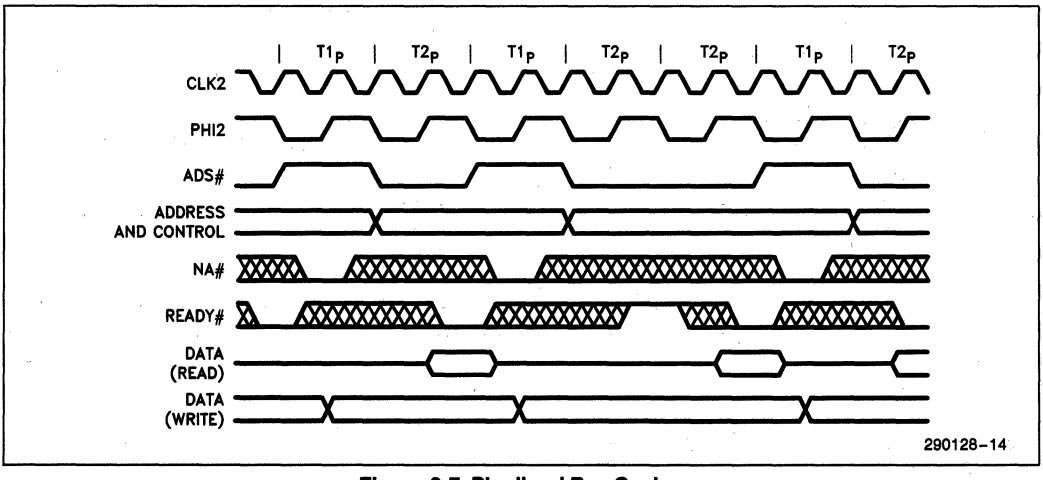
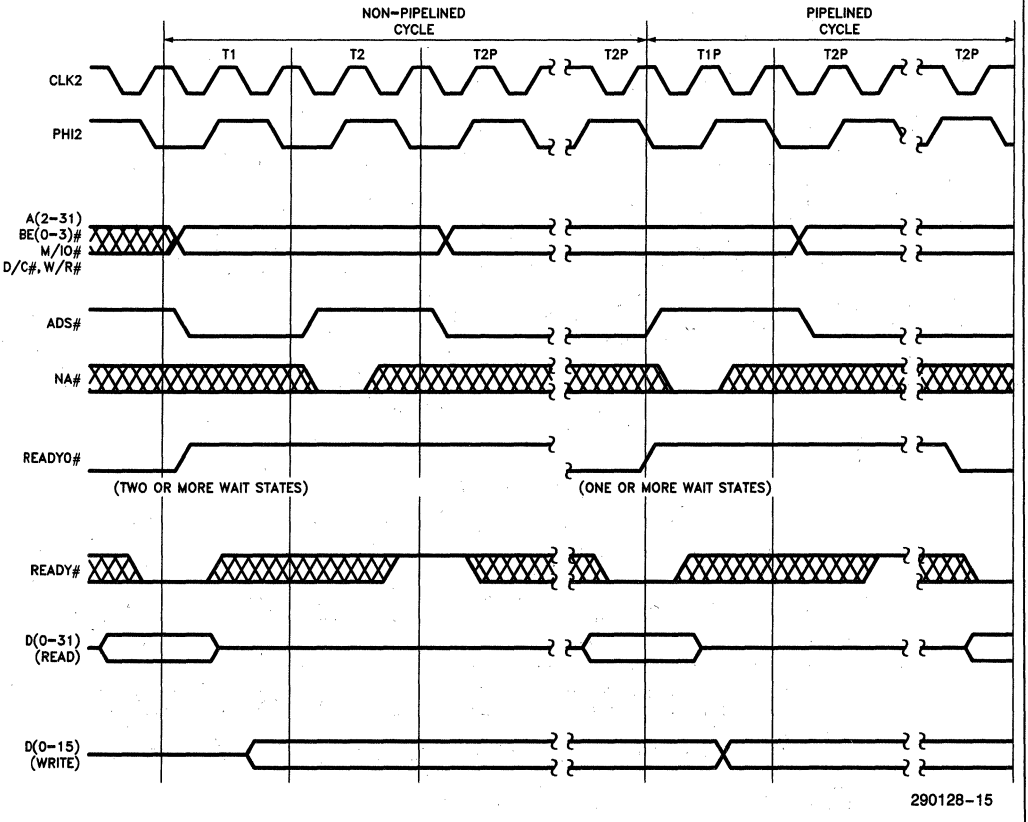


Figure 2-7. Pipelined Bus Cycles

2.3.3 SLAVE MODE BUS TIMING

Figure 2-8 shows the Slave Mode bus timing in both pipelined and non-pipelined cycles when the 82380 is being accessed. Recall that during Slave Mode, the 82380 will constantly monitor the ADS# and READY# signals to determine if the next cycle is pipelined. In Figure 2-8, the first cycle is non-pipelined and the second cycle is pipelined. In the pipelined cycle, the 82380 will start decoding the address and bus cycle signals one bus state earlier than in a non-pipelined cycle.

The READY# input signal is sampled by the 80386 host processor to determine the completion of a bus cycle. This occurs during the end of every T2 and T2P state. Normally, the output of the 82380 Wait State Generator, READYO#, is directly connected to the READY# input of the 80386 host processor and the 82380. In such case, READYO# and READY# will be identical (see Wait State Generator).



NOTE:
NA# is shown here only for timing reference. It is not sampled by the 82380 during Slave Mode.
When the 82380 registers are accessed, it will take one or more wait states in pipelined and two or more wait states in non-pipelined cycle to complete the internal access.

Figure 2-8. Slave Read/Write Timing

3.0 DMA Controller

The 82380 DMA Controller is capable of transferring data between any combination of memory and/or I/O, with any combination (8-, 16-, or 32-bits) of data path widths. Bus bandwidth is optimized through the use of an internal temporary register which can disassemble or assemble data to or from either an aligned or a non-aligned destination or source. Figure 3-1 is a block diagram of the 82380 DMA Controller.

The 82380 has eight channels of DMA. Each channel operates independently of the others. Within the operation of the individual channels, there are many different modes of data transfer available. Many of the operating modes can be intermixed to provide a very versatile DMA controller.

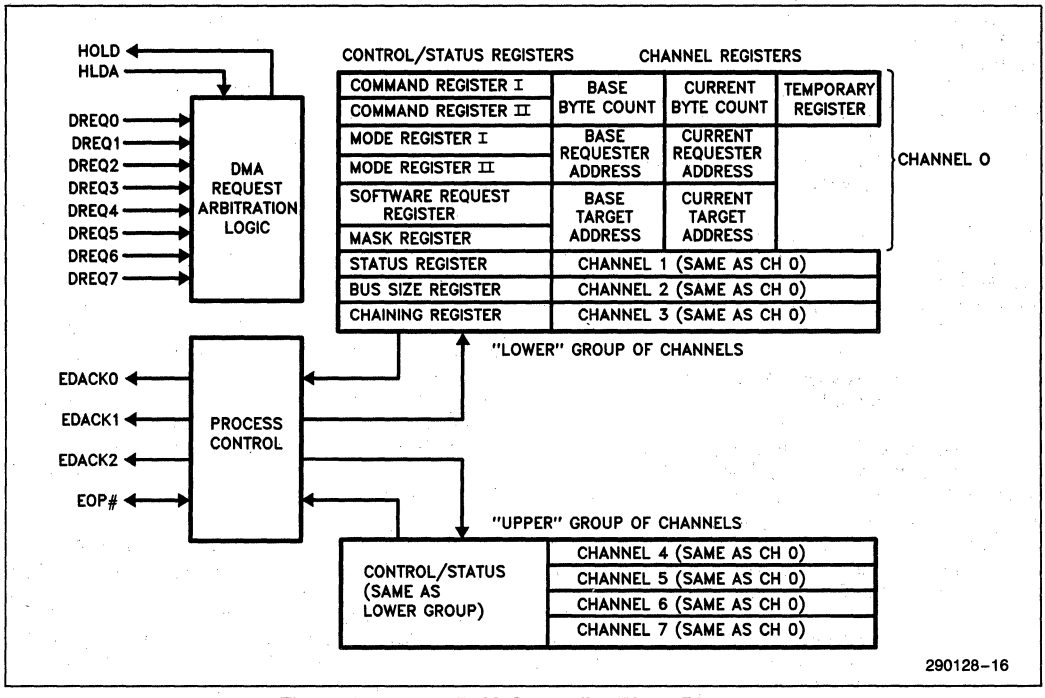


Figure 3-1. 82380 DMA Controller Block Diagram

3.1 Functional Description

In describing the operation of the 82380's DMA Controller, close attention to terminology is required. Before entering the discussion of the function of the 82380 DMA Controller, the following explanations of some of the terminology used herein may be of benefit. First, a few terms for clarification:

DMA PROCESS—A DMA process is the execution of a programmed DMA task from beginning to end. Each DMA process requires initial programming by the host 80386 microprocessor.

BUFFER—A contiguous block of data.

BUFFER TRANSFER—The action required by the DMA to transfer an entire buffer.

DATA TRANSFER—The DMA action in which a group of bytes, words, or double words are moved between devices by the DMA Controller. A data transfer operation may involve movement of one or many bytes.

BUS CYCLE—Access by the DMA to a single byte, word, or double word.

Each DMA channel consists of three major components. These components are identified by the contents of programmable registers which define the memory or I/O devices being serviced by the DMA. They are the Target, the Requester, and the Byte Count. They will be defined generically here and in greater detail in the DMA register definition section.

The Requester is the device which requires service by the 82380 DMA Controller, and makes the request for service. All of the control signals which the DMA monitors or generates for specific channels are logically related to the Requester. Only the Requester is considered capable of initiating or terminating a DMA process.

The Target is the device with which the Requester wishes to communicate. As far as the DMA process is concerned, the Target is a slave which is incapable of control over the process.

The direction of data transfer can be either from Requester to Target or from Target to Requester; i.e., each can be either a source or a destination.

The Requester and Target may each be either I/O or memory. Each has an address associated with it that can be incremented, decremented, or held constant. The addresses are stored in the Requester Address Registers and Target Address Registers, respectively. These registers have two parts: one which contains the current address being used in the DMA process (Current Address Register), and one which holds the programmed base address (Base Address Register). The contents of the Base Registers are never changed by the 82380 DMA Controller. The Current Registers are incremented or decremented according to the progress of the DMA process.

The Byte Count is the component of the DMA process which dictates the amount of data which must be transferred. Current and Base Byte Count Registers are provided. The Current Byte Count Register is decremented once for each byte transferred by the DMA process. When the register is decremented past zero, the Byte Count is considered 'expired' and the process is terminated or restarted, depending on the mode of operation of the channel. The point at which the Byte Count expires is called 'Terminal Count' and several status signals are dependent on this event.

Each channel of the 82380 DMA Controller also contains a 32-bit Temporary Register for use in assembling and disassembling non-aligned data. The operation of this register is transparent to the user, although the contents of it may affect the timing of some DMA handshake sequences. Since there is data storage available for each channel, the DMA Controller can be interrupted without loss of data.

The 82380 DMA Controller is a slave on the bus until a request for DMA service is received via either a software request command or a hardware request signal. The host processor may access any of the control/status or channel registers at any time the 82380 is a bus slave. Figure 3-2 shows the flow of operations that the DMA Controller performs.

At the time a DMA service request is received, the DMA Controller issues a bus hold request to the host processor. The 82380 becomes the bus master when the host relinquishes the bus by asserting a hold acknowledge signal. The channel to be serviced will be the one with the highest priority at the time the DMA Controller becomes the bus master. The DMA Controller will remain in control of the bus until the hold acknowledge signal is removed, or until the current DMA transfer is complete.

While the 82380 DMA Controller has control of the bus, it will perform the required data transfer(s). The type of transfer, source and destination addresses, and amount of data to transfer are programmed in the control registers of the DMA channel which received the request for service.

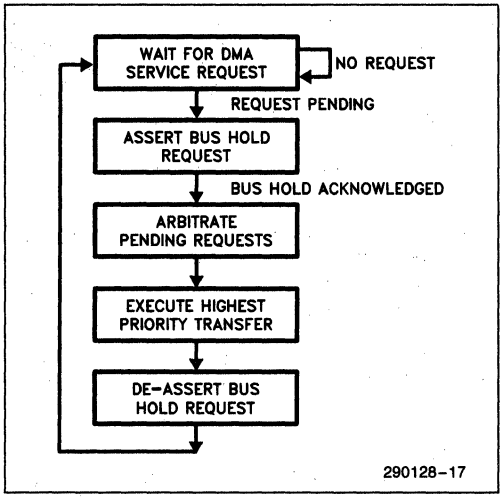


Figure 3-2. Flow of DMA Controller Operation

At completion of the DMA process, the 82380 will remove the bus hold request. At this time the 82380 becomes a slave again, and the host returns to being a master. If there are other DMA channels with requests pending, the controller will again assert the hold request signal and restart the bus arbitration and switching process.

3.2 Interface Signals

There are fourteen control signals dedicated to the DMA process. They include eight DMA Channel Requests (DREQn), three Encoded DMA Acknowledge signals (EDACKn), Processor Hold and Hold Acknowledge (HOLD, HLDA), and End-Of-Process (EOP#). The DREQn inputs and EDACK(0–2) outputs are handshake signals to the devices requiring DMA service. The HOLD output and HLDA input are handshake signals to the host processor. Figure 3-3 shows these signals and how they interconnect between the 82380 DMA Controller, and the Requester and Target devices.

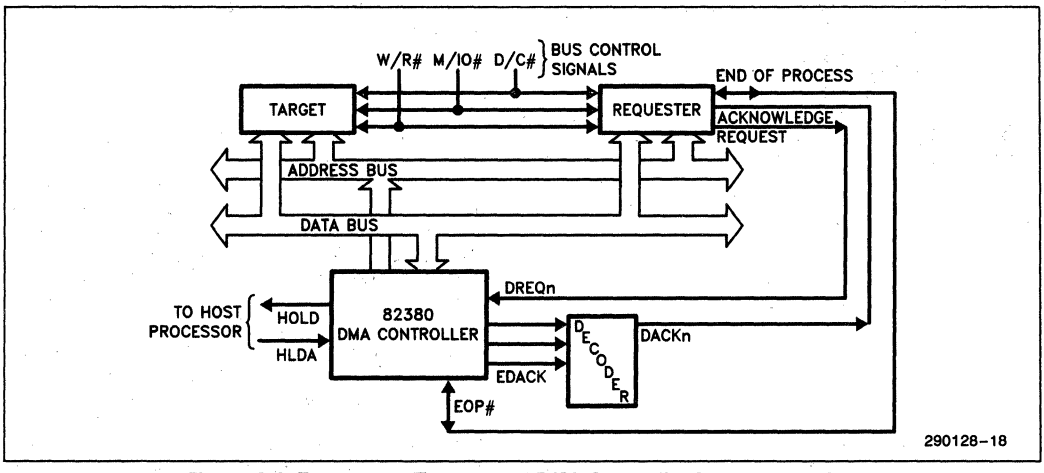


Figure 3-3. Requester, Target, and DMA Controller Interconnection (2-Cycle Configuration)

3.2.1 DREQn and EDACK(0–2)

These signals are the handshake signals between the peripheral and the 82380. When the peripheral requires DMA service, it asserts the DREQn signal of the channel which is programmed to perform the service. The 82380 arbitrates the DREQn against other pending requests and begins the DMA process after finishing other higher priority processes.

When the DMA service for the requested channel is in progress, the EDACK(0–2) signals represent the DMA channel which is accessing the Requester. The 3-bit code on the EDACK(0–2) lines indicates the number of the channel presently being serviced. Table 3-2 shows the encoding of these signals. Note that Channel 4 does not have a corresponding hardware acknowledge.

The DMA acknowledge (EDACK) signals indicate the active channel only during DMA accesses to the Requester. During accesses to the Target, EDACK(0–2) has the idle code (100). EDACK(0–2) can thus be used to select a Requester device during a transfer.

Table 3-2. EDACK Encoding During a DMA Transfer

EDACK2	EDACK1	EDACK0	Active Channel
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	Target Access
1	0	1	5
1	1	0	6
1	1	1	7

DREQn can be programmed as either an Asynchronous or Synchronous input. See section 3.4.1 for details on synchronous versus asynchronous operation of this pin.

The EDACKn signals are always active. They either indicate 'no acknowledge' or they indicate a bus access to the requester. The acknowledge code is either 100, for an idle DMA or during a DMA access to the Target, or 'n' during a Requester access, where n is the binary value representing the channel. A simple 3-line to 8-line decoder can be used to provide discrete acknowledge signals for the peripherals.

3.2.2 HOLD and HLDA

The Hold Request (HOLD) and Hold Acknowledge (HLDA) signals are the handshake signals between the DMA Controller and the host processor. HOLD is an output from the 82380 and HLDA is an input. HOLD is asserted by the DMA Controller when there is a pending DMA request, thus requesting the processor to give up control of the bus so the DMA process can take place. The 80386 responds by asserting HLDA when it is ready to relinquish control of the bus.

The 82380 will begin operations on the bus one clock cycle after the HLDA signal goes active. For this reason, other devices on the bus should be in the slave mode when HLDA is active.

HOLD and HLDA should not be used to gate or select peripherals requesting DMA service. This is because of the use of DMA-like operations by the DRAM Refresh Controller. The Refresh Controller is arbitrated with the DMA Controller for control of the bus, and refresh cycles have the highest priority. A refresh cycle will take place between DMA cycles without relinquishing bus control. See section 3.4.3 for a more detailed discussion of the interaction between the DMA Controller and the DRAM Refresh Controller.

3.2.3 EOP#

EOP# is a bi-directional signal used to indicate the end of a DMA process. The 82380 activates this as an output during the T2 states of the last Requester bus cycle for which a channel is programmed to execute. The Requester should respond by either withdrawing its DMA request, or interrupting the host processor to indicate that the channel needs to be programmed with a new buffer. As an input, this signal is used to tell the DMA Controller that the peripheral being serviced does not require any more data to be transferred. This indicates that the current buffer is to be terminated.

EOP# can be programmed as either an Asynchronous or a Synchronous input. See section 3.4.1 for details on synchronous versus asynchronous operation of this pin.

3.3 Modes of Operation

The 82380 DMA Controller has many independent operating functions. When designing peripheral interfaces for the 82380 DMA Controller, all of the functions or modes must be considered. All of the channels are independent of each other (except in priority of operation) and can operate in any of the modes. Many of the operating modes, though independently programmable, affect the operation of other modes. Because of the large number of com-

binations possible, each programmable mode is discussed here with its affects on the operation of other modes. The entire list of possible combinations will not be presented.

Table 3-1 shows the categories of DMA features available in the 82380. Each of the five major categories is independent of the others. The sub-categories are the available modes within the major function or mode category. The following sections explain each mode or function and its relation to other features.

Table 3-1. DMA Operating Modes

I. **Target/Requester Definition**
  a. Data Transfer Direction
  b. Device Type
  c. Increment/Decrement/Hold

II. **Buffer Processes**
  a. Single Buffer Process
  b. Buffer Auto-Initialize Process
  c. Buffer Chaining Process

III. **Data Transfer/Handshake Modes**
  a. Single Transfer Mode
  b. Demand Transfer Mode
  c. Block Transfer Mode
  d. Cascade Mode

IV. **Priority Arbitration**
  a. Fixed
  b. Rotating
  c. Programmable Fixed

V. **Bus Operation**
  a. Fly-By (Single-Cycle)/Two-Cycle
  b. Data Path Width
  c. Read, Write, or Verify Cycles

3.3.1 TARGET/REQUESTER DEFINITION

All DMA transfers involve three devices: the DMA Controller, the Requester, and the Target. Since the devices to be accessed by the DMA Controller vary widely, the operating characteristics of the DMA Controller must be tailored to the Requester and Target devices.

The Requester can be defined as either the source or the destination of the data to be transferred. This is done by specifying a Write or a Read transfer, respectively. In a Read transfer, the Target is the data source and the Requester is the destination for the data. In a Write transfer, the Requester is the source and the Target in the destination.

The Requester and Target addresses can each be independently programmed to be incremented, decremented, or held constant. As an example, the 82380 is capable of reversing a string or data by having a Requester address increment and the Target address decrement in a memory-to-memory transfer.

3.3.2 BUFFER TRANSFER PROCESSES

The 82380 DMA Controller allows three programmable Buffer Transfer Processes. These processes define the logical way in which a buffer of data is accessed by the DMA.

The three Buffer Transfer Processes include the Single Buffer Process, the Buffer Auto-Initialize Process, and the Buffer Chaining Process. These processes require special programming considerations. See the DMA Programming section for more details on setting up the Buffer Transfer Processes.

Single Buffer Process

The Single Buffer Process allows the DMA channel to transfer only one buffer of data. When the buffer has been completely transferred (Current Byte Count decremented past zero or EOP# input active), the DMA process ends and the channel becomes idle. In order for that channel to be used again, it must be reprogrammed.

The single Buffer Process is usually used when the amount of data to be transferred is known exactly, and it is also known that there is not likely to be any data to follow before the operating system can reprogram the channel.

Buffer Auto-Initialize Process

The Buffer Auto-Initialize Process allows multiple groups of data to be transferred to or from a single buffer. This process does not require reprogramming. The Current Registers are automatically reprogrammed from the Base Registers when the current process is terminated, either by an expired Byte Count or by an external EOP# signal. The data transferred will always be between the same Target and Requester.

The auto-initialization/process-execution cycle is repeated, with a HOLD/HLDA re-arbitration, until the channel is either disabled or re-programmed.

Buffer Chaining Process

The Buffer Chaining Process is useful for transferring large quantities of data into non-contiguous buffer areas. In this process, a single channel is used to process data from several buffers, while having to program the channel only once. Each new buffer is programmed in a pipelined operation that provides the new buffer information while the old buffer is being processed. The chain is created by loading new buffer information while the 82380 DMA Controller is processing the Current Buffer. When the Current Buffer expires, the 82380 DMA Controller automatically restarts the channel using the new buffer information.

Loading the new buffer information is done by an interrupt routine which is requested by the 82380. Interrupt Request 1 (IRQ1) is tied internally to the 82380 DMA Controller for this purpose. IRQ1 is generated by the 82380 when the new buffer information is loaded into the channel's Current Registers, leaving the Base Registers 'empty'. The interrupt service routine loads new buffer information into the Base Registers. The host processor is required to load the information for another buffer before the current Byte Count expires. The process repeats until the host programs the channel back to single buffer operation, or until the channel runs out of buffers.

The channel runs out of buffers when the Current Buffer expires and the Base Registers have not yet been loaded with new buffer information. When this occurs, the channel must be reprogrammed.

If an external EOP# is encountered while executing a Buffer Chaining Process, the current buffer is considered expired and the new buffer information is loaded into the Current Registers. If the Base Registers are 'empty', the chain is terminated.

The channel uses the Base Target Address Register as an indicator of whether or not the Base Registers are full. When the most significant byte of the Base Target Register is loaded, the channel considers all of the Base Registers loaded, and removes the interrupt request. This requires that the other Base Registers (Base Requester Address, Last Byte Count) must be loaded before the Base Target Address Register. The reason for implementing the reloading process this way is that, for most applications, the Byte Count and the Requester will not change from one buffer to the next, and therefore do not need to be reprogrammed. The details of programming the channel for the Buffer Chaining Process can be found in the section of DMA programming.

3.3.3 DATA TRANSFER MODES

Three Data Transfer modes are available in the 82380 DMA Controller. They are the Single Transfer, Block Transfer, and Demand Transfer Modes. These transfer modes can be used in conjunction with any one of three Buffer Transfer modes: Single Buffer, Auto-Initialized Buffer, and Buffer Chaining. Any Data Transfer Modes can be used under any of the Buffer Transfer Modes. These modes are independently available for all DMA channels.

Different devices being serviced by the DMA Controller require different handshaking sequences for data transfers to take place. Three handshaking modes are available on the 82380, giving the designer the opportunity to use the DMA Controller as efficiently as possible. The speed at which data can be presented or read by a device can affect the way a DMA controller uses the host's bus, thereby affecting not only data throughput during the DMA process, but also affecting the host's performance by limiting its access to the bus.

Single Transfer Mode

In the Single Transfer Mode, one data transfer to or from the Requester is performed by the DMA Controller at a time. The DREQn input is arbitrated and the HOLD/HLDA sequence is executed for each transfer. Transfers continue in this manner until the Byte Count expires, or until EOP# is sampled active. If the DREQn input is held active continuously, the entire DREQ-HOLD-HLDA-DACK sequence is repeated over and over until the programmed number of bytes has been transferred. Bus control is released to the host between each transfer. Figure 3-4 shows the logical flow of events which make up a buffer transfer using the Single Transfer Mode. Refer to section 3.4 for an explanation of the bus control arbitration procedure.

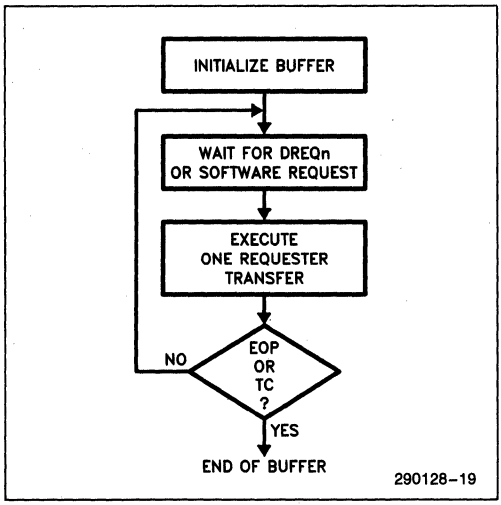


Figure 3-4. Buffer Transfer in Single Transfer Mode

The Single Transfer Mode is used for devices which require complete handshake cycles with each data access. Data is transferred to or from the Requester only when the Requester is ready to perform the transfer. Each transfer requires the entire DREQ-HOLD-HLDA-DACK handshake cycle. Figure 3-5 shows the timing of the Single Transfer Mode cycles.

Block Transfer Mode

In the Block Transfer Mode, the DMA process is initiated by a DMA request and continues until the Byte count expires, or until EOP# is activated by the Requester. The DREQn signal need only be held active until the first Requester access. Only a refresh cycle will interrupt the block transfer process.

Figure 3-6 illustrates the operation of the DMA during the Block Transfer Mode. Figure 3-7 shows the timing of the handshake signals during Block Mode Transfers.

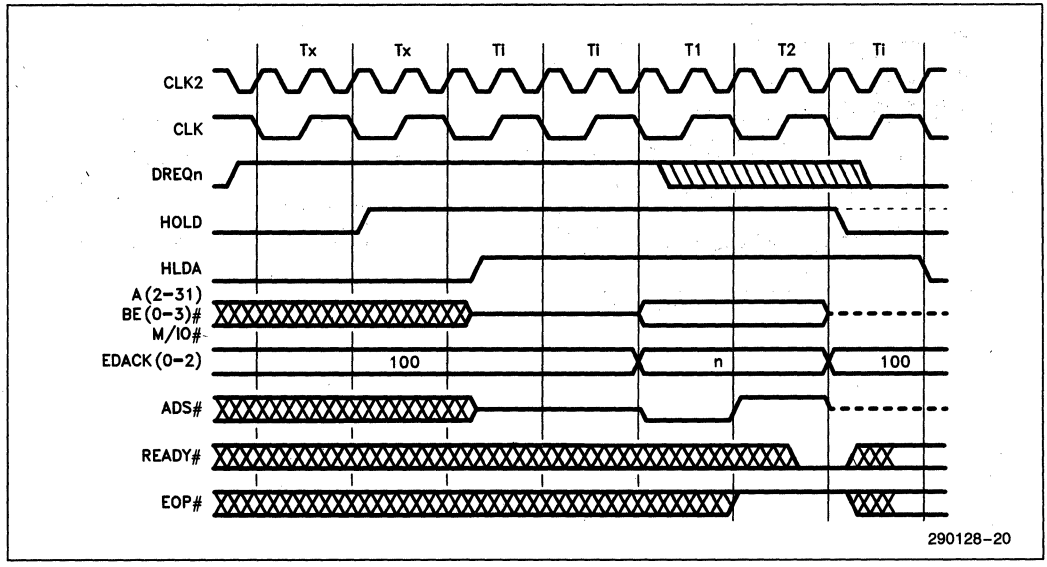


Figure 3-5. DMA Single Transfer Mode

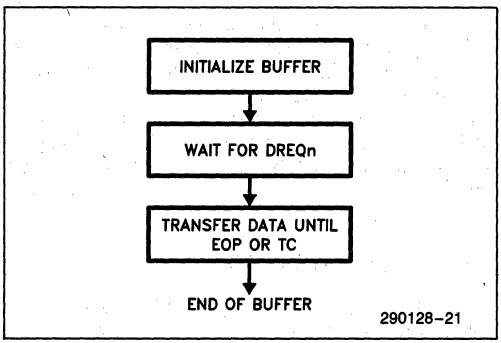


Figure 3-6. Buffer Transfer in Block Transfer Mode

Demand Transfer Mode

The Demand Transfer Mode provides the most flexible handshaking procedures during the DMA process. A Demand Transfer is initiated by a DMA request. The process continues until the Byte Count expires, or an external EOP# is encountered. If the device being serviced (Requester) desires, it can interrupt the DMA process by de-activating the DREQn line. Action is taken on the condition of DREQn during Requester accesses only. The access during which DREQn is sampled inactive is the last Requester access which will be performed during the current transfer. Figure 3-8 shows the flow of events during the transfer of a buffer in the Demand Mode.

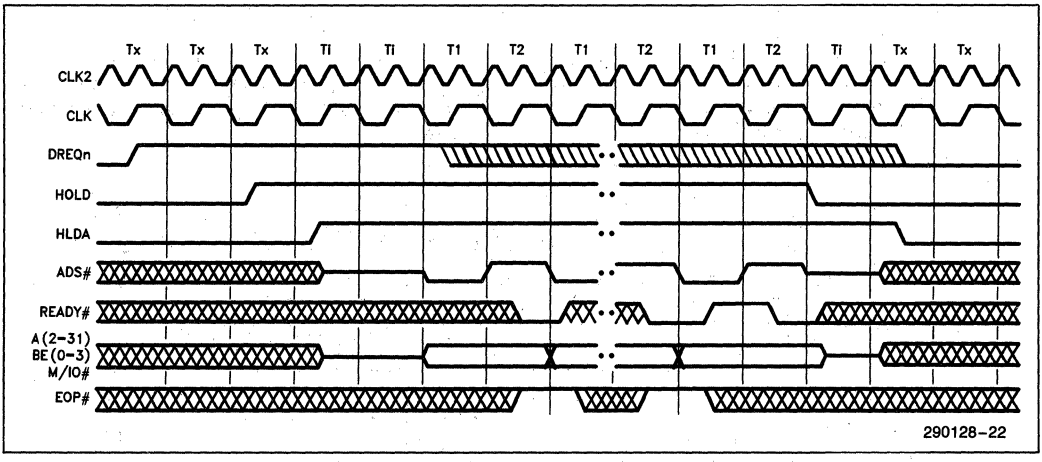


Figure 3-7. Block Mode Transfers

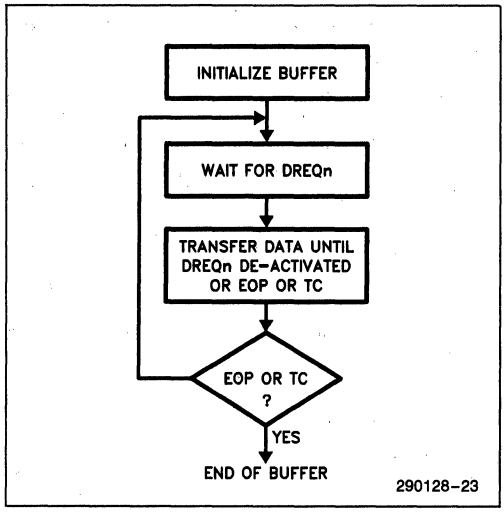


Figure 3-8. Buffer Transfer In Demand Transfer Mode

When the DREQn line goes inactive, the DMA controller will complete the current transfer, including any necessary accesses to the Target, and relinquish control of the bus to the host. The current process information is saved (byte count, Requester and Target addresses, and Temporary Register).

The Requester can restart the transfer process by reasserting DREQn. The 82380 will arbitrate the request with other pending requests and begin the process where it left off. Figure 3-9 shows the timing of handshake signals during Demand Transfer Mode operation.

Using the Demand Transfer Mode allows peripherals to access memory in small, irregular bursts without wasting bus control time. The 82380 is designed to give the best possible bus control latency in the Demand Transfer Mode. Bus control latency is defined here as the time from the last active bus cycle of the previous bus master to the first active bus cycle of the new bus master. The 82380 DMA Controller will perform its first bus access cycle two bus states after HLDA goes active. In the typical configuration, bus control is returned to the host one bus state after the DREQn goes inactive.

There are two cases where there may be more than one bus state of bus control latency at the end of a transfer. The first is at the end of an Auto-Initialize process, and the second is at the end of a process where the source is the Requester and Two-Cycle transfers are used.

When a Buffer Auto-Initialize Process is complete, the 82380 requires seven bus states to reload the

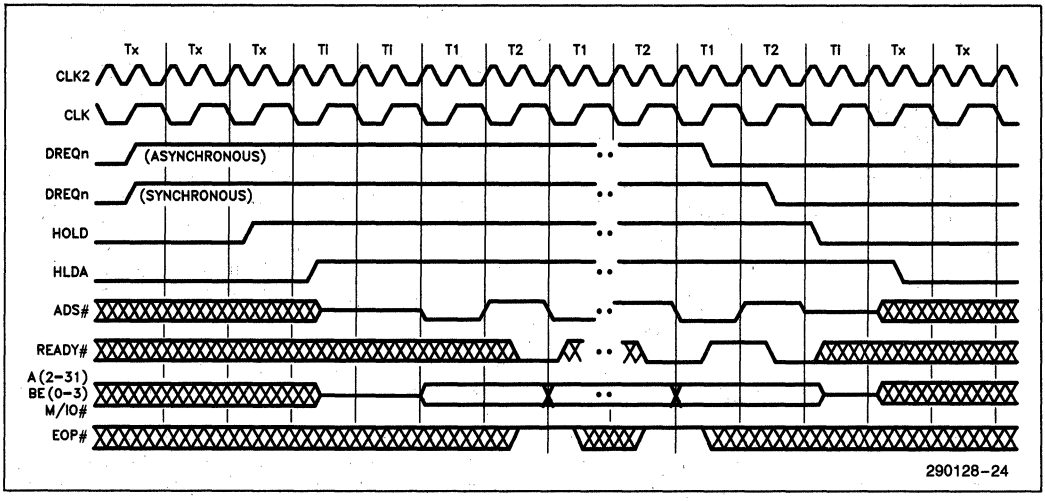


Figure 3-9. Demand Mode Transfers

Current Registers from the Base Registers of the Auto-Initialized channel. The reloading is done while the 82380 is still the bus master so that it is prepared to service the channel immediately after relinquishing the bus, if necessary.

In the case where the Requester is the source, and Two-Cycle transfers are being used, there are two extra idle states at the end of the transfer process. This occurs due to housekeeping in the DMA's internal pipeline. These two idle states are present only after the very last Requester access, before the DMA Controller de-activates the HOLD signal.

3.3.4 CHANNEL PRIORITY ARBITRATION

DMA channel priority can be programmed into one of two arbitration methods: Fixed or Rotating. The four lower DMA channels and the four upper DMA channels operate as if they were two separate DMA controllers operating in cascade. The lower group of four channels (0–3) is always prioritized between channels 7 and 4 of the upper group of channels (4–7). Figure 3-10 shows a pictorial representation of the priority grouping.

The priority can thus be set up as rotating for one group of channels and fixed for the other, or any other combination. While in Fixed Priority, the programmer can also specify which channel has the lowest priority.

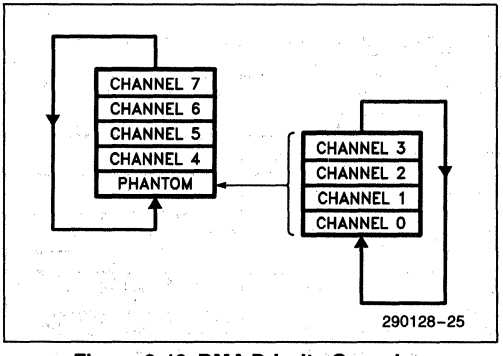


Figure 3-10. DMA Priority Grouping

The 82380 DMA Controller defaults to Fixed Priority. Channel 0 has the highest priority, then 1, 2, 3, 4, 5, 6, 7. Channel 7 has the lowest priority. Any time the DMA Controller arbitrates DMA requests, the requesting channel with the highest priority will be serviced next.

Fixed Priority can be entered into at any time by a software command. The priority levels in effect after the mode switch are determined by the current setting of the Programmable Priority.

Programmable Priority is available for fixing the priority of the DMA channels within a group to levels other than the default. Through a software command, the channel to have the lowest priority in a group can be specified. Each of the two groups of four channels can have the priority fixed in this way. The other channels in the group will follow the natural Fixed Priority sequence. This mode affects only the priority levels while operating with Fixed Priority.

For example, if channel 2 is programmed to have the lowest priority in its group, channel 3 has the highest priority. In descending order, the other channels would have the following priority: (3, 0, 1, 2), 4, 5, 6, 7 (channel 2 lowest, channel 3 highest). If the upper group were programmed to have channel 5 as the lowest priority channel, the priority would be (again, highest to lowest): 6, 7, (3, 0, 1, 2), 4, 5. Figure 3-11 shows this example pictorially. The lower group is always prioritized as a fifth channel of the upper group (between channels 4 and 7).

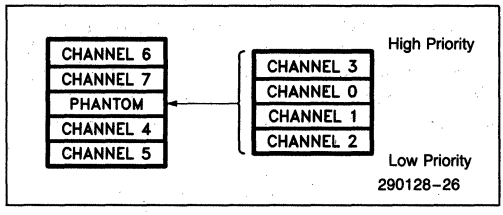


Figure 3-11. Example of Programmed Priority

The DMA Controller will only accept Programmable Priority commands while the addressed group is operating in Fixed Priority. Switching from Fixed to Rotating Priority preserves the current priority levels. Switching from Rotating to Fixed Priority returns the priority levels to those which were last programmed by use of Programmable Priority.

Rotating Priority allows the devices using DMA to share the system bus more evenly. An individual channel does not retain highest priority after being serviced, priority is passed to the next highest priority channel in the group. The channel which was most recently serviced inherits the lowest priority. This rotation occurs each time a channel is serviced. Figure 3-12 shows the sequence of events as priority is passed between channels. Note that the lower group rotates within the upper group, and that servicing a channel within the lower group causes rotation within the group as well as rotation of the upper group.

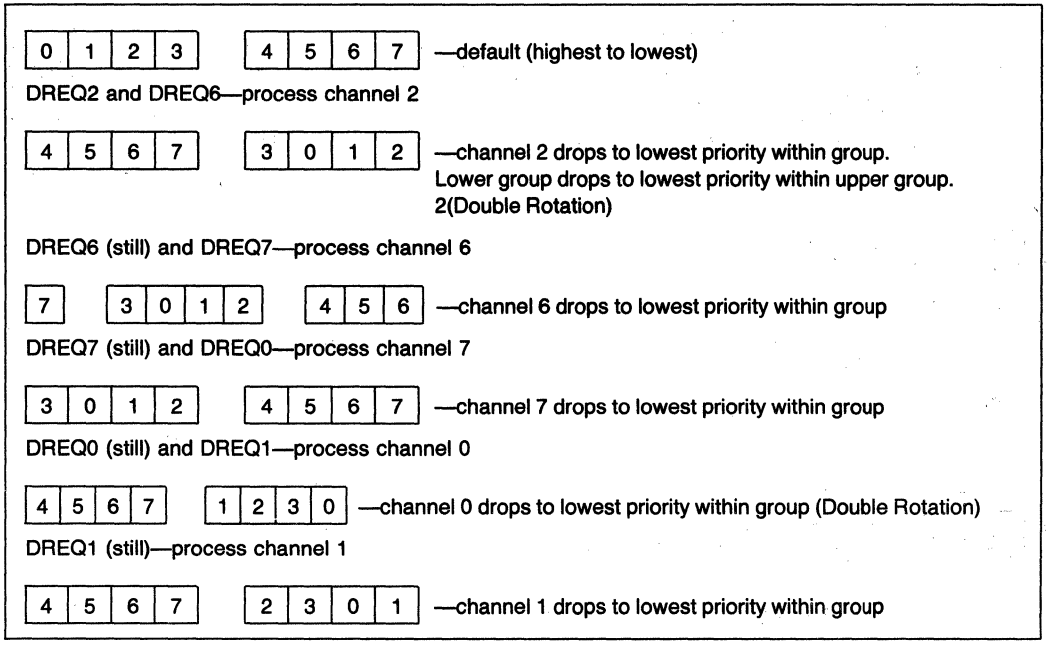


Figure 3-12. Rotating Channel Priority. Lower and Upper groups are programmed for the Rotating Priority Mode.

3.3.5 COMBINING PRIORITY MODES

Since the DMA Controller operates as two four-channel controllers in cascade, the overall priority scheme of all eight channels can take on a variety of forms. There are four possible combinations of priority modes between the two groups of channels: Fixed Priority only (default), Fixed Priority upper group/Rotating Priority lower group, Rotating Priority upper group/Fixed Priority lower group, and Rotating Priority only. Figure 3-13 illustrates the operation of the two combined priority methods.

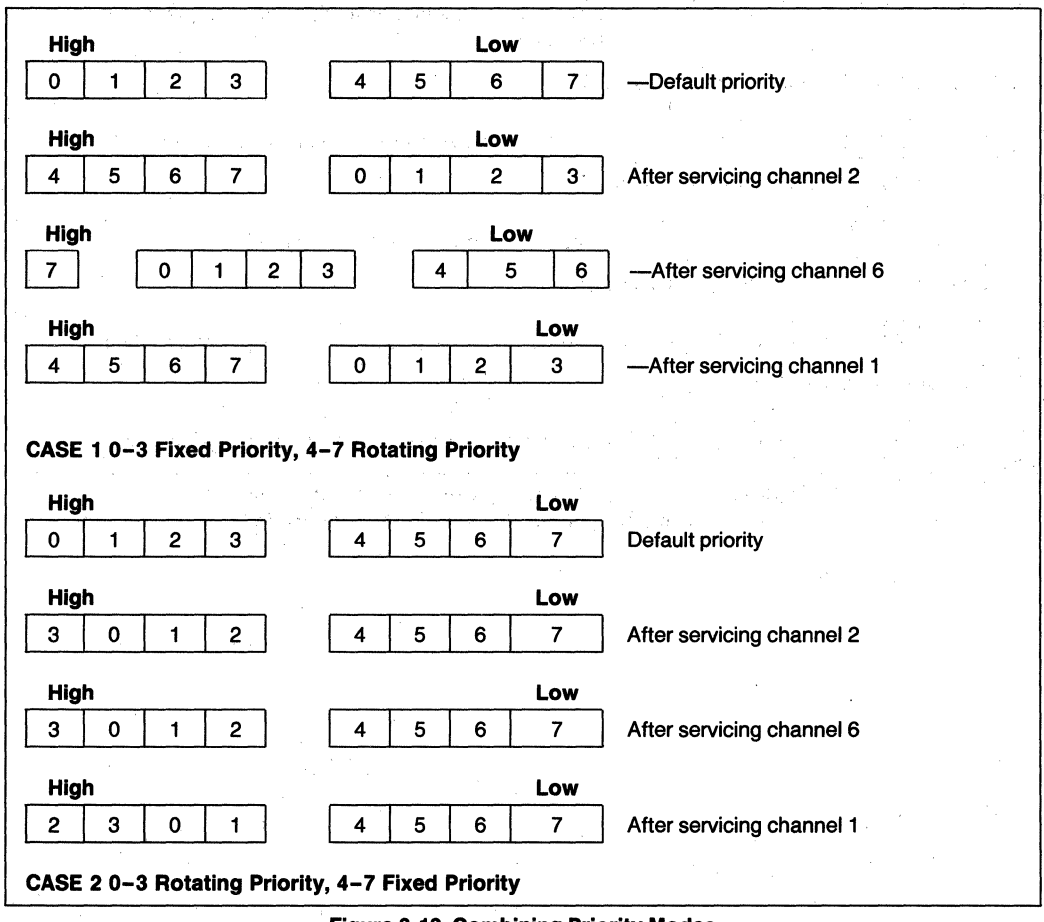


Figure 3-13. Combining Priority Modes

3.3.6 BUS OPERATION

Data may be transferred by the DMA Controller using two different bus cycle operations: Fly-By (one-cycle) and Two-Cycle. These bus handshake methods are selectable independently for each channel through a command register. Device data path widths are independently programmable for both Target and Requester. Also selectable through software is the direction of data transfer. All of these parameters affect the operation of the 82380 on a bus-cycle by bus-cycle basis.

3.3.6.1 Fly-By Transfers

The Fly-By Transfer Mode is the fastest and most efficient way to use the 82380 DMA Controller to transfer data. In this method of transfer, the data is written to the destination device at the same time it is read from the source. Only one bus cycle is used to accomplish the transfer.

In the Fly-By Mode, the DMA acknowledge signal is used to select the Requester. The DMA Controller simultaneously places the address of the Target on the address bus. The state of M/IO# and W/R# during the Fly-By transfer cycle indicate the type of Target and whether the target is being written to or read from. The Target's Bus Size is used as an incrementer for the Byte Count. The Requester address registers are ignored during Fly-By transfers.

Note that memory-to-memory transfers cannot be done using the Fly-By Mode. Only one memory or I/O address is generated by the DMA Controller at a time during Fly-By transfers. Only one of the devices being accessed can be selected by an address. Also, the Fly-By method of data transfer limits the hardware to accesses of devices with the same data bus width. The Temporary Registers are not affected in the Fly-By Mode.

Fly-By transfers also require that the data paths of the Target and Requester be directly connected. This requires that successive Fly-By accesses be to doubleword boundaries, or that the Requester be capable of switching its connections to the data bus.

3.3.6.2 Two-Cycle Transfers

Two-Cycle transfers can also be performed by the 82380 DMA Controller. These transfers require at least two bus cycles to execute. The data being transferred is read into the DMA Controller's Temporary Register during the first bus cycle(s). The second bus cycle is used to write the data from the Temporary Register to the destination.

If the addresses of the data being transferred are not word or doubleword aligned, the 82380 will recognize the situation and read and write the data in groups of bytes, placing them always at the proper destination. This process of collecting the desired bytes and putting them together is called 'byte assembly'. The reverse process (reading from aligned locations and writing to non-aligned locations) is called 'byte disassembly'.

The assembly/disassembly process takes place transparent to the software, but can only be done while using the Two-Cycle transfer method. The 82380 will always perform the assembly/disassembly process as necessary for the current data transfer. Any data path widths for either the Requester or Target can be used in the Two-Cycle Mode. This is very convenient for interfacing existing 8- and 16-bit peripherals to the 80386's 32-bit bus.

The 82380 DMA Controller always attempts to fill the Temporary Register from the source before writing any data to the destination. If the process is terminated before the Temporary Register is filled (TC or EOP#), the 82380 will write the partial data to the destination. If a process is temporarily suspended (such as when DREQn is de-activated during a demand transfer), the contents of a partially filled Temporary Register will be stored within the 82380 until the process is restarted.

For example, if the source is specified as an 8-bit device and the destination as a 32-bit device, there will be four reads as necessary from the 8-bit source to fill the Temporary Register. Then the 82380 will write the 32-bit contents to the destination. This cycle will repeat until the process is terminated or suspended.

Note that for a Single-Cycle transfer mode of operation (see section 3.3.3), the internal circuitry of the DMA Controller actually executes single transfers by removing the DREQ from the internal arbitration. Thus single transfers from an 8-bit requester to a 32-bit target will consist of four complete and independent 8-bit requester cycles, between which bus control is released and re-requested. Finally, the 32-bit data will be transferred to the target device from the temporary register before the fifth requester cycle.

With Two-Cycle transfers, the devices that the 82380 accesses can reside at any address within I/O or memory space. The device must be able to decode the byte-enables (BEn#). Also, if the device cannot accept data in byte quantities, the programmer must take care not to allow the DMA Controller to access the device on any address other than the device boundary.

3.3.6.3 Data Path Width and Data Transfer Rate Considerations

The number of bus cycles used to transfer a single 'word' of data is affected by whether the Two-Cycle or the Fly-By (Single-Cycle) transfer method is used.

The number of bus cycles used to transfer data directly affects the data transfer rate. Inefficient use of bus cycles will decrease the effective data transfer rate that can be obtained. Generally, the data transfer rate is halved by using Two-Cycle transfers instead of Fly-By transfers.

The choice of data path widths of both Target and Requester affects the data transfer rate also. During each bus cycle, the largest pieces of data possible should be transferred.

The data path width of the devices to be accessed must be programmed into the DMA controller. The 82380 defaults after reset to 8-bit-to-8-bit data transfers, but the Target and Requester can have different data path widths, independent of each other and independent of the other channels. Since this is a software programmable function, more discussion of the uses of this feature are found in the section on programming.

3.3.6.4 Read, Write, and Verify Cycles

Three different bus cycle types may be used in a data transfer. They are the Read, Write, and Verify cycles. These cycle types dictate the way in which the 82380 operates on the data to be transferred.

A Read Cycle transfers data from the Target to the Requester. A Write Cycle transfers data from the Requester to the target. In a Fly-By transfer, the address and bus status signals indicate the access (read or write) to the Target; the access to the Requester is assumed to be the opposite.

The Verify Cycle is used to perform a data read only. No write access is indicated or assumed in a Verify Cycle. The Verify Cycle is useful for validating block fill operations. An external comparator must be provided to do any comparisons on the data read.

3.4 Bus Arbitration and Handshaking

Figure 3-14 shows the flow of events in the DMA request arbitration process. The arbitration sequence starts when the Requester asserts a DREQn (or DMA service is requested by software). Figure 3-15 shows the timing of the sequence of events following a DMA request. This sequence is executed for each channel that is activated. The DREQn signal can be replaced by a software DMA channel request with no change in the sequence.

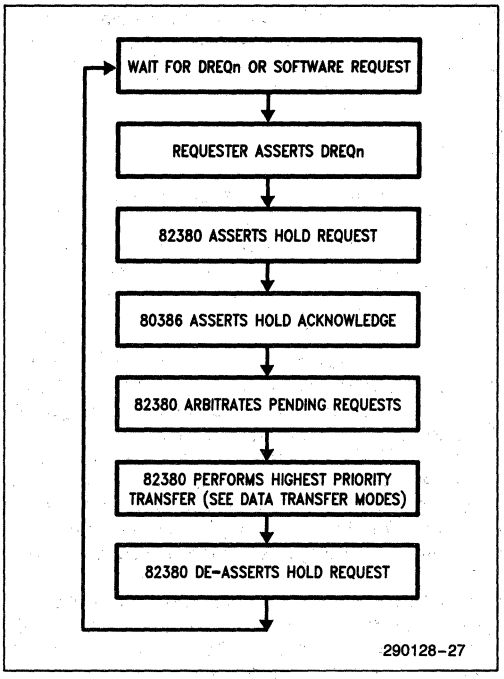


Figure 3-14. Bus Arbitration and DMA Sequence

After the Requester asserts the service request, the 82380 will request control of the bus via the HOLD signal. The 82380 will always assert the HOLD signal one bus state after the service request is asserted. The 80386 responds by asserting the HLDA signal, thus releasing control of the bus to the 82380 DMA Controller.

Priority of pending DMA service requests is arbitrated during the first state after HLDA is asserted by the 80386. The next state will be the beginning of the first transfer access of the highest priority process.

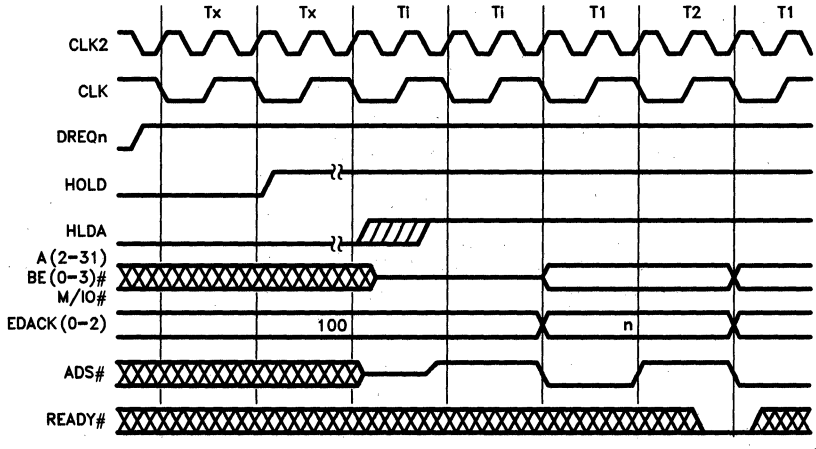
When the 82380 DMA Controller is finished with its current bus activity, it returns control of the bus to the host processor. This is done by driving the HOLD signal inactive. The 82380 does not drive any address or data bus signals after HOLD goes low. It enters the Slave Mode until another DMA process is requested. The processor acknowledges that it has regained control of the bus by forcing the HLDA signal inactive. Note that the 82380's DMA Controller will not re-request control of the bus until the entire HOLD/HLDA handshake sequence is complete.

The 82380 DMA Controller will terminate a current DMA process for one of three reasons: expired byte count, end-of-process command (EOP# activated) from a peripheral, or de-activated DMA request signal. In each case, the controller will de-assert HOLD immediately after completing the data transfer in progress. These three methods of process termination are illustrated in Figures 3-16, 3-19, and 3-18, respectively.

An expired byte count indicates that the current process is complete as programmed and the channel has no further transfers to process. The channel must be restarted according to the currently programmed Buffer Transfer Mode, or reprogrammed completely, including a new Buffer Transfer Mode.

If the peripheral activates the EOP# signal, it is indicating that it will not accept or deliver any more data for the current buffer. The 82380 DMA Controller considers this as a completion of the channel's current process and interprets the condition the same way as if the byte count expired.

The action taken by the 82380 DMA Controller in response to a de-activated DREQn signal depends on the Data Transfer Mode of the channel. In the Demand Mode, data transfers will take place as long as the DREQn is active and the byte count has not expired. In the Block Mode, the controller will complete the entire block transfer without relinquishing



NOTE:
Channel priority resolution takes place during the bus state before HLDA is asserted, allowing the DMA Controller to respond to HLDA without extra idle bus states.

Figure 3-15. Beginning of a DMA process

the bus, even if DREQn goes inactive before the transfer is complete. In the Single Mode, the controller will execute single data transfers, relinquishing the bus between each transfer, as long as DREQn is active.

Normal termination of a DMA process due to expiration of the byte count (Terminal Count-TC) is shown in Figure 3-16. The condition of DREQn is ignored until after the process is terminated. If the channel is programmed to auto-initialize, HOLD will be held active for an additional seven clock cycles while the auto-initialization takes place.

Table 3-3 shows the DMA channel activity due to EOP# or Byte Count expiring (Terminal Count).

Buffer Process:	Single or Chaining-Base Empty		Auto-Initialize		Chaining-Base Loaded	
Event						
Terminal Count	True	X	True	X	True	X
EOP# Input	X	0	X	0	X	0
Results						
Current Registers	—	—	Load	Load	Load	Load
Channel Mask	Set	Set	—	—	—	—
EOP# Output	0	X	0	X	1	X
Terminal Count Status	Set	Set	Set	Set	—	—
Software Request	CLR	CLR	CLR	CLR	—	—

Table 3-3. DMA Channel Activity Due to Terminal Count or External EOP#

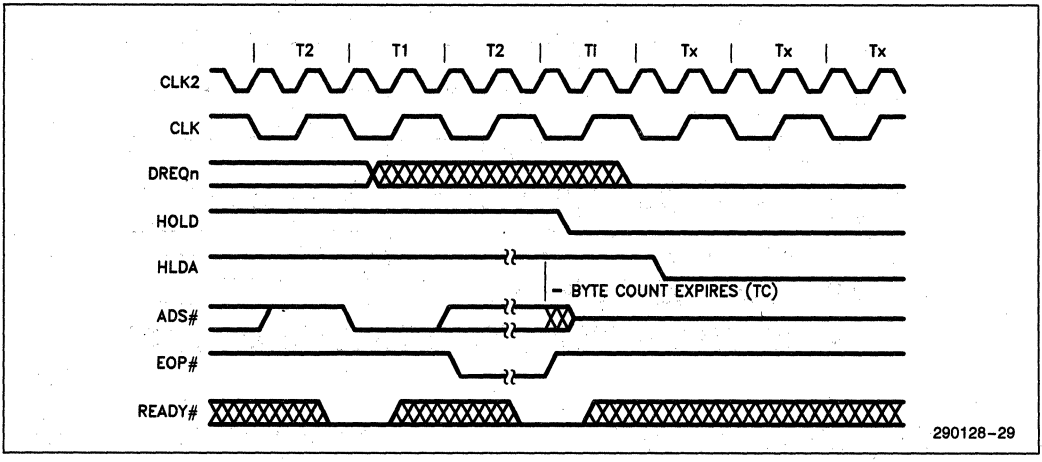


Figure 3-16. Termination of a DMA Process Due to Expiration of Current Byte Count

The 82380 always relinquishes control of the bus between channel services. This allows the hardware designer the flexibility to externally arbitrate bus hold requests, if desired. If another DMA request is pending when a higher priority channel service is completed, the 82380 will relinquish the bus until the hold acknowledge is inactive. One bus state after the HLDA signal goes inactive, the 82380 will assert HOLD again. This is illustrated in Figure 3-17.

3.4.1 SYNCHRONOUS AND ASYNCHRONOUS SAMPLING OF DREQn AND EOP#

As an indicator that a DMA service is to be started, DREQn is always sampled asynchronously. It is sampled at the beginning of a bus state and acted upon at the end of the state. Figure 3-15 illustrates the start of a DMA process due to a DREQn input.

The DREQn and EOP# inputs can be programmed to be sampled either synchronously or asynchronously to signal the end of a transfer.

The synchronous mode affords the Requester one bus state of extra time to react to an access. This means the Requester can terminate a process on the current access, without losing any data. The asynchronous mode requires that the input signal be presented prior to the beginning of the last state of the Requester access.

The timing relationships of the DREQn and EOP# signals to the termination of a DMA transfer are shown in Figures 3-18 and 3-19. Figure 3-18 shows the termination of a DMA transfer due to inactive DREQn. Figure 3-19 shows the termination of a DMA process due to an active EOP# input.

In the Synchronous Mode, DREQn and EOP# are sampled at the end of the last state of every Requester data transfer cycle. If EOP# is active or DREQn is inactive at this time, the 82380 recognizes this access to the Requester as the last transfer. At this point, the 82380 completes the transfer in progress, if necessary, and returns bus control to the host.

In the asynchronous mode, the inputs are sampled at the beginning of every state of a Requester access. The 82380 waits until the end of the state to act on the input.

DREQn and EOP# are sampled at the latest possible time when the 82380 can determine if another transfer is required. In the Synchronous Mode, DREQn and EOP# are sampled on the trailing edge of the last bus state before another data access cycle begins. The Asynchronous Mode requires that the signals be valid one clock cycle earlier.

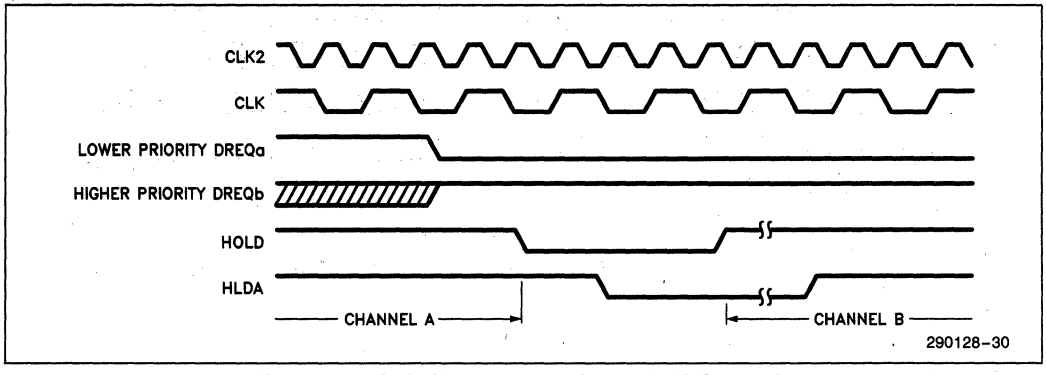


Figure 3-17. Switching between Active DMA Channels

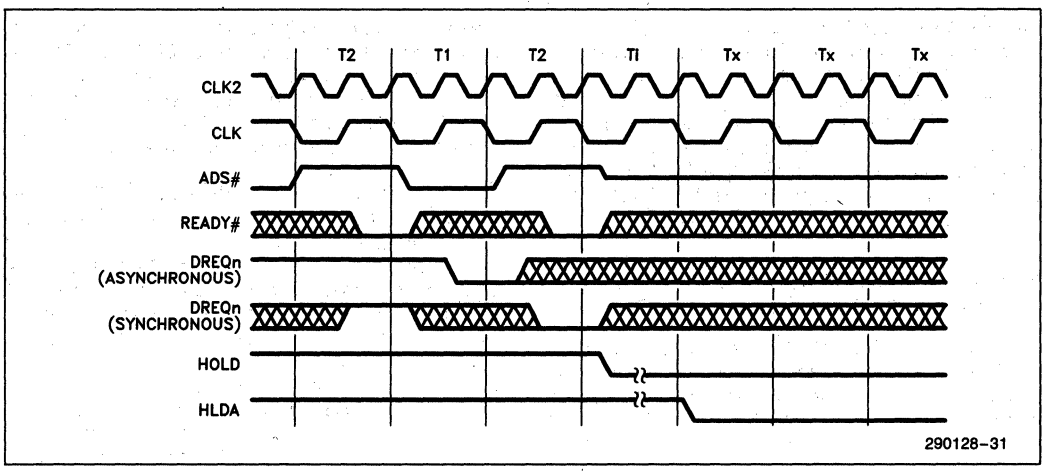


Figure 3-18. Termination of a DMA Process Due to De-Asserting DREQn

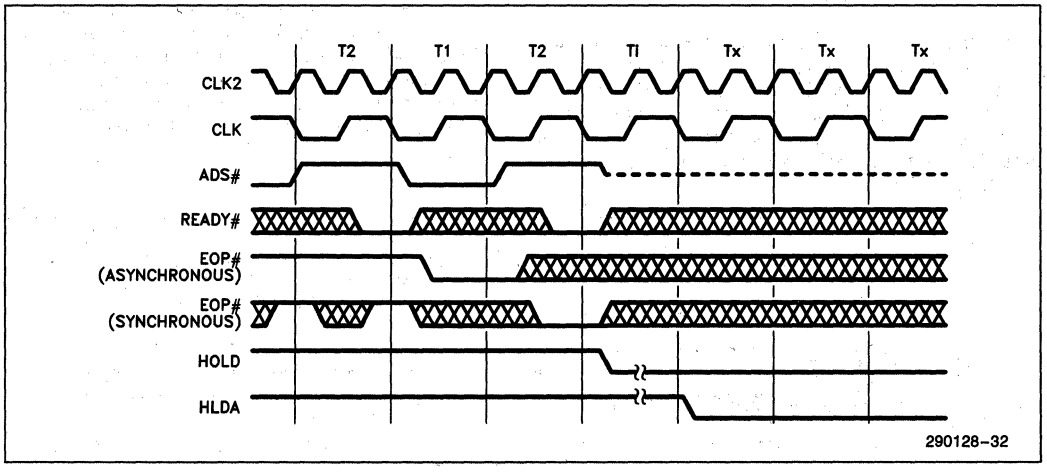


Figure 3-19. Termination of a DMA Process Due to an External EOP#

While in the Pipeline Mode, if the NA# signal is sampled active during a transfer, the end of the state where NA# was sampled active is when the 82380 decides whether to commit to another transfer. The device must de-assert DREQn or assert EOP# before NA# is asserted, otherwise the 82380 will commit to another, possibly undesired, transfer.

Synchronous DREQn and EOP# sampling allows the peripheral to prevent the next transfer from occurring by de-activating DREQn or asserting EOP# during the current Requester access, before the 82380 DMA Controller commits itself to another transfer. The DMA Controller will not perform the next transfer if it has not already begun the bus cycle. Asynchronous sampling allows less stringent timing requirements than the Synchronous Mode, but requires that the DREQn signal be valid at the beginning of the next to last bus state of the current Requester access.

Using the Asynchronous Mode with zero wait states can be very difficult. Since the addresses and control signals are driven by the 82380 near half-way through the first bus state of a transfer, and the Asynchronous Mode requires that DREQn be active before the end of the state, the peripheral being accessed is required to present DREQn only a few nanoseconds after the control information is available. This means that the peripheral's control logic must be extremely fast (practically non-causal). An alternative is the Synchronous Mode.

3.4.2 ARBITRATION OF CASCADED MASTER REQUESTS

The Cascade Mode allows another DMA-type device to share the bus by arbitrating its bus accesses with the 82380's. Seven of the eight DMA channels (0–3 and 5–7) can be connected to a cascaded device. The cascaded device requests bus control through the DREQn line of the channel which is programmed to operate in Cascade Mode. Bus hold acknowledge is signaled to the cascaded device through the EDACK lines. When the EDACK lines are active with the code for the requested cascade channel, the bus is available to the cascaded master device.

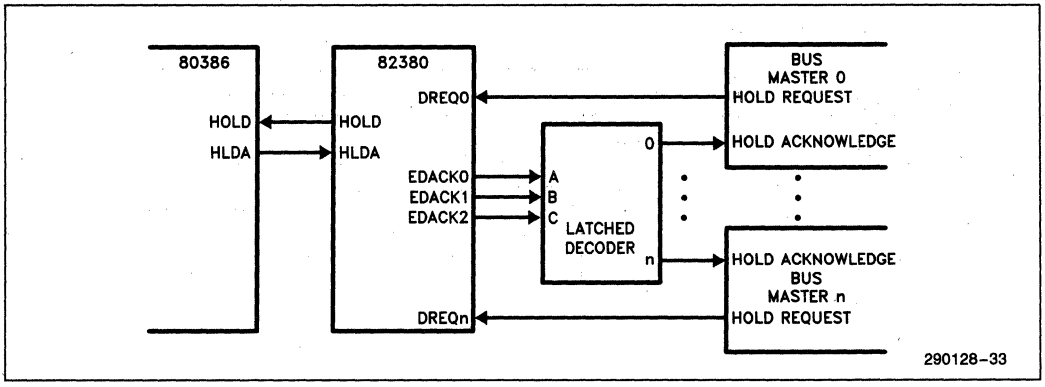


Figure 3-20. Cascaded Bus Master

A Cascade cycle begins the same way a regular DMA cycle begins. The requesting bus master asserts the DREQn line on the 82380. This bus control request arbitrated as any other DMA request would be. If any channel receives a DMA request, the 82380 requests control of the bus. When the host acknowledges that it has released bus control, the 82380 acknowledges to the requesting master that it may access the bus. The 82380 enters an idle state until the new master relinquishes control.

A cascade cycle will be terminated by one of two events: DREQn going inactive, or HLDA going inactive. The normal way to terminate the cascade cycle is for the cascaded master to drop the DREQn signal. Figure 3-21 shows the two cascade cycle termination sequences.

The Refresh Controller may interrupt the cascaded master to perform a refresh cycle. If this occurs, the 82380 DMA Controller will de-assert the EDACK signal (hold acknowledge to cascaded master) and wait for the cascaded master to remove its hold request. When the 82380 regains bus control, it will perform the refresh cycle in its normal fashion. After the refresh cycle has been completed, and if the cascaded device has re-asserted its request, the 82380 will return control to the cascaded master which was interrupted.

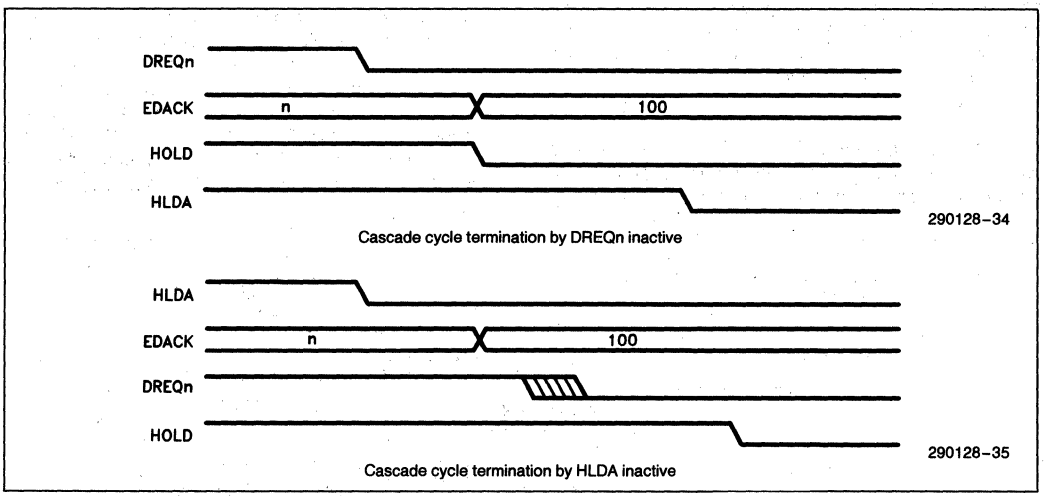


Figure 3-21. Cascade Cycle Termination

The 82380 assumes that it is the only device monitoring the HLDA signal. If the system designer wishes to place other devices on the bus as bus masters, the HLDA from the processor must be intercepted before presenting it to the 82380. Using the Cascade capability of the 82380 DMA Controller offers a much better solution.

3.4.3 ARBITRATION OF REFRESH REQUESTS

The arbitration of refresh requests by the DRAM Refresh Controller is slightly different from normal DMA channel request arbitration. The 82380 DRAM Refresh Controller always has the highest priority of any DMA process. It also can interrupt a process in progress. Two types of processes in progress may be encountered: normal DMA, and bus master cascade.

In the event of a refresh request during a normal DMA process, the DMA Controller will complete the data transfer in progress and then execute the refresh cycle before continuing with the current DMA process. The priority of the interrupted process is not lost. If the data transfer cycle interrupted by the Refresh Controller is the last of a DMA process, the refresh cycle will always be executed before control of the bus is transferred back to the host.

When the Refresh Controller request occurs during a cascade cycle, the Refresh Controller must be assured that the cascaded master device has relinquished control of the bus before it can execute the refresh cycle. To do this, the DMA Controller drops the EDACK signal to the cascaded master and waits for the corresponding DREQn input to go inactive. By dropping the DREQn signal, the cascaded master relinquishes the bus. The Refresh Controller then performs the refresh cycle. Control of the bus is returned to the cascaded master if DREQn returns to an active state before the end of the refresh cycle, otherwise control is passed to the processor and the cascaded master loses its priority.

3.5 DMA Controller Register Overview

The 82380 DMA Controller contains 44 registers which are accessable to the host processor. Twenty-four of these registers contain the device addresses and data counts for the individual DMA channels (three per channel). The remaining registers are control and status registers for initiating and monitoring the operation of the 82380 DMA Controller. Table 3-4 lists the DMA Controller's registers and their accessability.

Register Name	Access
Control/Status Register—One Each Per Group	
Command Register I	Write Only
Command Register II	Write Only
Mode Register I	Write Only
Mode Register II	Write Only
Software Request Register	Read/Write
Mask Set-Reset Register	Write Only
Mask Read-Write Register	Read/Write
Status Register	Read Only
Bus Size Register	Write Only
Chaining Register	Read/Write
Channel Registers—One Each Per Channel	
Base Target Address	Write Only
Current Target Address	Read Only
Base Requester Address	Write Only
Current Requester Address	Read Only
Base Byte Count	Write Only
Current Byte Count	Read Only

Table 3-4. DMA Controller Registers

3.5.1 CONTROL/STATUS REGISTERS

The following registers are available to the host processor for programming the 82380 DMA Controller into its various modes and for checking the operating status of the DMA processes. Each set of four DMA channels has one of each of these registers associated with it.

Command Register I

Enables or disables the DMA channels as a group. Sets the Priority Mode (Fixed or Rotating) of the group. This write-only register is cleared by a hardware reset, defaulting to all channels enabled and Fixed Priority Mode.

Command Register II

Sets the sampling mode of the DREQn and EOP# inputs. Also sets the lowest priority channel for the group in the Fixed Priority Mode. The functions programmed through Command Register II default after a hardware reset to: asynchronous DREQn and EOP#, and channels 3 and 7 lowest priority.

Mode Register I

Mode Register I is identical in function to the Mode register of the 8237A. It programs the following functions for an individually selected channel:

Type of Transfer—read, write, verify
Auto—Initialize—enable or disable
Target Address Count—increment or decrement
Data Transfer Mode—demand, single, block, cascade

Mode Register I functions default to the following after reset: verify transfer, Auto-Initialize disabled, Increment Target address, Demand Mode.

Mode Register II

Programs the following functions for an individually selected channel:

Target Address Hold—enable or disable
Requester Address Count—increment or decrement
Requester Address Hold—enable or disable
Target Device Type—I/O or Memory
Requester Device Type—I/O or Memory
Transfer Cycles—Two-Cycle or Fly-By

Mode Register II functions are defined as follows after a hardware reset: Disable Target Address Hold, Increment Requester Address, Target (and Requester) in memory, Fly-By Transfer Cycles. Note: Requester Device Type ignored in Fly-By Transfers.

Software Request Register

The DMA Controller can respond to service requests which are initiated by software. Each channel has an internal request status bit associated with it. The host processor can write to this register to set or reset the request bit of a selected channel.

The status of the group's software DMA service requests can be read from this register as well. Each request bit is cleared upon Terminal Count or external EOP#.

The software DMA requests are non-maskable and subject to priority arbitration with all other software and hardware requests. The entire register is cleared by a hardware reset.

Mask Registers

Each channel has associated with it a mask bit which can be set/reset to disable/enable that channel. Two methods are available for setting and clearing the mask bits. The Mask Set/Reset Register is a write-only register which allows the host to select an individual channel and either set or reset the mask bit for that channel only. The Mask Read/Write Register is available for reading the mask bit status and for writing mask bits in groups of four.

The mask bits of a group may be cleared in one step by executing the Clear Mask Command. See the DMA Programming section for details. A hardware reset sets all of the channel mask bits, disabling all channels.

Status Register

The Status register is a read-only register which contains the Terminal Count (TC) and Service Request status for a group. Four bits indicate the TC status and four bits indicate the hardware request status for the four channels in the group. The TC bits are set when the Byte Count expires, or when an external EOP# is asserted. These bits are cleared by reading from the Status Register. The Service Request bit for a channel indicates when there is a hardware DMA request (DREQn) asserted for that channel. When the request has been removed, the bit is cleared.

Bus Size Register

This write-only register is used to define the bus size of the Target and Requester of a selected channel. The bus sizes programmed will be used to dictate the sizes of the data paths accessed when the DMA channel is active. The values programmed into this register affect the operation of the Temporary Register. Any byte-assembly required to make the transfers using the specified data path widths will be done in the Temporary Register. The Bus Size register of the Target is used as an increment/decrement value for the Byte Counter and Target Address when in the Fly-By Mode. Upon reset, all channels default to 8-bit Targets and 8-bit Requesters.

Chaining Register

As a command or write register, the Chaining register is used to enable or disable the Chaining Mode for a selected channel. Chaining can either be disabled or enabled for an individual channel, independently of the Chaining Mode status of other channels. After a hardware reset, all channels default to Chaining disabled.

When read by the host, the Chaining Register provides the status of the Chaining Interrupt of each of the channels. These interrupt status bits are cleared when the new buffer information has been loaded.

3.5.2 CHANNEL REGISTERS

Each channel has three individually programmable registers necessary for the DMA process; they are the Base Byte Count, Base Target Address, and Base Requester Address registers. The 24-bit Base

Byte Count register contains the number of bytes to be transferred by the channel. The 32-bit Base Target Address Register contains the beginning address (memory or I/O) of the Target device. The 32-bit Base Requester Address register contains the base address (memory or I/O) of the device which is to request DMA service.

Three more registers for each DMA channel exist within the DMA Controller which are directly related to the registers mentioned above. These registers contain the current status of the DMA process. They are the Current Byte Count register, the Current Target Address, and the Current Requester Address. It is these registers which are manipulated (incremented, decremented, or held constant) by the 82380 DMA Controller during the DMA process. The Current registers are loaded from the Base registers.

The Base registers are loaded when the host processor writes to the respective channel register addresses. Depending on the mode in which the channel is operating, the Current registers are typically loaded in the same operation. Reading from the channel register addresses yields the contents of the corresponding Current register.

To maintain compatibility with software which accesses an 8237A, a Byte Pointer Flip-Flop is used to control access to the upper and lower bytes of some words of the Channel Registers. These words are accessed as byte pairs at single port addresses. The Byte Pointer Flip-Flop acts as a one-bit pointer which is toggled each time a qualifying Channel Register byte is accessed. It always points to the next logical byte to be accessed of a pair of bytes.

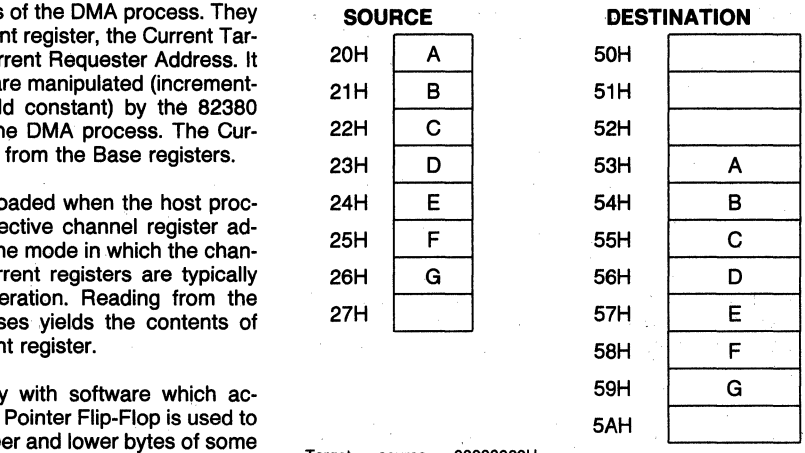
The Channel registers are arranged as pairs of words, each pair with its own port address. Addressing the port with the Byte Pointer Flip-Flop reset accesses the least significant byte of the pair. The most significant byte is accessed when the Byte Pointer is set.

For compatibility with existing 8237A designs, there is one exception to the above statements about the Byte Pointer Flip-Flop. The third byte (bits 16–23) of the Target Address is accessed through its own port address. The Byte Pointer Flip-Flop is not affected by any accesses to this byte.

The upper eight bits of the Byte Count Register are cleared when the least significant byte of the register is loaded. This provides compatibility with software which accesses an 8237A. The 8237A has 16-bit Byte Count Registers.

3.5.3 TEMPORARY REGISTERS

Each channel has a 32-bit Temporary Register used for temporary data storage during two-cycle DMA transfers. It is this register in which any necessary byte assembly and disassembly of non-aligned data is performed. Figure 3-22 shows how a block of data will be moved between memory locations with different boundaries. Note that the order of the data does not change.



Target = source = 00000020H
Requester = destination = 00000053H
Byte Count = 000006H

Figure 3-22. Transfer of Data between Memory Locations with Different Boundaries. This will be the result, independent of data path width.

If the destination is the Requester and an early process termination has been indicated by the EOP# signal or DREQn inactive in the Demand Mode, the Temporary Register is not affected. If data remains in the Temporary Register due to differences in data path widths of the Target and Requester, it will not be transferred or otherwise lost, but will be stored for later transfer.

If the destination is the Target and the EOP# signal is sensed active during the Requester access of a transfer, the DMA Controller will complete the transfer by sending to the Target whatever information is in the Temporary Register at the time of process termination. This implies that the Target could be accessed with partial data. For this reason it is advisable to have an I/O device designated as a Requester, unless it is capable of handling partial data transfers.

3.6 DMA Controller Programming

Programming a DMA Channel to perform a needed DMA function is in general a four step process. First the global attributes of the DMA Controller are programmed via the two Command Registers. These global attributes include: priority levels, channel group enables, priority mode, and DREQn/EOP# input sampling.

The second step involves setting the operating modes of the particular channel. The Mode Registers are used to define the type of transfer and the handshaking modes. The Bus Size Register and Chaining Register may also need to be programmed in this step.

The third step is setting up the channel is to load the Base Registers in accordance with the needs of the operating modes chosen in step two. The Current Registers are automatically loaded from the Base Registers, if required by the Buffer Transfer Mode in effect. The information loaded and the order in which it is loaded depends on the operating mode. A channel used for cascading, for example, needs no buffer information and this step can be skipped entirely.

The last step is to enable the newly programmed channel using one of the Mask Registers. The channel is then available to perform the desired data transfer. The status of the channel can be observed at any time through the Status Register, Mask Register, Chaining Register, and Software Request register.

Once the channel is programmed and enabled, the DMA process may be initiated in one of two ways, either by a hardware DMA request (DREQn) or a software request (Software Request Register).

Once programmed to a particular Process/Mode configuration, the channel will operate in that configuration until programmed otherwise. For this reason, restarting a channel after the current buffer expires does not require complete reprogramming of the channel. Only those parameters which have changed need to be reprogrammed. The Byte Count Register is always changed and must be reprogrammed. A Target or Requester Address Register which is incremented or decremented should be reprogrammed also.

3.6.1 BUFFER PROCESSES

The Buffer Process is determined by the Auto-Initialize bit of Mode Register I and the Chaining Register. If Auto-Initialize is enabled, Chaining should not be used.

3.6.1.1 Single Buffer Process

The Single Buffer Process is programmed by disabling Chaining via the Chaining Register and programming Mode Register I for non-Auto-Initialize.

3.6.1.2 Buffer Auto-Initialize Process

Setting the Auto-Initialize bit in Mode Register I is all that is necessary to place the channel in this mode. Buffer Auto-Initialize must not be enabled simultaneous to enabling the Buffer Chaining Mode as this will have unpredictable results.

Once the Base Registers are loaded, the channel is ready to be enabled. The channel will reload its Current Registers from the Base Registers each time the Current Buffer expires, either by an expired Byte Count or an external EOP#.

3.6.1.3 Buffer Chaining Process

The Buffer Chaining Process is entered into from the Single Buffer Process. The Mode Registers should be programmed first, with all of the Transfer Modes defined as if the channel were to operate in the Single Buffer Process. The channel's Base and Current Registers are then loaded. When the channel has been set up in this way, and the chaining interrupt service routine is in place, the Chaining Process can be entered by programming the Chaining Register. Figure 3.23 illustrates the Buffer Chaining Process.

An interrupt (IRQ1) will be generated immediately after the Chaining Process is entered, as the channel

then perceives the Base Registers as empty and in need of reloading. It is important to have the interrupt service routine in place at the time the Chaining Process is entered into. The interrupt request is removed when the most significant byte of the Base Target Address is loaded.

The interrupt will occur again when the first buffer expires and the Current Registers are loaded from the Base Registers. The cycle continues until the Chaining Process is disabled, or the host fails to respond to IRQ1 before the Current Buffer expires.

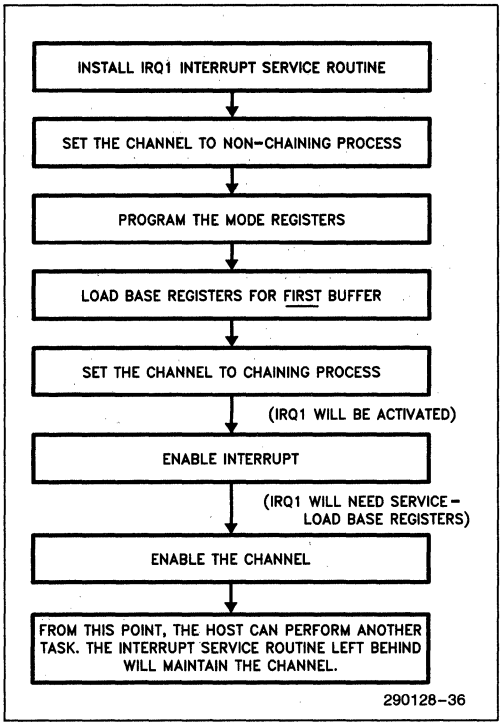


Figure 3-23. Flow of Events in the Buffer Chaining Process

Exiting the Chaining Process can be done by resetting the Chaining Mode Register. If an interrupt is pending for the channel when the Chaining Register is reset, the interrupt request will be removed. The Chaining Process can be temporarily disabled by setting the channel's Mask bit in the Mask Register.

The interrupt service routine for IRQ1 has the responsibility of reloading the Base Register as necessary. It should check the status of the channel to determine the cause of channel expiration, etc. It should also have access to operating system information regarding the channel, if any exists. The IRQ1 service routine should be capable of determining whether the chain should be continued or terminated and act on that information.

3.6.2 DATA TRANSFER MODES

The Data Transfer Modes are selected via Mode Register I. The Demand, Single, and Block Modes are selected by bits D6 and D7. The individual transfer type (Fly-By vs Two-Cycle, Read-Write-Verify, and I/O vs Memory) is programmed through both of the Mode registers.

3.6.3 CASCADED BUS MASTERS

The Cascade Mode is set by writing ones to D7 and D6 of Mode Register I. When a channel is programmed to operate in the Cascade Mode, all of the other modes associated with Mode Registers I and II are ignored. The priority and DREQn/EOP# definitions of the Command Registers will have the same effect on the channel's operation as any other mode.

3.6.4 SOFTWARE COMMANDS

There are five port addresses which, when written to, command certain operations to be performed by the 82380 DMA Controller. The data written to these locations is not of consequence, writing to the location is all that is necessary to command the 82380 to perform the indicated function. Following are descriptions of the command function.

Clear Byte Pointer Flip-Flop—location 000CH

Resets the Byte Pointer Flip-Flop. This command should be performed at the beginning of any access to the channel registers in order to be assured of beginning at a predictable place in the register programming sequence.

Master Clear—location 000DH

All DMA functions are set to their default states. This command is the equivalent of a hardware reset to the DMA Controller. Functions other than those in the DMA Controller section of the 82380 are not affected by this command.

Clear Mask Register —Channels 0–3—location 000EH
Channels 4–7—location 00CEH

This command simultaneously clears the Mask Bits of all channels in the addressed group, enabling all of the channels in the group.

Clear TC Interrupt Request—location 001EH

This command resets the Terminal Count Interrupt Request Flip-Flop. It is provided to allow the program which made a software DMA request to acknowledge that it has responded to the expiration of the requested channel(s).

3.7 Register Definitions

The following diagrams outline the bit definitions and functions of the 82380 DMA Controller's Status and Control Registers. The function and programming of the registers is covered in the previous section on DMA Controller Programming. An entry of 'X' as a bit value indicates "don't care."

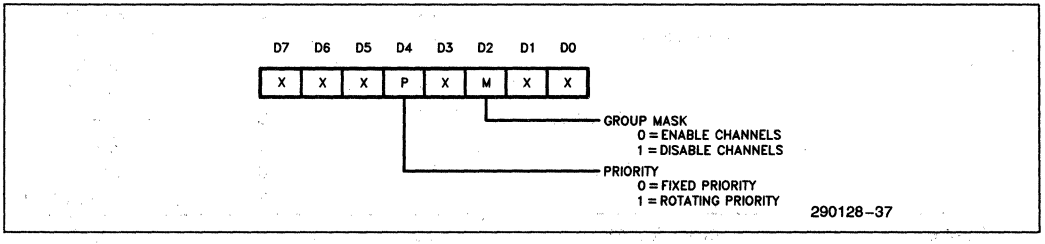
Channel Registers Channel	Register Name	(Read Current, Write Base) Address (Hex)	Byte Pointer	Bits Accessed
Channel 0	Target Address	00	0	0–7
			1	8–15
		87	x	16–23
		10	0	24–31
	Byte Count	01	0	0–7
			1	8–15
		11	0	16–23
	Requester Address	90	0	0-7
			1	8–15
		91	0	16–23
			1	24–31
Channel 1	Target Address	02	0	0–7
			1	8–15
		83	x	16–23
		12	0	24–31
	Byte Count	03	0	0–7
			1	8–15
		13	0	16–23
	Requester Address	92	0	0-7
			1	8–15
		93	0	16–23
			1	24–31

Channel Registers Channel	Register Name	(Read Current, Write Base) Address (Hex)	Byte Pointer	Bits Accessed
Channel 2	Target Address	04	0	0–7
			1	8–15
		81	x	16–23
		14	0	24–31
	Byte Count	05	0	0–7
			1	8–15
		15	0	16–23
	Requester Address	94	0	0-7
			1	8–15
		95	0	16–23
			1	24–31
Channel 3	Target Address	06	0	0–7
			1	8–15
		82	x	16–23
		16	0	24–31
	Byte Count	07	0	0–7
			1	8–15
		17	0	16–23
	Requester Address	96	0	0-7
			1	8–15
		97	0	16–23
			1	24–31
Channel 4	Target Address	C0	0	0–7
			1	8–15
		8F	x	16–23
		D0	0	24–31
	Byte Count	C1	0	0–7
			1	8–15
		D1	0	16–23
	Requester Address	98	0	0-7
			1	8–15
		99	0	16–23
			1	24–31
Channel 5	Target Address	C2	0	0–7
			1	8–15
		8B	x	16–23
		D2	0	24–31
	Byte Count	C3	0	0–7
			1	8–15
		D3	0	16–23
	Requester Address	9A	0	0-7
			1	8–15
		9B	0	16–23
			1	24–31

Channel Registers Channel	Register Name	(Read Current, Write Base) Address (Hex)	Byte Pointer	Bits Accessed
Channel 6	Target Address	C4	0	0–7
			1	8–15
		89	x	16–23
		D4	0	24–31
	Byte Count	C5	0	0–7
			1	8–15
		D5	0	16–23
	Requester Address	9C	0	0-7
			1	8–15
		9D	0	16–23
			1	24–31
Channel 7	Target Address	C6	0	0–7
			1	8–15
		8A	x	16–23
		D6	0	24–31
	Byte Count	C7	0	0–7
			1	8–15
		D7	0	16–23
	Requester Address	9E	0	0-7
			1	8–15
		9F	0	16–23
			1	24–31

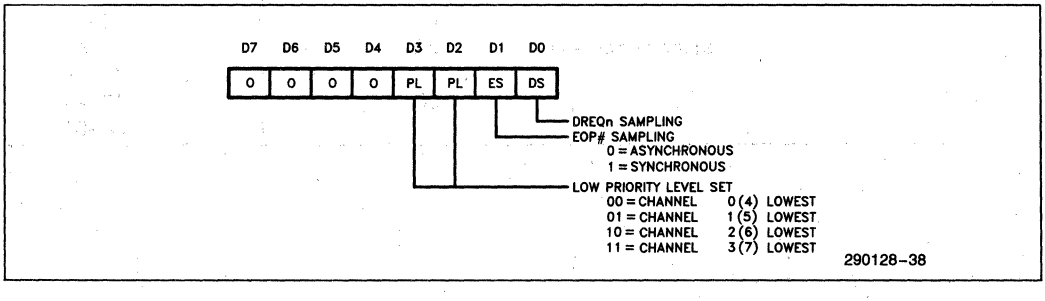
Command Register I (Write Only)

Port Address—Channels 0–3—0008H
Channels 4–7—00C8H



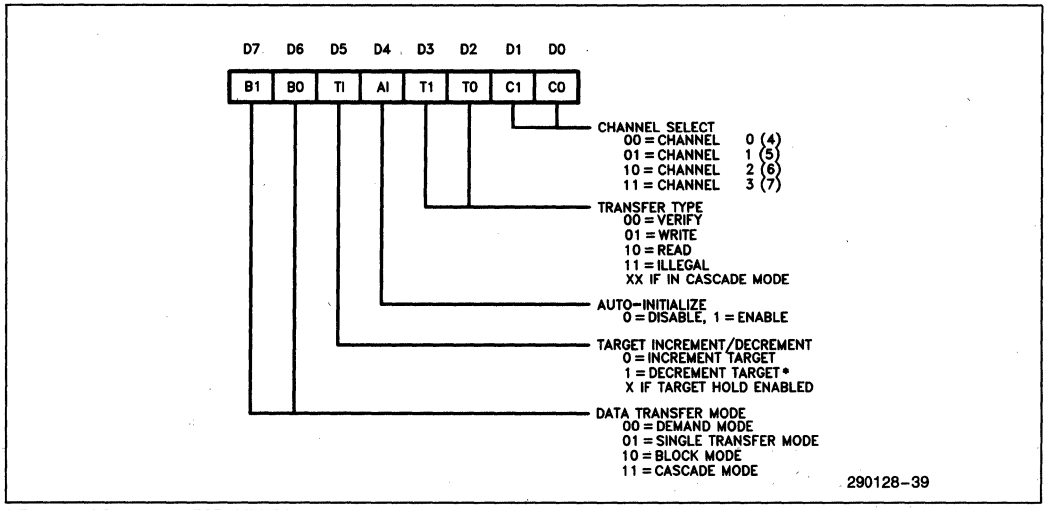
Command Register II (Write Only)

Port Addresses—Channels 0–3—001AH
Channels 4–7—00DAH



Mode Register I (Write Only)

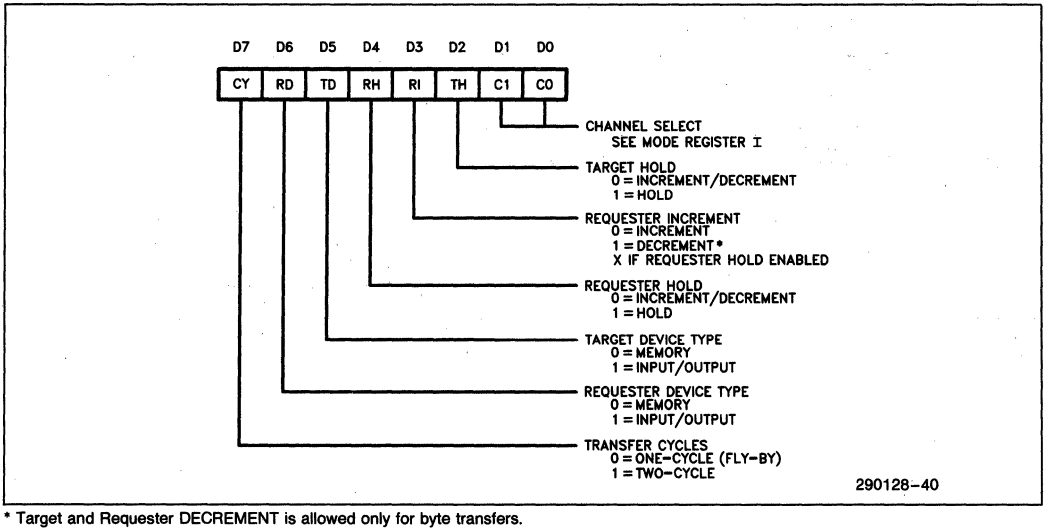
Port Addresses—Channels 0-3—000BH
Channels 4-7—00CBH



* Target and Requester DECREMENT is allowed only for byte transfers.

Mode Register II (Write Only)

Port Addresses—Channels 0-3—001BH
Channels 4-7—00DBH

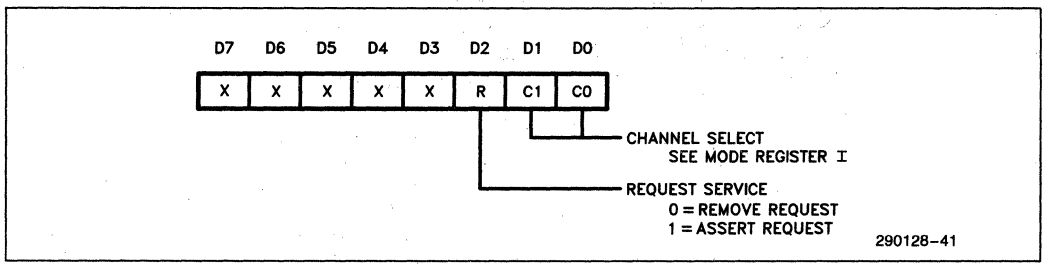


* Target and Requester DECREMENT is allowed only for byte transfers.

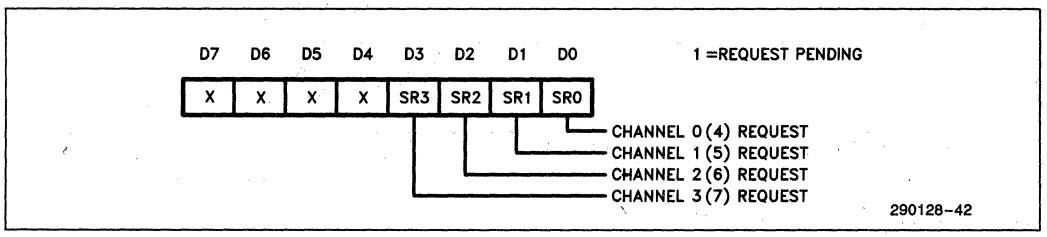
Software Request Register (Read/Write)

Port Addresses—Channels 0-3—0009H
Channels 4-7—00C9H

Write Format: Software DMA Service Request

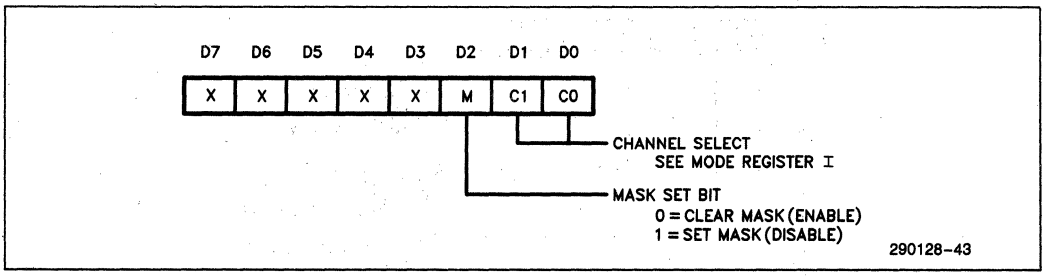


Read Format: Software Requests Pending



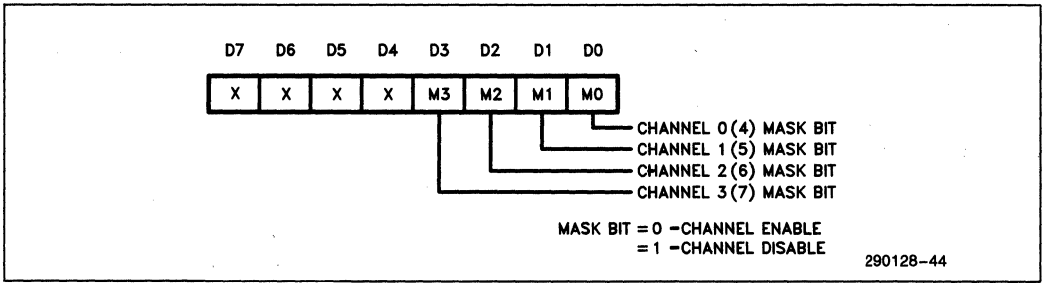
Mask Set/Reset Register Individual Channel Mask (Write Only)

Port Addresses—Channels 0-3—000AH
Channels 4-7—00CAH



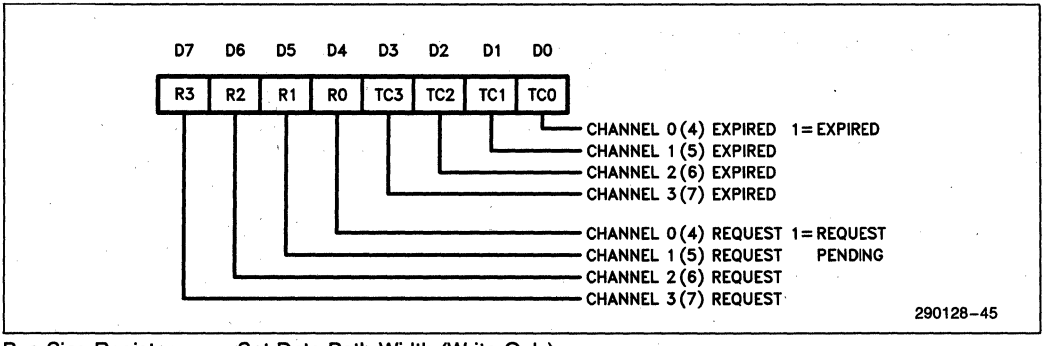
Mask Read/Write Register Group Channel Mask (Read/Write)

Port Addresses—Channels 0-3—000FH
Channels 4-7—00CFH



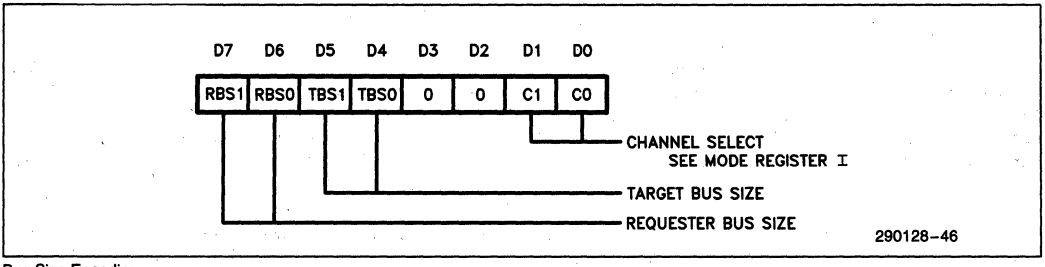
Status Register Channel Process Status (Read Only)

Port Addresses—Channels 0-3—0008H
Channels 4-7—00C8H



Bus Size Register Set Data Path Width (Write Only)

Port Addresses—Channels 0-3—0018H
Channels 4-7—00D8H

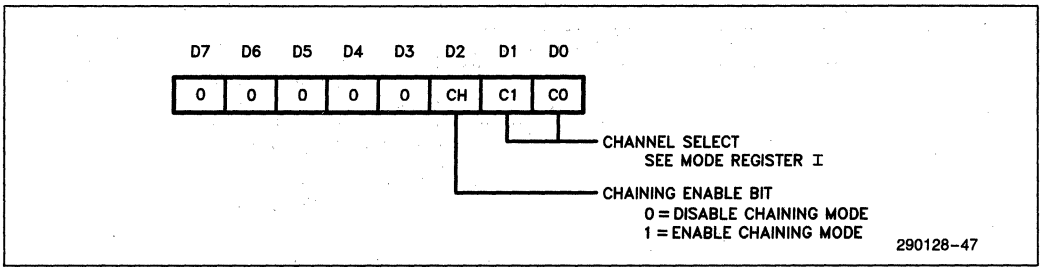


Bus Size Encoding:
00 = Reserved by Intel 10 = 16-bit Bus
01 = 32-bit Bus 11 = 8-bit Bus

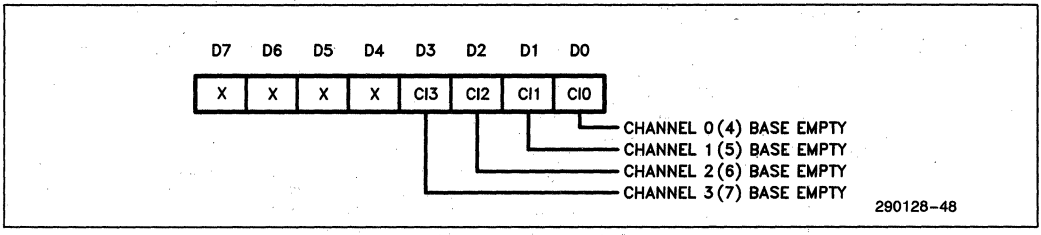
Chaining Register (Read/Write)

Port Addresses—Channels 0–3—0019H
Channels 4–7—00D9H

Write Format: Set Chaining Mode



Read Format: Channel Interrupt Status



3.8 8237A Compatibility

The register arrangement of the 82380 DMA Controller is a superset of the 8237A DMA Controller. Functionally the 82380 DMA Controller is very different from the 8237A. Most of the functions of the 8237A are performed also by the 82380. The following discussion points out the differences between the 8237A and the 82380.

The 8237A is limited to transfers between I/O and memory only (except in one special case, where two channels can be used to perform memory-to-memory transfers). The 82380 DMA Controller can transfer between any combination of memory and I/O. Several other features of the 8237A are enhanced or expanded in the 82380 and other features are added.

The 8237A is an 8-bit only DMA device. For programming compatibility, all of the 8-bit registers are preserved in the 82380. The 82380 is programmed via 8-bit registers. The address registers in the 82380 are 32-bit registers in order to support the 80386's 32-bit bus. The Byte Count Registers are 24-bit registers, allowing support of larger data blocks than possible with the 8237A.

All of the 8237A's operating modes are supported by the 82380 (except the cumbersome two-channel memory-to-memory transfer). The 82380 performs memory-to-memory transfers using only one channel. The 82380 has the added features of buffer pipelining (Buffer Chaining Process), programmable priority levels, and Byte Assembly.

The 82380 also adds the feature of address registers for both destination and source. These addresses may be incremented, decremented, or held constant, as required by the application of the individual channel. This allows any combination of destination and source device.

Each DMA channel has associated with it a Target and a Requester. In the 8237A, the Target is the device which can be accessed by the address register, the Requester is the device which is accessed by the DMA Acknowledge signals and must be an I/O device.

4.0 Programmable Interrupt Controller (PIC)

4.1 Functional Description

The 82380 Programmable Interrupt Controller (PIC) consists of three enhanced 82C59A Interrupt Controllers. These three controllers together provide 15 external and 5 internal interrupt request inputs. Each external request input can be cascaded with an additional 82C59A slave collector. This scheme allows the 82380 to support a maximum of 120 (15 x 8) external interrupt request inputs.

Following one or more interrupt requests, the 82380 PIC issues an interrupt signal to the 80386. When the 80386 host processor responds with an interrupt acknowledge signal, the PIC will arbitrate between the pending interrupt requests and place the interrupt vector associated with the highest priority pending request on the data bus.

The major enhancement in the 82380 PIC over the 82C59A is that each of the interrupt request inputs can be individually programmed with its own interrupt vector, allowing more flexibility in interrupt vector mapping.

4.1.1 INTERNAL BLOCK DIAGRAM

The block diagram of the 82380 Programmable Interrupt Controller is shown in Figure 4-1. Internally, the PIC consists of three 82C59A banks: A, B and C. The three banks are cascaded to one another: C is cascaded to B, B is cascaded to A. The INT output of Bank A is used externally to interrupt the 80386.

Bank A has nine interrupt request inputs (two are unused), and Banks B and C have eight interrupt request inputs. Of the fifteen external interrupt request inputs, two are shared by other functions. Specifically, the Interrupt Request 3 input (IRQ3#) can be used as the Timer 2 output (TOUT2#). This pin can be used in three different ways: IRQ3# input only, TOUT2# output only, or using TOUT2# to generate an IRQ3# interrupt request. Also, the Interrupt Request 9 input (IRQ 9#) can be used as DMA Request 4 input (DREQ4). Typically, only IRQ9# or DREQ4 can be used at a time.

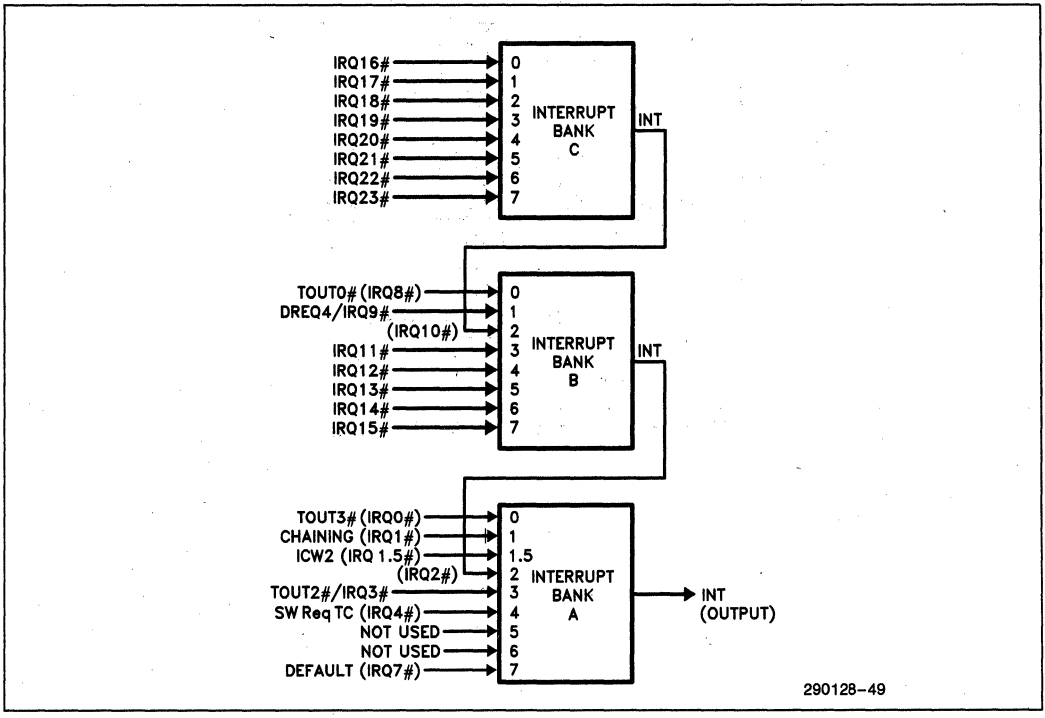


Figure 4-1. Interrupt Controller Block Diagram

4.1.2 INTERRUPT CONTROLLER BANKS

All three banks are identical, with the exception of the IRQ1.5 on Bank A. Therefore, only one bank will be discussed. In the 82380 PIC, all external requests can be cascaded into and each interrupt controller bank behaves like a master. As compared to the 82C59A, the enhancements in the banks are:

— All interrupt vectors are individually programmable. (In the 82C59A, the vectors must be programmed in eight consecutive interrupt vector locations.)

— The cascade address is provided on the Data Bus (D0–D7). (In the 82C59A, three dedicated control signals (CAS0, CAS1, CAS2) are used for master/slave cascading.)

The block diagram of a bank is shown in Figure 4-2. As can be seen from this figure, the bank consists of six major blocks: the Interrupt Request Register (IRR), the In-Service Register (ISR), the Interrupt Mask Register (IMR), the Priority Resolver (PR), the Vector Register (VR), and the Control Logic. The functional description of each block follows.

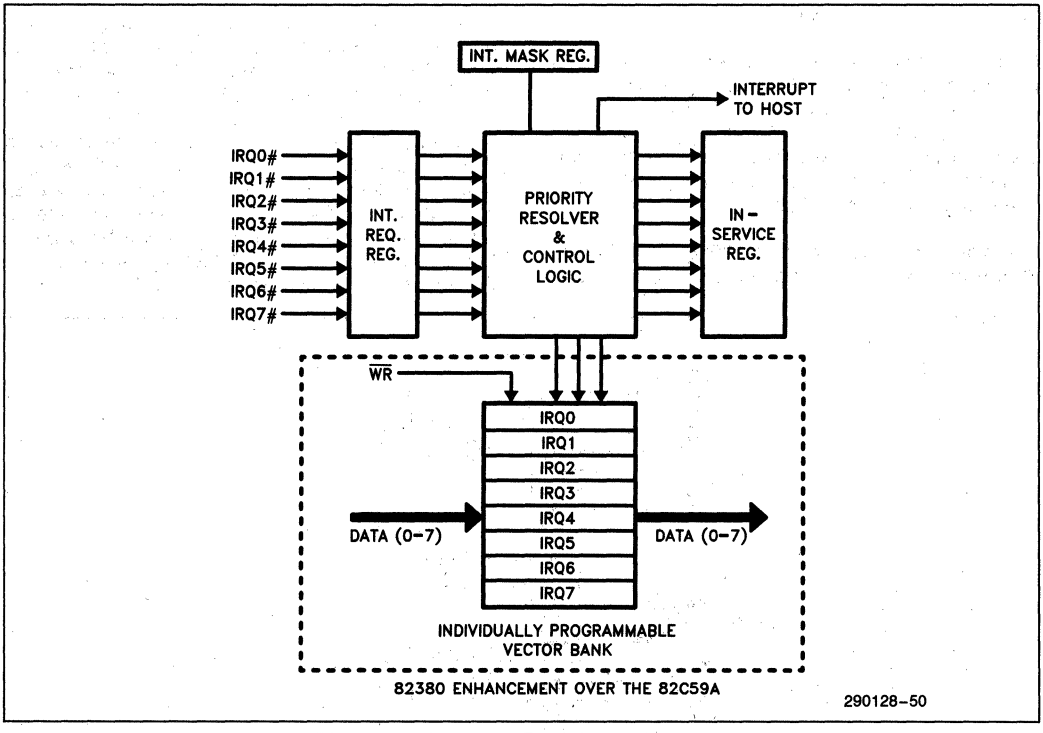


Figure 4-2. Interrupt Bank Block Diagram

INTERRUPT REQUEST (IRR) AND IN-SERVICE REGISTER (ISR)

The interrupts at the Interrupt Request (IRQ) input lines are handled by two registers in cascade, the Interrupt Request Register (IRR) and the In-Service Register (ISR). The IRR is used to store all interrupt levels which are requesting service; and the ISR is used to store all interrupt levels which are being serviced.

PRIORITY RESOLVER (PR)

This logic block determines the priorities of the bits set in the IRR. The highest priority is selected and strobed into the corresponding bit of the ISR during an Interrupt Acknowledge cycle.

INTERRUPT MASK REGISTER (IMR)

The IMR stores the bits which mask the interrupt lines to be masked (disabled). The IMR operates on the IRR. Masking of a higher priority input will not affect the interrupt request lines of lower priority.

VECTOR REGISTERS (VR)

This block contains a set of Vector Registers, one for each interrupt request line, to store the pre-programmed interrupt vector number. The corresponding vector number will be driven onto the Data Bus of the 82380 during the Interrupt Acknowledge cycle.

CONTROL LOGIC

The Control Logic coordinates the overall operations of the other internal blocks within the same bank. This logic will drive the Interrupt Output signal (INT) HIGH when one or more unmasked interrupt inputs are active (LOW). The INT output signal goes directly to the 80386 (in Bank A) or to another bank to which this bank is cascaded (see Figure 4-1). Also, this logic will recognize an Interrupt Acknowledge cycle (via M/IO#, D/C# and W/R# signals). During this bus cycle, the Control Logic will enable the corresponding Vector Register to drive the interrupt vector onto the Data Bus.

In Bank A, the Control Logic is also responsible for handling the special ICW2 interrupt request input (IRQ1.5#).

4.2 Interface Signals

4.2.1 INTERRUPT INPUTS

There are 15 external Interrupt Request inputs and 5 internal Interrupt Requests. The external request inputs are: IRQ3#, IRQ9#, IRQ11# to IRQ23#. They are shown in bold arrows in Figure 4-1. All IRQ inputs are active LOW and they can be programmed (via a control bit in the Initialization Command Word 1 (ICW1)) to be either edge-triggered or level-triggered. In order to be recognized as a valid interrupt request, the interrupt input must be active (LOW) until the first INTA# cycle (see Bus Functional Description). Note that all 15 external Interrupt Request inputs have weak internal pull-up resistors.

As mentioned earlier, an 82C59A can be cascaded to each external interrupt input to expand the interrupt capacity to a maximum of 120 levels. Also, two of the interrupt inputs are dual functions: IRQ3# can be used as Timer 2 output (TOUT2#) and IRQ9# can be used as DREQ4 input. IRQ3# is a bidirectional dual function pin. This interrupt request input is wired-OR with the output of Timer 2 (TOUT2#). If only IRQ3# function is to be used, Timer 2 should be programmed so that OUT2 is LOW. Note that TOUT2# can also be used to generate an interrupt request to IRQ3# input.

The five internal interrupt requests serve special system functions. They are shown in Table 4-1. The following paragraphs describe these interrupts.

Table 4-1. 82380 Internal Interrupt Requests

Interrupt Request	Interrupt Source
IRQ0#	Timer 3 Output (TOUT3#)
IRQ8#	Timer 0 Output (TOUT0#)
IRQ1#	DMA Chaining Request
IRQ4#	DMA Terminal Count
IRQ1.5#	ICW2 Written

TIMER 0 AND TIMER 3 INTERRUPT REQUESTS [IRQ0#]

IRQ8# and IRQ0# interrupt requests are initiated by the output of Timers 0 and 3, respectively. Each of these requests is generated by an edge-detector flip-flop. The flip-flops are activated by the following conditions:

Set— Rising edge of timer output (TOUT);

Clear— Interrupt acknowledge for this request; OR Request is masked (disabled); OR Hardware Reset.

CHAINING AND TERMINAL COUNT INTERRUPTS [IRQ1#]

These interrupt requests are generated by the 82380 DMA Controller. The chaining request (IRQ1#) indicates that the DMA Base Register is not loaded. The Terminal Count request (IRQ4#) indicates that a software DMA request was cleared.

ICW2 INTERRUPT REQUEST [IRQ1.5#]

Whenever an Initialization Control Word 2 (ICW2) is written to a Bank, a special ICW2 interrupt request is generated. The interrupt will be cleared when the newly programmed ICW2 Register is read. This interrupt request is in Bank A at level 1.5. This interrupt request is internally ORed with the Cascaded Request from Bank B and is always assigned a higher priority than the Cascaded Request.

This special interrupt is provided to support compatibility with the original 82C59A. A detailed description of this interrupt is discussed in the Programming section.

DEFAULT INTERRUPT [IRQ7#]

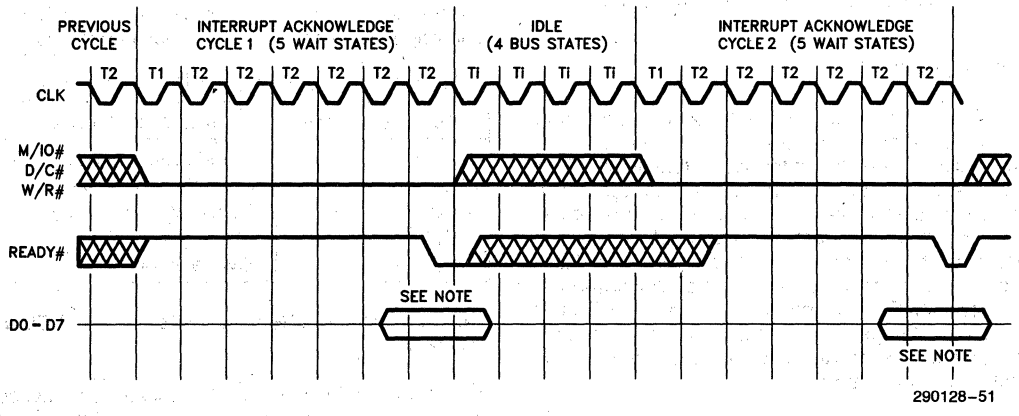
During an Interrupt Acknowledge cycle, if there is no active pending request, the PIC will automatically generate a default vector. This vector corresponds to the IRQ7# vector in Bank A.

4.2.2 INTERRUPT OUTPUT (INT)

The INT output pin is taken directly from bank A. This signal should be tied to the Maskable Interrupt Request (INTR) of the 80386. When this signal is active (HIGH), it indicates that one or more internal/external interrupt requests are pending. The 80386 is expected to respond with an interrupt acknowledge cycle.

4.3 Bus Functional Description

The INT output of bank A will be activated as a result of any unmasked interrupt request. This may be a non-cascaded or cascaded request. After the PIC has driven the INT signal HIGH, 80386 will respond by performing two interrupt acknowledge cycles. The timing diagram in Figure 4-3 shows a typical interrupt acknowledge process between the 82380 and the 80386 CPU.



NOTE:
What is actually driven on the Data Bus depends on if the current interrupt request is a Slave Request.

	INTA Cycle 1	INTA Cycle 2
NON-SLAVE REQUEST	00H	Vector
SLAVE REQUEST	Slave Address	High Impedance*

*Slave will place a vector at this time.

Figure 4-3. Interrupt Acknowledge Cycle

After activating the INT signal, the 82380 monitors the status lines (M/IO#, D/C#, W/R#) and waits for the 80386 to initiate the first interrupt acknowledge cycle. In the 80386 environment, two successive interrupt acknowledge cycles (INTA) marked by M/IO# = LOW, D/C# = LOW, and W/R# = LOW are performed. During the first INTA cycle, the PIC will determine the highest priority request. Assuming this interrupt input has no external Slave Controller cascaded to it, the 82380 will drive the Data Bus with 00H in the first INTA cycle. During the second INTA cycle, the 82380 PIC will drive the Data Bus with the corresponding preprogrammed interrupt vector.

If the PIC determines (from the ICW3) that this interrupt input has an external Slave Controller cascaded to it, it will drive the Data Bus with the specific Slave Cascade Address (instead of 00H) during the first INTA cycle. This Slave Cascade Address is the preprogrammed content in the corresponding Vector Register. This means that no Slave Address should be chosen to be 00H. Note that the Slave Address and Interrupt Vector are different interpretations of the same thing. They are both the contents of the programmable Vector Register. During the second INTA cycle, the Data Bus will be floated so that the external Slave Controller can drive its interrupt vector on the bus. Since the Slave Interrupt Controller resides on the system bus, bus transceiver enable and direction control logic must take this into consideration.

In order to have a successful interrupt service, the interrupt request input must be held active (LOW) until the beginning of the first interrupt acknowledge cycle. If there is no pending interrupt request when the first INTA cycle is generated, the PIC will generate a default vector, which is the IRQ7 vector (bank A level 7).

According to the Bus Cycle definition of the 80386, there will be four Bus Idle States between the two interrupt acknowledge cycles. These idle bus cycles will be initiated by the 80386. Also, during each interrupt acknowledge cycle, the internal Wait State Generator of the 82380 will automatically generate the required number of wait states for internal delays.

4.4 Mode of Operation

A variety of modes and commands are available for controlling the 82380 PIC. All of them are programmable; that is, they may be changed dynamically under software control. In fact, each bank can be programmed individually to operate in different modes. With these modes and commands, many possible configurations are conceivable, giving the user enough versatility for almost any interrupt controlled application.

This section is not intended to show how the 82380 PIC can be programmed. Rather, it describes the operation in different modes.

4.4.1 END-OF-INTERRUPT

Upon completion of an interrupt service routine, the interrupted bank needs to be notified so its ISR can be updated. This allows the PIC to keep track of which interrupt levels are in the process of being serviced and their relative priorities. Three different End-Of-Interrupt (EOI) formats are available. They are: Non-Specific EOI Command, Specific EOI Command, and Automatic EOI Mode. Selection of which EOI to use is dependent upon the interrupt operations the user wishes to perform.

If the 82380 is NOT programmed in the Automatic EOI Mode, an EOI command must be issued by the 80386 to the specific 82380 PIC Controller Bank. Also, if this controller bank is cascaded to another internal bank, an EOI command must also be sent to the bank to which this bank is cascaded. For example, if an interrupt request of Bank C in the 82380 PIC is serviced, an EOI should be written into Bank C, Bank B and Bank A. If the request comes from an external interrupt controller cascaded to Bank C, then an EOI should be written into the external controller as well.

NON-SPECIFIC EOI COMMAND

A Non-Specific EOI command sent from the 80386 lets the 82380 PIC bank know when a service routine has been completed, without specification of its exact interrupt level. The respective interrupt bank automatically determines the interrupt level and resets the correct bit in the ISR.

To take advantage of the Non-Specific EOI, the interrupt bank must be in a mode of operation in which it can predetermine its in-service routine levels. For this reason, the Non-Specific EOI command should only be used when the most recent level acknowledged and serviced is always the highest priority level (i.e., in the Fully Nested Mode structure to be described below). When the interrupt bank receives a Non-Specific EOI command, it simply resets the highest priority ISR bit to indicate that the highest priority routine in service is finished.

Special consideration should be taken when deciding to use the Non-Specific EOI command. Here are two operating conditions in which it is best NOT

used since the Fully Nested Mode structure will be destroyed:

— Using the Set Priority command within an interrupt service routine.

— Using a Special Mask Mode.

These conditions are covered in more detail in their own sections, but are listed here for reference.

SPECIFIC EOI COMMAND

Unlike a Non-Specific EOI command which automatically resets the highest priority ISR bit, a Specific EOI command specifies an exact ISR bit to be reset. Any one of the IRQ levels of an interrupt bank can be specified in the command.

The Specific EOI command is needed to reset the ISR bit of a completed service routine whenever the interrupt bank is not able to automatically determine it. The Specific EOI command can be used in all conditions of operation, including those that prohibit Non-Specific EOI command usage mentioned above.

AUTOMATIC EOI MODE

When programmed in the Automatic EOI Mode, the 80386 no longer needs to issue a command to notify the interrupt bank it has completed an interrupt routine. The interrupt bank accomplishes this by performing a Non-Specific EOI automatically at the end of the second INTA cycle.

Special consideration should be taken when deciding to use the Automatic EOI Mode because it may disturb the Fully Nested Mode structure. In the Automatic EOI Mode, the ISR bit of a routine in service is reset right after it is acknowledged, thus leaving no designation in the ISR that a service routine is being executed. If any interrupt request within the same bank occurs during this time and interrupts are enabled, it will get serviced regardless of its priority. Therefore, when using this mode, the 80386 should keep its interrupt request input disabled during execution of a service routine. By doing this, higher priority interrupt levels will be serviced only after the completion of a routine in service. This guideline restores the Fully Nested Mode structure. However, in this scheme, a routine in service cannot be interrupted since the host's interrupt request input is disabled.

4.4.2 INTERRUPT PRIORITIES

The 82380 PIC provides various methods for arranging the interrupt priorities of the interrupt request inputs to suit different applications. The following subsections explain these methods in detail.

4.4.2.1 Fully Nested Mode

The Fully Nested Mode of operation is a general purpose priority mode. This mode supports a multi-level interrupt structure in which all of the Interrupt Request (IRQ) inputs within one bank are arranged from highest to lowest.

Unless otherwise programmed, the Fully Nested Mode is entered by default upon initialization. At this time, IRQ0# is assigned the highest priority (priority = 0) and IRQ7# the lowest (priority = 7). This default priority can be changed, as will be explained later in the Rotating Priority Mode.

When an interrupt is acknowledged, the highest priority request is determined from the Interrupt Request Register (IRR) and its vector is placed on the bus. In addition, the corresponding bit in the In-Service Register (ISR) is set to designate the routine in service. This ISR bit will remain set until the 80386 issues an End Of Interrupt (EOI) command immediately before returning from the service routine; or alternately, if the Automatic End Of Interrupt (AEOI) bit is set, the ISR bit will be reset at the end of the second INTA cycle.

While the ISR bit is set, all further interrupts of the same or lower priority are inhibited. Higher level interrupts can still generate an interrupt, which will be acknowledged only if the 80386 internal interrupt enable flip-flop has been re-enabled (through software inside the current service routine).

4.4.2.2 Automatic Rotation—Equal Priority Devices

Automatic rotation of priorities serves in applications where the interrupting devices are of equal priority within an interrupt bank. In this kind of environment, once a device is serviced, all other equal priority peripherals should be given a chance to be serviced before the original device is serviced again. This is accomplished by automatically assigning a device the lowest priority after being serviced. Thus, in the worst case, the device would have to wait until all other peripherals connected to the same bank are serviced before it is serviced again.

There are two methods of accomplishing automatic rotation. One is used in conjunction with the Non-Specific EOI command and the other is used with the Automatic EOI mode. These two methods are discussed below.

ROTATE ON NON-SPECIFIC EOI COMMAND

When the Rotate On Non-Specific EOI command is issued, the highest ISR bit is reset as in a normal Non-Specific EOI command. However, after it is reset, the corresponding Interrupt Request (IRQ) level is assigned the lowest priority. Other IRQ priorities rotate to conform to the Fully Nested Mode based on the newly assigned low priority.

Figure 4-4 shows how the Rotate On Non-Specific EOI command affects the interrupt priorities. Assume the IRQ priorities were assigned with IRQ0 the highest and IRQ7 the lowest. IRQ6 and IRQ4 are already in service but neither is completed. Being the higher priority routine, IRQ4 is necessarily the routine being executed. During the IRQ4 routine, a rotate on Non-Specific EOI command is executed. When this happens, Bit 4 in the ISR is reset. IRQ4 then becomes the lowest priority and IRQ5 becomes the highest.

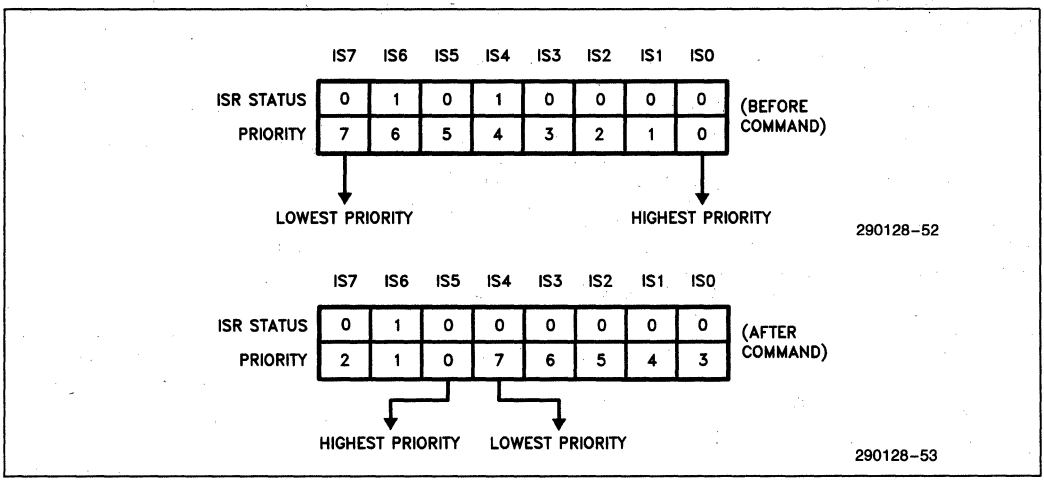


Figure 4-4. Rotate On Non-Specific EOI Command

ROTATE ON AUTOMATIC EOI MODE

The Rotate On Automatic EOI Mode works much like the Rotate On Non-Specific EOI Command. The main difference is that priority rotation is done automatically after the second INTA cycle of an interrupt request. To enter or exit this mode, a Rotate-On-Automatic-EOI Set Command and Rotate-On-Automatic-EOI Clear Command is provided. After this mode is entered, no other commands are needed as in the normal Automatic EOI Mode. However, it must be noted again that when using any form of the Automatic EOI Mode, special consideration should be taken. The guideline presented in the Automatic EOI Mode also applies here.

4.4.2.3 Specific Rotation—Specific Priority

Specific rotation gives the user versatile capabilities in interrupt controlled operations. It serves in those applications in which a specific device's interrupt priority must be altered. As opposed to Automatic Rotation which will automatically set priorities after each interrupt request is serviced, specific rotation is completely user controlled. That is, the user selects which interrupt level is to receive the lowest or the highest priority. This can be done during the main program or within interrupt routines. Two specific rotation commands are available to the user: Set Priority Command and Rotate On Specific EOI Command.

SET PRIORITY COMMAND

The Set Priority Command allows the programmer to assign an IRQ level the lowest priority. All other interrupt levels will conform to the Fully Nested Mode based on the newly assigned low priority.

ROTATE ON SPECIFIC EOI COMMAND

The Rotate On Specific EOI Command is literally a combination of the Set Priority Command and the Specific EOI Command. Like the Set Priority Command, a specified IRQ level is assigned lowest priority. Like the Specific EOI Command, a specified level will be reset in the ISR. Thus, this command accomplishes both tasks in one single command.

4.4.2.4 Interrupt Priority Mode Summary

In order to simplify understanding the many modes of interrupt priority, Table 4-2 is provided to bring out their summary of operations.

Table 4-2. Interrupt Priority Mode Summary

Interrupt Priority Mode	Operation Summary	Effect On Priority After EOI	
		Non-Specific/Automatic	Specific
Fully-Nested Mode	IRQ0#-Highest Priority IRQ7#-Lowest Priority	No change in priority. Highest ISR bit is reset.	Not Applicable.
Automatic Rotation (Equal Priority Devices)	Interrupt level just serviced is the lowest priority. Other priorities rotate to conform to Fully-Nested Mode.	Highest ISR bit is reset and the corresponding level becomes the lowest priority.	Not Applicable.
Specific Rotation (Specific Priority Devices)	User specifies the lowest priority level. Other priorities rotate to conform to Fully-Nested Mode.	Not Applicable.	As described under 'Operation Summary'.

4.4.3 INTERRUPT MASKING

VIA INTERRUPT MASK REGISTER

Each bank in the 82380 PIC has an Interrupt Mask Register (IMR) which enhances interrupt control capabilities. This IMR allows individual IRQ masking. When an IRQ is masked, its interrupt request is disabled until it is unmasked. Each bit in the 8-bit IMR disables one interrupt channel if it is set (HIGH). Bit 0 masks IRQ0, Bit 1 masks IRQ1 and so forth. Masking an IRQ channel will only disable the corresponding channel and does not affect the others operations.

The IMR acts only on the output of the IRR. That is, if an interrupt occurs while its IMR bit is set, this request is not 'forgotten'. Even with an IRQ input masked, it is still possible to set the IRR. Therefore, when the IMR bit is reset, an interrupt request to the 80386 will then be generated, providing that the IRQ request remains active. If the IRQ request is removed before the IMR is reset, the Default Interrupt Vector (Bank A, level 7) will be generated during the interrupt acknowledge cycle.

SPECIAL MASK MODE

In the Fully Nested Mode, all IRQ levels of lower priority than the routine in service are inhibited. However, in some applications, it may be desirable to let a lower priority interrupt request to interrupt the routine in service. One method to achieve this is by using the Special Mask Mode. Working in conjunction with the IMR, the Special Mask Mode enables interrupts from all levels except the level in service. This is usually done inside an interrupt service routine by masking the level that is in service and then issuing the Special Mask Mode Command. Once the Special Mask Mode is enabled, it remains in effect until it is disabled.

4.4.4 EDGE OR LEVEL INTERRUPT TRIGGERING

Each bank in the 82380 PIC can be programmed independently for either edge or level sensing for the interrupt request signals. Recall that all IRQ inputs are active LOW. Therefore, in the edge triggered mode, an active edge is defined as an input transition from an inactive (HIGH) to active (LOW) state. The interrupt input may remain active without generating another interrupt. During level triggered mode, an interrupt request will be recognized by an active (LOW) input, and there is no need for edge detection. However, the interrupt request must be removed before the EOI Command is issued, or the 80386 must be disabled to prevent a second false interrupt from occurring.

In either modes, the interrupt request input must be active (LOW) during the first INTA cycle in order to be recognized. Otherwise, the Default Interrupt Vector will be generated at level 7 of Bank A.

4.4.5 INTERRUPT CASCADING

As mentioned previously, the 82380 allows for external Slave interrupt controllers to be cascaded to any of its external interrupt request pins. The 82380 PIC indicates that a external Slave Controller is to be serviced by putting the contents of the Vector Register associated with the particular request on the 80386 Data Bus during the first INTA cycle (instead of 00H during a non-slave service). The external logic should latch the vector on the Data Bus using the INTA status signals and use it to select the external Slave Controller to be serviced (see Figure 4-5). The selected Slave will then respond to the second INTA cycle and place its vector on the Data Bus. This method requires that if external Slave Controllers

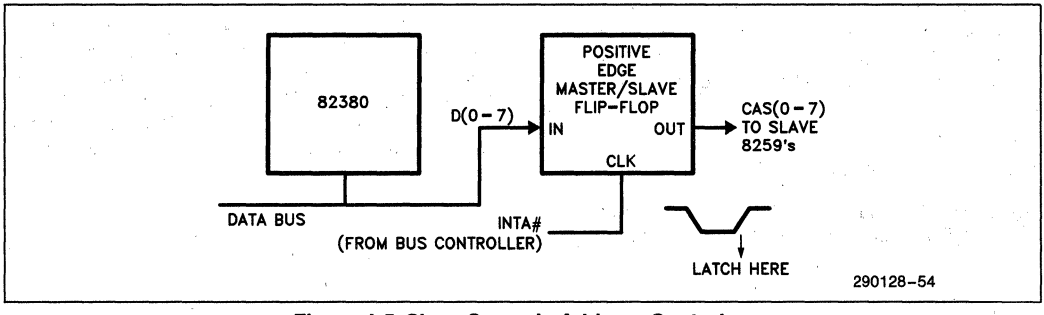


Figure 4-5. Slave Cascade Address Capturing

are used in the system, no vector should be programmed to 00H.

Since the external Slave Cascade Address is provided on the Data Bus during INTA cycle 1, an external latch is required to capture this address for the Slave Controller. A simple scheme is depicted in Figure 4-5.

4.4.5.1 Special Fully Nested Mode

This mode will be used where cascading is employed and the priority is to be conserved within each Slave Controller. The Special Fully Nested Mode is similar to the 'regular' Fully Nested Mode with the following exceptions:

— When an interrupt request from a Slave Controller is in service, this Slave Controller is not locked out from the Master's priority logic. Further interrupt requests from the higher priority logic within the Slave Controller will be recognized by the 82380 PIC and will initiate interrupts to the 80386. In comparing to the 'regular' Fully Nested Mode, the Slave Controller is masked out when its request is in service and no higher requests from the same Slave Controller can be serviced.

— Before exiting the interrupt service routine, the software has to check whether the interrupt serviced was the only request from the Slave Controller. This is done by sending a Non-Specific EOI Command to the Slave Controller and then reading its In Service Register. If there are no requests in the Slave Controller, a Non-Specific EOI can be sent to the corresponding 82380 PIC bank also. Otherwise, no EOI should be sent.

4.4.6 READING INTERRUPT STATUS

The 82380 PIC provides several ways to read different status of each interrupt bank for more flexible interrupt control operations. These include polling the highest priority pending interrupt request and reading the contents of different interrupt status registers.

4.4.6.1 Poll Command

The 82380 PIC supports status polling operations with the Poll Command. In a Poll Command, the pending interrupt request with the highest priority can be determined. To use this command, the INT output is not used, or the 80386 interrupt is disabled. Service to devices is achieved by software using the Poll Command.

This mode is useful if there is a routine command common to several levels so that the INTA sequence is not needed. Another application is to use the Poll Command to expand the number of priority levels.

Notice that the ICW2 mechanism is not supported for the Poll Command. However, if the Poll Command is used, the programmable Vector Registers are of no concern since no INTA cycle will be generated.

4.4.6.2 Reading Interrupt Registers

The contents of each interrupt register (IRR, ISR, and IMR) can be read to update the user's program on the present status of the 82380 PIC. This can be a versatile tool in the decision making process of a service routine, giving the user more control over interrupt operations.

The reading of the IRR and ISR contents can be performed via the Operation Control Word 3 by using a Read Status Register Command and the content of IMR can be read via a simple read operation of the register itself.

4.5 Register Set Overview

Each bank of the 82380 PIC consists of a set of 8-bit registers to control its operations. The address map of all the registers is shown in Table 4-3. Since all three register sets are identical in functions, only one set will be described.

Functionally, each register set can be divided into five groups. They are: the four Initialization Command Words (ICW's), the three Operation Control Words (OCW's), the Poll/Interrupt Request/In-Service Register, the Interrupt Mask Register, and the Vector Registers. A description of each group follows.

Table 4-3. Interrupt Controller Register Address Map

Port Address	Access	Register Description
20H	Write	Bank B ICW1, OCW2, or OCW3
	Read	Bank B Poll, Request or In-Service Status Register
21H	Write	Bank B ICW2, ICW3, ICW4, OCW1
	Read	Bank B Mask Register
22H	Read	Bank B ICW2
28H	Read/Write	IRQ8 Vector Register
29H	Read/Write	IRQ9 Vector Register
2AH	Read/Write	Reserved
2BH	Read/Write	IRQ11 Vector Register
2CH	Read/Write	IRQ12 Vector Register
2DH	Read/Write	IRQ13 Vector Register
2EH	Read/Write	IRQ14 Vector Register
2FH	Read/Write	IRQ15 Vector Register
A0H	Write	Bank C ICW1, OCW2, or OCW3
	Read	Bank C Poll, Request or In-Service Status Register
A1H	Write	Bank C ICW2, ICW3, ICW4, OCW1
	Read	Bank C Mask Register
A2H	Read	Bank C ICW2
A8H	Read/Write	IRQ16 Vector Register
A9H	Read/Write	IRQ17 Vector Register
AAH	Read/Write	IRQ18 Vector Register
ABH	Read/Write	IRQ19 Vector Register
ACH	Read/Write	IRQ20 Vector Register
ADH	Read/Write	IRQ21 Vector Register
AEH	Read/Write	IRQ22 Vector Register
AFH	Read/Write	IRQ23 Vector Register
30H	Write	Bank A ICW1, OCW2, or OCW3
	Read	Bank A Poll, Request or In-Service Status Register
31H	Write	Bank A ICW2, ICW3, ICW4, OCW1
	Read	Bank A Mask Register
32H	Read	Bank ICW2
38H	Read/Write	IRQ0 Vector Register
39H	Read/Write	IRQ1 Vector Register
3AH	Read/Write	IRQ1.5 Vector Register
3BH	Read/Write	IRQ3 Vector Register
3CH	Read/Write	IRQ4 Vector Register
3DH	Read/Write	Reserved
3EH	Read/Write	Reserved
3FH	Read/Write	IRQ7 Vector Register

4.5.1 INITIALIZATION COMMAND WORDS (ICW)

Before normal operation can begin, the 82380 PIC must be brought to a known state. There are four 8-bit Initialization Command Words in each interrupt bank to setup the necessary conditions and modes for proper operation. Except for the second common word (ICW2) which is a read/write register, the other three are write-only registers. Without going into detail of the bit definitions of the command words, the following subsections give a brief description of what functions each command word controls.

ICW1

The ICW1 has three major functions. They are:

— To select between the two IRQ input triggering modes (edge-or level-triggered);

— To designate whether or not the interrupt bank is to be used alone or in the cascade mode. If the cascade mode is desired, the interrupt bank will accept ICW3 for further cascade mode programming. Otherwise, no ICW3 will be accepted;

— To determine whether or not ICW4 will be issued; that is, if any of the ICW4 operations are to be used.

ICW2

ICW2 is provided for compatibility with the 82C59A only. Its contents do not affect the operation of the interrupt bank in any way. Whenever the ICW2 of any of the three banks is written into, an interrupt is generated from Bank A at level 1.5. The interrupt request will be cleared after the ICW2 register has been read by the 80386. The user is expected to program the corresponding vector register or to use it as an indicator that an attempt was made to alter the contents. Note that each ICW2 register has different addresses for read and write operations.

ICW3

The interrupt bank will only accept an ICW3 if programmed in the external cascade mode (as indicated in ICW1). ICW3 is used for specific programming within the cascade mode. The bits in ICW3 indicate which interrupt request inputs have a Slave cascaded to them. This will subsequently affect the interrupt vector generation during the interrupt acknowledge cycles as described previously.

ICW4

The ICW4 is accepted only if it was selected in ICW1. This command word register serves two functions:

— To select either the Automatic EOI mode or software EOI mode;

— To select if the Special Nested mode is to be used in conjunction with the cascade mode.

4.5.2 OPERATION CONTROL WORDS (OCW)

Once initialized by the ICW's, the interrupt banks will be operating in the Fully Nested Mode by default and they are ready to accept interrupt requests. However, the operations of each interrupt bank can be further controlled or modified by the use of OCW's. Three OCW's are available for programming various modes and commands. Note that all OCW's are 8-bit write-only registers.

The modes and operations controlled by the OCW's are:

— Fully Nested Mode;
— Rotating Priority Mode;
— Special Mask Mode;
— Poll Mode;
— EOI Commands;
— Read Status Commands.

OCW1

OCW1 is used solely for masking operations. It provides a direct link to the Interrupt Mask Register (IMR). The 80386 can write to this OCW register to enable or disable the interrupt inputs. Reading the pre-programmed mask can be done via the Interrupt Mask Register which will be discussed shortly.

OCW2

OCW2 is used to select End-Of-Interrupt, Automatic Priority Rotation, and Specific Priority Rotation operations. Associated commands and modes of these operations are selected using the different combinations of bits in OCW2.

Specifically, the OCW2 is used to:

— Designate an interrupt level (0–7) to be used to reset a specific ISR bit or to set a specific priority. This function can be enabled or disabled;

— Select which software EOI command (if any) is to be executed (i.e., Non-Specific or Specific EOI);

— Enable one of the priority rotation operations (i.e., Rotate On Non-Specific EOI, Rotate On Automatic EOI, or Rotate on Specific EOI).

OCW3

There are three main categories of operation that OCW3 controls. That are summarized as follows:

— To select and execute the Read Status Register Commands, either reading the Interrupt Request Register (IRR) or the In-Service Register (ISR);

— To issue the Poll Command. The Poll Command will override a Read Register Command if both functions are enabled simultaneously;

— To set or reset the Special Mask Mode.

4.5.3 POLL/INTERRUPT REQUEST/IN-SERVICE STATUS REGISTER

As the name implies, this 8-bit read-only register has multiple functions. Depending on the command issued in the OCW3, the content of this register reflects the result of the command executed. For a Poll Command, the register read contains the binary code of the highest priority level requesting service (if any). For a Read IRR Command, the register content will show the current pending interrupt request(s). Finally, for a Read ISR Command, this register will specify all interrupt levels which are being serviced.

4.5.4 INTERRUPT MASK REGISTER (IMR)

This is a read-only 8-bit register which, when read, will specify all interrupt levels within the same bank that are masked.

4.5.5 VECTOR REGISTER (VR)

Each interrupt request input has an 8-bit read/write programmable vector register associated with it. The registers should be programmed to contain the interrupt vector for the corresponding request. The contents of the Vector Register will be placed on the Data Bus during the INTA cycles as described previously.

4.6 Programming

Programming the 82380 PIC is accomplished by using two types of command words: ICW's and OCW's. All modes and commands explained in the previous sections are programmable using the ICW's and OCW's. The ICW's are issued from the 80386 in a sequential format and are used to setup the banks in the 82380 PIC in an initial state of operation. The OCW's are issued as needed to vary and control the 82380 PIC's operations.

Both ICW's and OCW's are sent by the 80386 to the interrupt banks via the Data Bus. Each bank distinguishes between the different ICW's and OCW's by the I/O address map, the sequence they are issued (ICW's only), and by some dedicated bits among the ICW's and OCW's.

All three interrupt banks are programmed in a similar way. Therefore, only a single bank will be described.

4.6.1 INITIALIZATION (ICW)

Before normal operation can begin, each bank must be initialized by programming a sequence of two to four bytes written into the ICW's.

Figure 4-6 shows the initialization flow for an interrupt bank. Both ICW1 and ICW2 must be issued for any form of operation. However, ICW3 and ICW4 are used only if designated in ICW1. Once initialized, if any programming changes within the ICW's are to be made, the entire ICW sequence must be reprogrammed, not just an individual ICW.

Note that although the ICW2's in the 82380 PIC do not affect the Bank's operation, they still must be programmed in order to preserve the compatibility with the 82C59A. The contents programmed are not relevant to the overall operations of the interrupt banks. Also, whenever one of the three ICW2's is programmed, an interrupt level 1.5 in Bank A will be generated. This interrupt request will be cleared upon reading of the ICW2 registers. Since the three ICW2's share the same interrupt level and the system may not know the origin of the interrupt, all three ICW2's must be read.

However, it is not necessary to provide an interrupt service routine for the ICW2 interrupt. One way to avoid this is as follows. At the beginning of the initialization of the interrupt banks, the 80386 interrupt should be disabled. After each ICW2 register write operation is performed during the initialization, the corresponding ICW2 register is read. This read operation will clear the interrupt request of the 82380. At the end of the initialization, the 80386 interrupt is reenabled. With this method, the 80386 will not detect the ICW2 interrupt request, thus eliminating the need of an interrupt service routine.

Certain internal setup conditions occur automatically within the interrupt bank after the first ICW (ICW1) has been issued. They are:

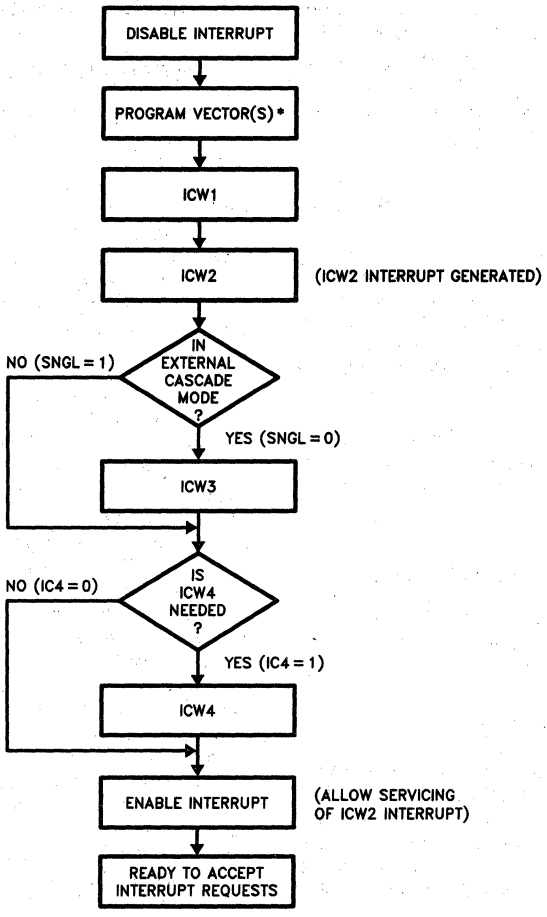
— The edge sensitive circuit is reset, which means that following initialization, an interrupt request input must make a HIGH-to-LOW transition to generate an interrupt;

— The Interrupt Mask Register (IMR) is cleared; that is, all interrupt inputs are enabled;

— IRQ7 input of each bank is assigned priority 7 (lowest);

— Special Mask Mode is cleared and Status Read is set to IRR;

— If no ICW4 is needed, then no Automatic-EOI is selected.



*ICW2 vector address must be programmed now.
Other vector addresses may be programmed via ICW2 interrupt service routine.

Figure 4-6. Initialization Sequence

4.6.2 VECTOR REGISTERS (VR)

Each interrupt request input has a separate Vector Register. These Vector Registers are used to store the pre-programmed vector number corresponding to their interrupt sources. In order to guarantee proper interrupt handling, all Vector Registers must be programmed with the predefined vector numbers. Since an interrupt request will be generated whenever an ICW2 is written during the initialization sequence, it is important that the Vector Register of IRQ1.5 in Bank A should be initialized and the interrupt service routine of this vector is set up before the ICW's are written.

4.6.3 OPERATION CONTROL WORDS (OCW)

After the ICW's are programmed, the operations of each interrupt controller bank can be changed by writing into the OCW's as explained before. There is no special programming sequence required for the OCW's. Any OCW may be written at any time in order to change the mode of or to perform certain operations on the interrupt banks.

4.6.3.1 Read Status and Poll Commands (OCW3)

Since the reading of IRR and ISR status as well as the result of a Poll Command are available on the

same read-only Status Register, a special Read Status/Poll Command must be issued before the Poll/Interrupt Request/In-Service Status Register is read. This command can be specified by writing the required control word into OCW3. As mentioned earlier, if both the Poll Command and the Status Read Command are enabled simultaneously, the Poll Command will override the Status Read. That is, after the command execution, the Status Register will contain the result of the Poll Command.

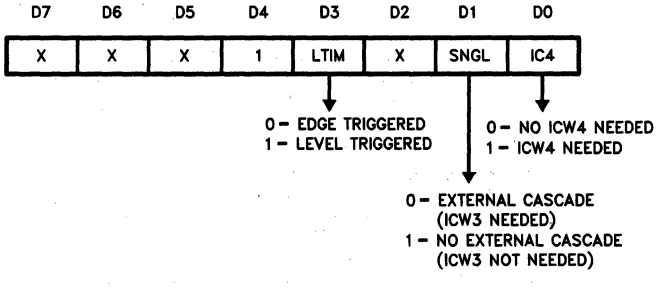
Note that for reading IRR and ISR, there is no need to issue a Read Status Command to the OCW3 every time the IRR or ISR is to be read. Once a Read Status Command is received by the interrupt bank, it 'remembers' which register is selected. However, this is not true when the Poll Command is used.

In the Poll Command, after the OCW3 is written, the 82380 PIC treats the next read to the Status Register as an interrupt acknowledge. This will set the appropriate IS bit if there is a request and read the priority level. Interrupt Request input status remains unchanged from the Poll Command to the Status Read.

In addition to the above read commands, the Interrupt Mask Register (IMR) can also be read. When read, this register reflects the contents of the preprogrammed OCW1 which contains information on which interrupt request(s) is(are) currently disabled.

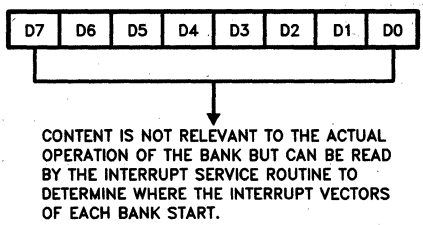
4.7 Register Bit Definition

INITIALIZATION COMMAND WORD 1 (ICW1)



290128-56

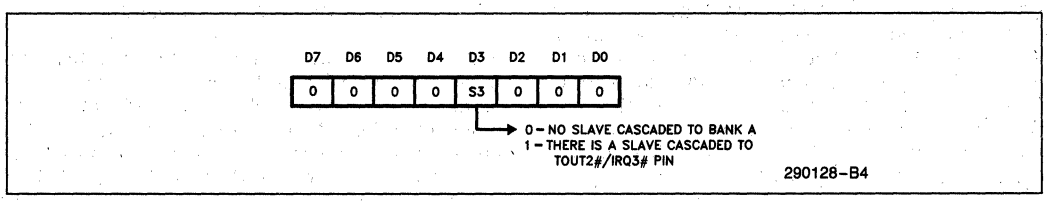
INITIALIZATION COMMAND WORD 2 (ICW2)



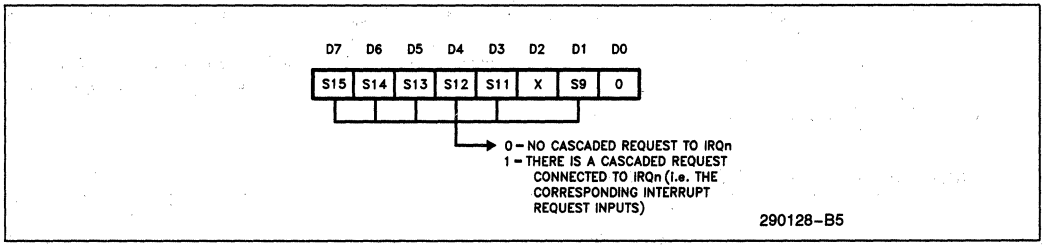
290128-57

INITIALIZATION COMMAND WORD 3 (ICW3)

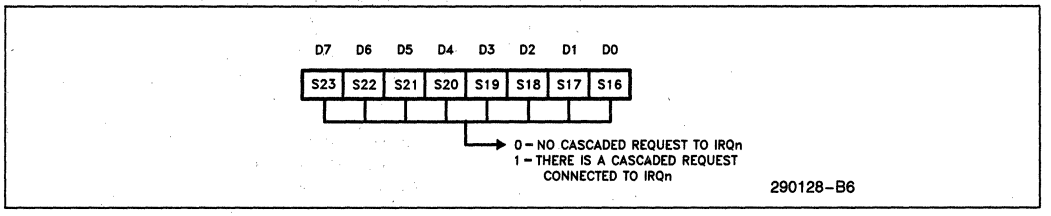
ICW3 for Bank A:



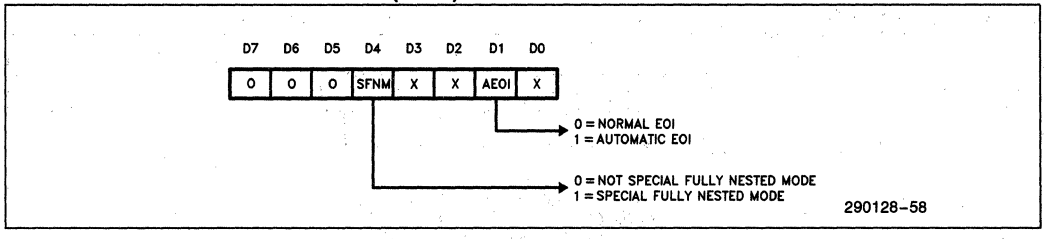
ICW3 for Bank B:



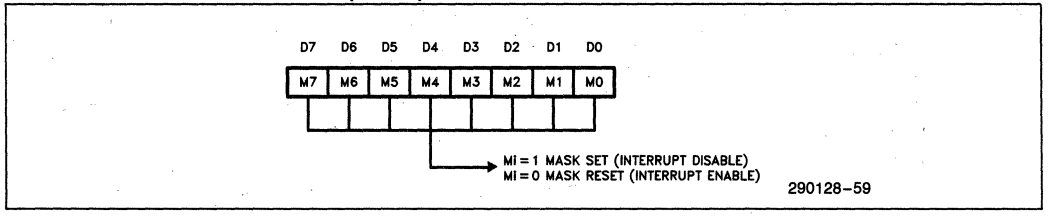
ICW3 for Bank C:



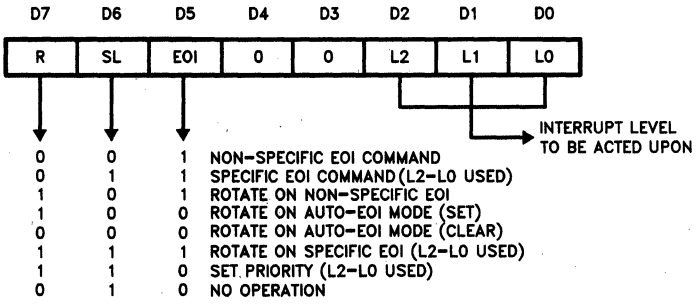
INITIALIZATION COMMAND WORD 4 (ICW4)



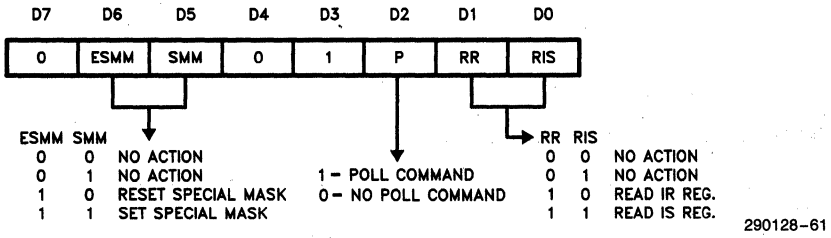
OPERATION CONTROL WORD 1 (OCW1)



OPERATION CONTROL WORD 2 (OCW2)



OPERATION CONTROL WORD 3 (OCW3)

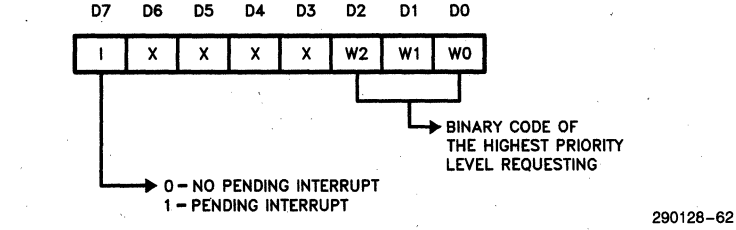


ESMM—Enable Special Mask Mode. When this bit is set to 1, it enables the SMM bit to set or reset the Special Mask Mode. When this bit is set to 0, SMM bit becomes don't care.

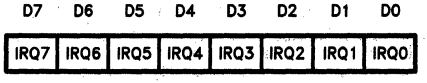
SMM—Special Mask Mode. If ESMM = 1 and SMM = 1, the interrupt controller bank will enter Special Mask Mode. If ESMM = 1 and SMM = 0, the bank will revert to normal mask mode. When ESMM = 0, SMM has no effect.

Poll/Interrupt Request/In-Service Status Register

POLL COMMAND STATUS



INTERRUPT REQUEST STATUS

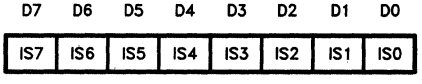


IF IRQ BIT IS: 0 – NO REQUEST
1 – REQUEST PENDING

290128-63

NOTE:
Although all Interrupt Request inputs are active LOW, the internal logical will invert the state of the pins so that when there is a pending interrupt request at the input, the corresponding IRQ bit will be set to HIGH in the Interrupt Request Status register.

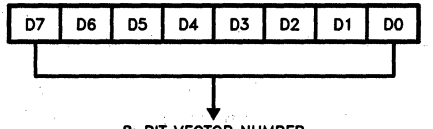
IN-SERVICE STATUS



IF IS_n BIT IS: 0 – NOT IN-SERVICE
1 – REQUEST IS IN-SERVICE

290128-64

VECTOR REGISTER (VR)



290128-65

4.8 Register Operational Summary

For ease of reference, Table 4-4 gives a summary of the different operating modes and commands with their corresponding registers.

Table 4-4 Register Operational Summary

Operational Description	Command Words	Bits
Fully Nested Mode	OCW-Default	—
Non-specific EOI Command	OCW2	EOI
Specific EOI Command	OCW2	SL, EOI, L0–L2
Automatic EOI Mode	ICW1, ICW4	IC4, AEOI
Rotate On Non-Specific EOI Command	OCW2	EOI
Rotate On Automatic EOI Mode	OCW2	R, SL, EOI
Set Priority Command	OCW2	L0–L2
Rotate On Specific EOI Command	OCW2	R, SL, EOI
Interrupt Mask Register	OCW1	M0–M7
Special Mask Mode	OCW3	ESMM, SMM
Level Triggered Mode	ICW1	LTIM
Edge Triggered Mode	ICW1	LTIM
Read Register Command, IRR	OCW3	RR, RIS
Read Register Command, ISR	OCW3	RR, RIS
Red IMR	IMR	M0–M7
Poll Command	OCW3	P
Special Fully Nested Mode	ICW2, ICW4	IC4, SFNM

5.0 PROGRAMMABLE INTERVAL TIMER

5.1 Functional Description

The 82380 contains four independently Programmable Interval Timers: Timer 0–3. All four timers are functionally compatible to the Intel 82C54. The first three timers (Timer 0–2) have specific functions. The fourth timer, Timer 3, is a general purpose timer. Table 5-1 depicts the functions of each timer. A brief description of each timer's function follows.

Table 5-1. Programmable Interval Timer Functions

Timer	Output	Function
0	IRQ8	Event Based IRQ8 Generator
1	TOUT1/REF#	Gen. Purpose/DRAM Refresh Req.
2	TOUT2#/IRQ3#	Gen. Purpose/Speaker Out/IRQ3#
3	TOUT3#	Gen. Purpose/IRQ0 Generator

TIMER 0— Event Based IRQ8 Generator

Timer 0 is intended to be used as an Event Counter. The output of this timer will generate an Interrupt Request 8 (IRQ8) upon a rising edge of the timer output (TOUT0). Typically, this timer is used to implement a time-of-day clock or system tick. The Timer 0 output is not available as an external signal.

TIMER 1— General Purpose/DRAM Refresh Request

The output of Timer 1, TOUT1, can be used as a general purpose timer or as a DRAM Refresh Request signal. The rising edge of this output creates a DRAM refresh request to the 82380 DRAM Refresh Controller. Upon reset, the Refresh Request function is disabled, and the output pin is the Timer 1 output.

TIMER 2—General Purpose/Speaker Out/IRQ3#

The Timer 2 output, TOUT2#, could be used to support tone generation to an external speaker. This pin is a bidirectional signal. When used as an input, a logic LOW asserted at this pin will generate an Interrupt Register 3 (IRQ3#) (see Programmable Interrupt Controller).

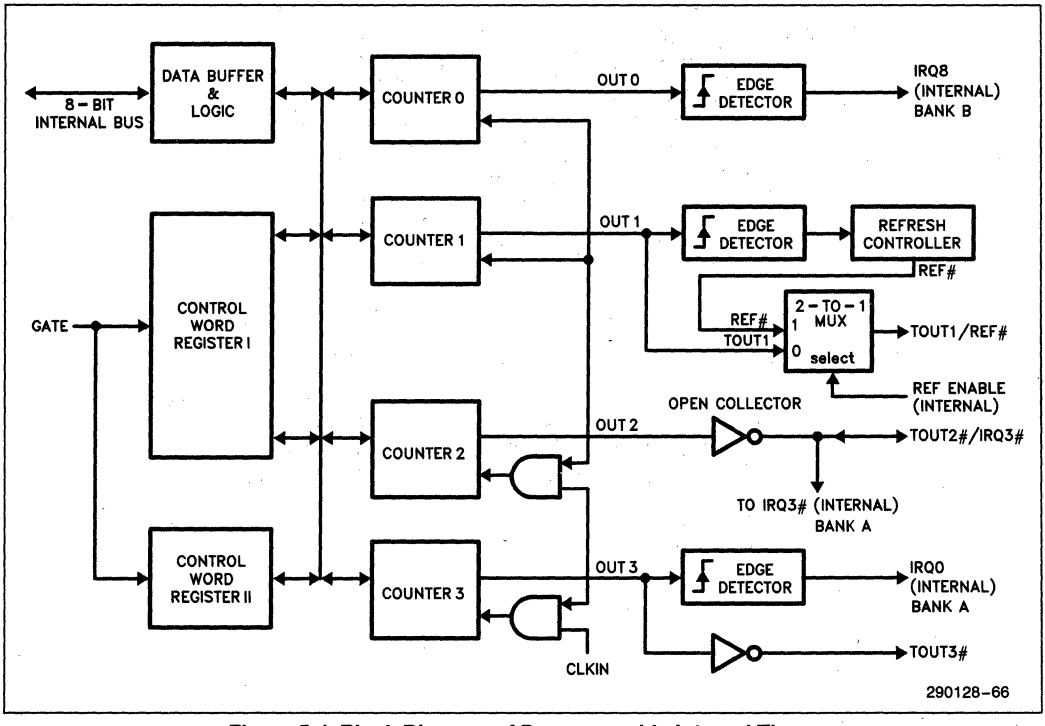


Figure 5-1. Block Diagram of Programmable Interval Timer

TIMER 3—General Purpose/Interrupt Request 0 Generator

The output of Timer 3 is fed to an edge detector and generates an Interrupt Request 0 (IRQ0) in the 82380. The inverted output of this timer (TOUT3#) is also available as an external signal for general purpose use.

5.1.1 INTERNAL ARCHITECTURE

The functional block diagram of the Programmable Interval Timer section is shown in Figure 5-1. Following is a description of each block.

DATA BUFFER & READ/WRITE LOGIC

This part of the Programmable Interval Timer is used to interface the four timers to the 82380 internal bus. The Data Buffer is for transferring commands and data between the 8-bit internal bus and the timers.

The Read/Write Logic accepts inputs from the internal bus and generates signals to control other functional blocks within the timer section.

CONTROL WORD REGISTERS I & II

The Control Word Registers are write-only registers. They are used to control the operating modes of the timers. Control Word Register I controls Timers 0, 1 and 2, and Control Word Register II controls Timer 3. Detailed description of the Control Word Registers will be included in the Register Set Overview section.

COUNTER 0, COUNTER 1, COUNTER 2, COUNTER 3

Counters 0, 1, 2, and 3 are the major parts of Timers 0, 1, 2, and 3, respectively. These four functional blocks are identical in operation, so only a single counter will be described. The internal block diagram of one counter is shown in Figure 5-2.

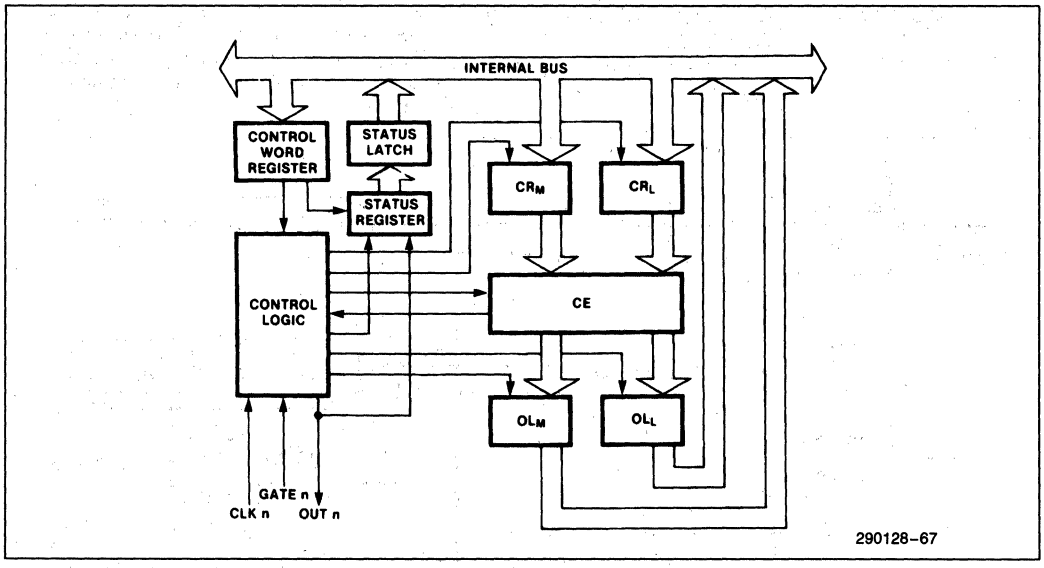


Figure 5-2. Internal Block Diagram of A Counter

The four counters share a common clock input (CLKIN), but otherwise are fully independent. Each counter is programmable to operate in a different Mode.

Although the Control Word Register is shown in the Figure 5-2, it is not part of the counter itself. Its programmed contents are used to control the operations of the counters.

The Status Register, when latched, contains the current contents of the Control Word Register and status of the output and Null Count Flag (see Read Back Command).

The Counting Element (CE) is the actual counter. It is a 16-bit presettable synchronous down counter.

The Output Latches (OL) contain two 8-bit latches (OLM and OLL). Normally, these latches 'follow' the content of the CE. OLM contains the most significant byte of the counter and OLL contains the least significant byte. If the Counter Latch Command is sent to the counter, OL will latch the present count until read by the 80386 and then return to follow the CE. One latch at a time is enabled by the timer's Control Logic to drive the internal bus. This is how the 16-bit Counter communicates over the 8-bit internal bus. Note that CE cannot be read. Whenever the count is read, it is one of the OL's that is being read.

When a new count is written into the counter, the value will be stored in the Count Registers (CR), and transferred to CE. The transferring of the contents from CR's to CE is defined as 'loading' of the counter. The Count Register contains two 8-bit registers: CRM (which contains the most significant byte) and CRL (which contains the least significant byte). Similar to the OL's, the Control Logic allows one register at a time to be loaded from the 8-bit internal bus. However, both bytes are transferred from the CR's to the CE simultaneously. Both CR's are cleared when the Counter is programmed. This way, if the Counter has been programmed for one byte count (either the most significant or the least significant byte only), the other byte will be zero. Note that CE cannot be written into directly. Whenever a count is written, it is the CR that is being written.

As shown in the diagram, the Control Logic consists of three signals: CLKIN, GATE, and OUT. CLKIN and GATE will be discussed in detail in the section that follows. OUT is the internal output of the counter. The external outputs of some timers (TOUT) are the inverted version of OUT (see TOUT1, TOUT2#, TOUT3#). The state of OUT depends on the mode of operation of the timer.

5.2 Interface Signals

5.2.1 CLKIN

CLKIN is an input signal used by all four timers for internal timing reference. This signal can be independent of the 82380 system clock, CLK2. In the following discussion, each 'CLK Pulse' is defined as the time period between a rising edge and a falling edge, in that order, of CLKIN.

During the rising edge of CLKIN, the state of GATE is sampled. All new counts are loaded and counters are decremented on the falling edge of CLKIN.

5.2.2 TOUT1, TOUT2#, TOUT3#

TOUT1, TOUT2# and TOUT3# are the external output signals of Timer 1, Timer 2 and Timer 3, respectively. TOUT2# and TOUT3# are the inverted signals of their respective counter outputs, OUT. There is no external output for Timer 0.

If Timer 2 is to be used as a tone generator of a speaker, external buffering must be used to provide sufficient drive capability.

The Outputs of Timer 2 and 3 are dual function pins. The output pin of Timer 2 (TOUT2#/IRQ3#), which is a bidirectional open-collector signal, can also be used as interrupt request input. When the interrupt function is enabled (through the Programmable Interrupt Controller), a LOW on this input will generate an Interrupt Request 3# to the 82380 Programmable Interrupt Controller. This pin has a weak internal pull-up resistor. To use the IRQ3# function, Timer 2 should be programmed so that OUT2 is LOW. Additionally, OUT3 of Timer 3 is connected to an edge detector which will generate an Interrupt Request 0 (IRQ0) to the 82380 after the rising edge of OUT3 (see Figure 5-1).

5.2.3 GATE

GATE is not an externally controllable signal. Rather, it can be software controlled with the Internal Control Port. The state of GATE is always sampled on the rising edge of CLKIN. Depending on the mode of operation, GATE is used to enable/disable counting or trigger the start of an operation.

For Timer 0 and 1, GATE is always enabled (HIGH). For Timer 2 and 3, GATE is connected to Bit 0 and 6, respectively, of an Internal Control Port (at address 61H) of the 82380. After a hardware reset, the state of GATE of Timer 2 and 3 is disabled (LOW).

5.3 Modes of Operation

Each timer can be independently programmed to operate in one of six different modes. Timers are programmed by writing a Control Word into the control Word Register followed by an Initial Count (see Programming).

The following are defined for use in describing the different modes of operation.

CLK Pulse—A rising edge, then a falling edge, in that order of CLKIN.
Trigger—A rising edge of a timer's GATE input.
Timer/Counter Loading—The transfer of a count from Count Register (CR) to Count Element (CE).

5.3.1 MODE 0—INTERRUPT ON TERMINAL COUNT

Mode 0 is typically used for event counting. After the Control Word is written, OUT is initially LOW, and will remain LOW until the counter reaches zero. OUT then goes HIGH and remains HIGH until a new count or a new Mode 0 Control Word is written into the counter.

In this mode, GATE = HIGH enables counting; GATE = LOW disables counting. However, GATE has no effect on OUT.

After the Control Word and initial count are written to a timer, the initial count will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not go HIGH until N + 1 CLK pulses after the initial count is written.

If a new count is written to the timer, it will be loaded on the next CLK pulse and counting will continue from the new count. If a two-byte count is written, the following happens:

1. Writing the first byte disables counting, OUT is set LOW immediately (i.e., no CLK pulse required).
2. Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the counting sequence to be synchronized by software. Again, OUT does not go HIGH until N + 1 CLK pulses after the new count of N is written.

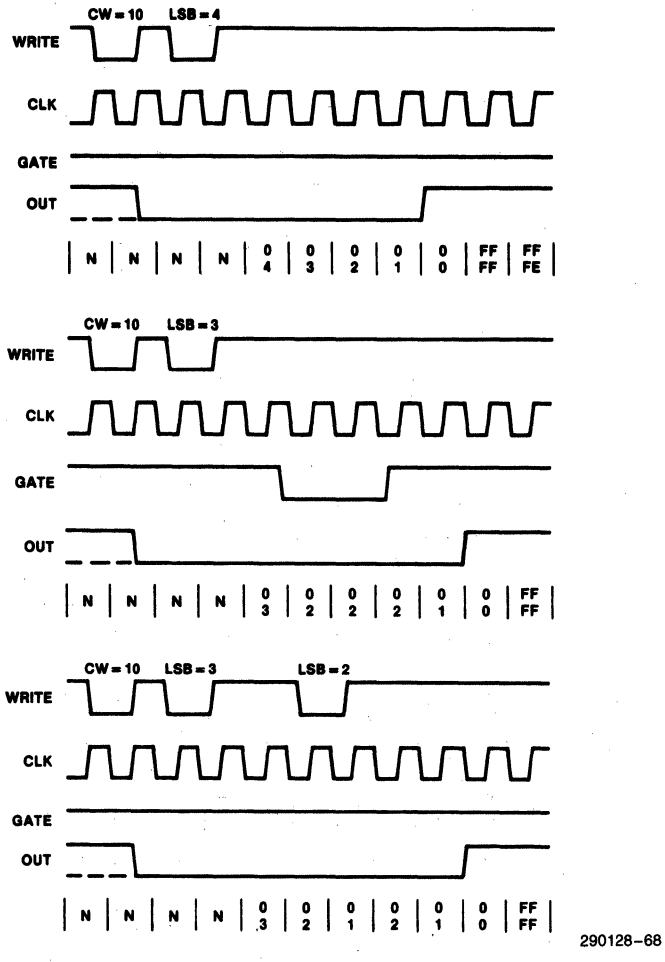
If an initial count is written while GATE is LOW, the counter will be loaded on the next CLK pulse. When GATE goes HIGH, OUT will go HIGH N CLK pulses later; no CLK pulse is needed to load the counter as this has already been done.

5.3.2 MODE 1—GATE RETRIGGERABLE ONE-SHOT

In this mode, OUT will be initially HIGH. OUT will go LOW on the CLK pulse following a trigger to start the one-shot operation. The OUT signal will then remain LOW until the timer reaches zero. At this point, OUT will stay HIGH until the next trigger comes in. Since the state of GATE signals of Timer 0 and 1 are internally set to HIGH.

After writing the Control Word and initial count, the timer is considered 'armed'. A trigger results in loading the timer and setting OUT LOW on the next CLK pulse. Therefore, an initial count of N will result in a one-shot pulse width of N CLK cycles. Note that this one-shot operation is retriggerable; i.e., OUT will remain LOW for N CLK pulses after every trigger. The one-shot operation can be repeated without rewriting the same count into the timer.

If a new count is written to the timer during a one-shot operation, the current one-shot pulse width will not be affected until the timer is retriggered. This is because loading of the new count to CE will occur only when the one-shot is triggered.



NOTES:
The following conventions apply to all mode timing diagrams.
1. Counters are programmed for binary (not BCD) counting and for reading/writing least significant byte (LSB) only.
2. The counter is always selected ($\overline{CS}$ always low).
3. CW stands for "Control Word"; CW = 10 means a control word of 10, Hex is written to the counter.
4. LSB stands for "least significant byte" of count.
5. Numbers below diagrams are count values.
   The lower number is the least significant byte.
   The upper number is the most significant byte. Since the counter is programmed to read/write LSB only, the most significant byte cannot be read.
   N stands for an undefined count.
   Vertical lines show transitions between count values.

Figure 5-3. Mode 0

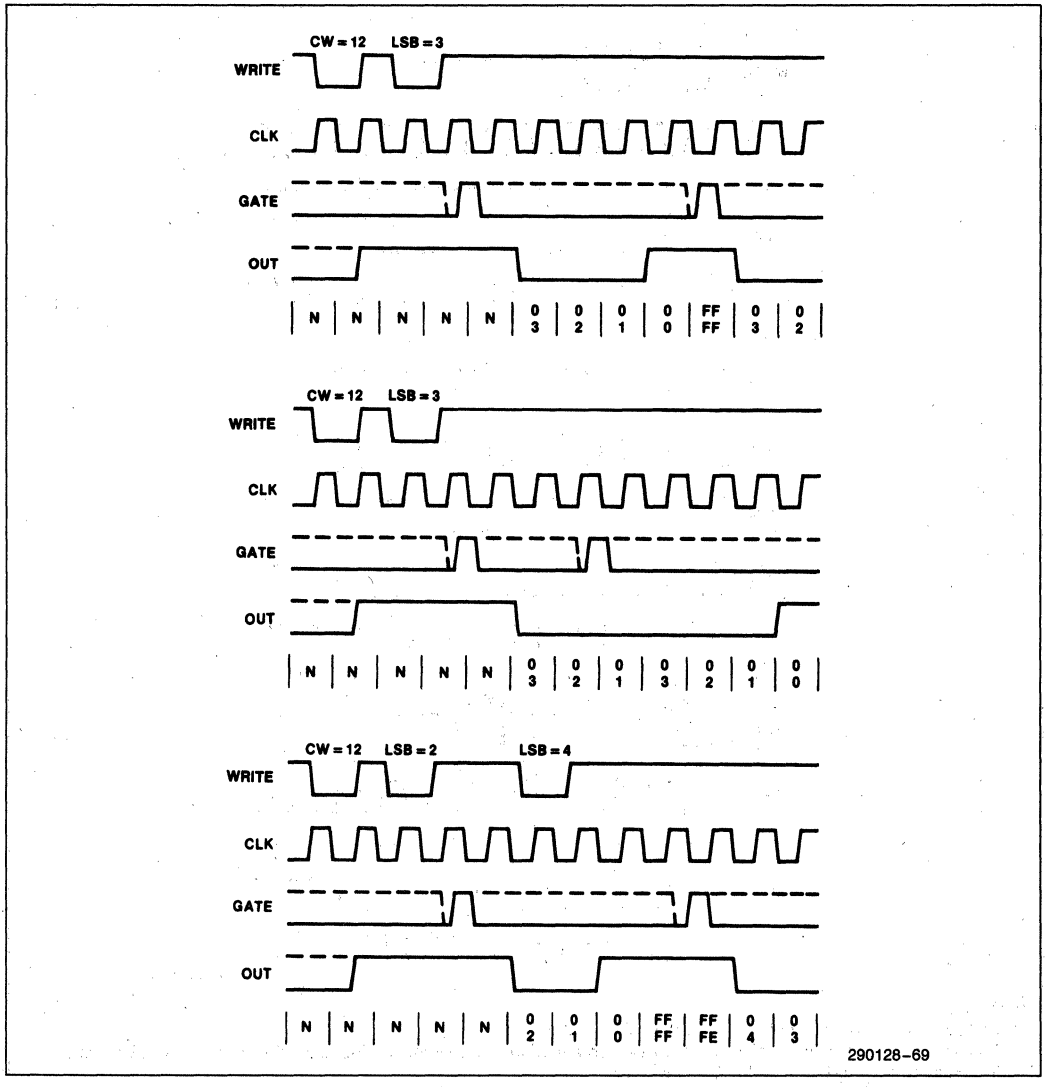
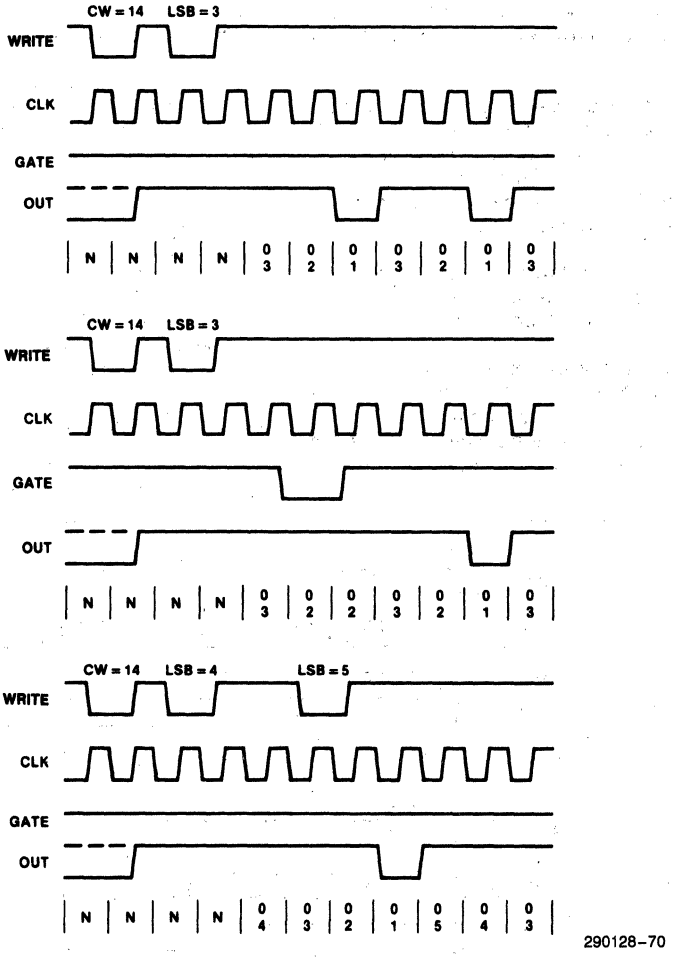


Figure 5-4. Mode 1

5.3.3 MODE 2—RATE GENERATOR

This mode is a divide-by-N counter. It is typically used to generate a Real Time Clock interrupt. OUT will initially be HIGH. When the initial count has decremented to 1, OUT goes LOW for one CLK pulse, then OUT goes HIGH again. Then the timer reloads the initial count and the process is repeated. In other words, this mode is periodic since the same sequence is repeated itself indefinitely. For an initial count of N, the sequence repeats every N CLK cycles.

Similar to Mode 0, GATE = HIGH enables counting, where GATE = LOW disables counting. If GATE goes LOW during an output pulse (LOW), OUT is set HIGH immediately. A trigger (rising edge on GATE) will reload the timer with the initial count on the next CLK pulse. Then, OUT will go LOW (for one CLK pulse) N CLK pulses after the new trigger. Thus, GATE can be used to synchronize the timer.



NOTE:
A GATE transition should not occur one clock prior to terminal count.

Figure 5-5. Mode 2

After writing a Control Word and initial count, the timer will be loaded on the next CLK pulse. OUT goes LOW (for the CLK pulse) N CLK pulses after the initial count is written. This is another way the timer may be synchronized by software.

Writing a new count while counting does not affect the current counting sequence because the new count will not be loaded until the end of the current counting cycle. If a trigger is received after writing a new count but before the end of the current period, the timer will be loaded with the new count on the next CLK pulse after the trigger, and counting will continue with the new count.

5.3.4 MODE 3—SQUARE WAVE GENERATOR

Mode 3 is typically used for Baud Rate generation. Functionally, this mode is similar to Mode 2 except for the duty cycle of OUT. In this mode, OUT will be initially HIGH. When half of the initial count has expired, OUT goes low for the remainder of the count.

The counting sequence will be repeated, thus this mode is also periodic. Note that an initial count of N results in a square wave with a period of N CLK pulses.

The GATE input can be used to synchronize the timer. GATE = HIGH enables counting; GATE = LOW disables counting. If GATE goes LOW while OUT is LOW, OUT is set HIGH immediately (i.e., no CLK pulse is required). A trigger reloads the timer with the initial count on the next CLK pulse.

After writing a Control Word and initial count, the timer will be loaded on the next CLK pulse. This allows the timer to be synchronized by software.

Writing a new count while counting does not affect the current counting sequence. If a trigger is received after writing a new count but before the end of the current half-cycle of the square wave, the timer will be loaded with the new count on the next CLK pulse and counting will continue from the new count. Otherwise, the new count will be loaded at the end of the current half-cycle.

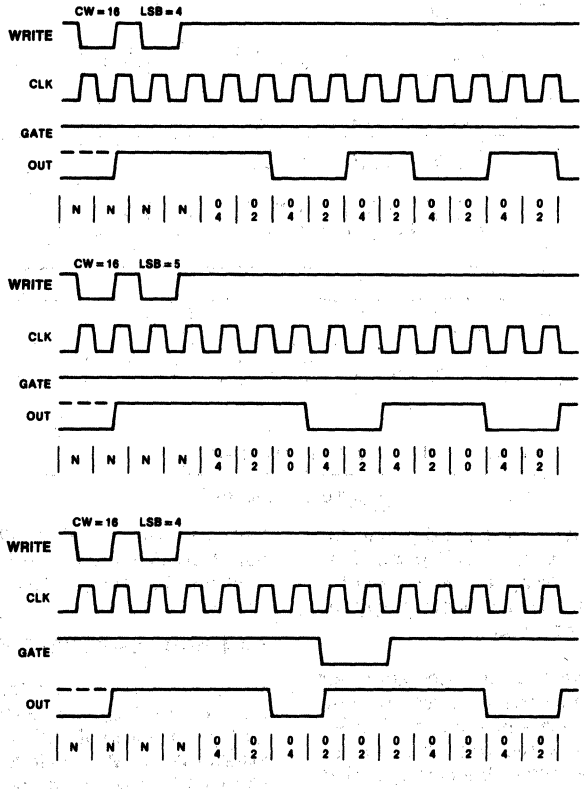
There is a slight difference in operation depending on whether the initial count is EVEN or ODD. The following description is to show exactly how this mode is implemented.

EVEN COUNTS:

OUT is initially HIGH. The initial count is loaded on one CLK pulse and is decremented by two on succeeding CLK pulses. When the count expires (decremented to 2), OUT changes to LOW and the timer is reloaded with the initial count. The above process is repeated indefinitely.

ODD COUNTS:

OUT is initially HIGH. The initial count minus one (which is an even number) is loaded on one CLK



290128-71

NOTE:
A-GATE transition should not occur one clock prior to terminal count.

Figure 5-6. Mode 3

pulse and is decremented by two on succeeding CLK pulses. One CLK pulse after the count expires (decremented to 2), OUT goes LOW and the timer is loaded with the initial count minus one again. Succeeding CLK pulses decrement the count by two. When the count expires, OUT goes HIGH immediately and the timer is reloaded with the initial count minus one. The above process is repeated indefinitely. So for ODD counts, OUT will be HIGH for (N + 1)/2 counts and LOW for (N − 1)/2 counts.

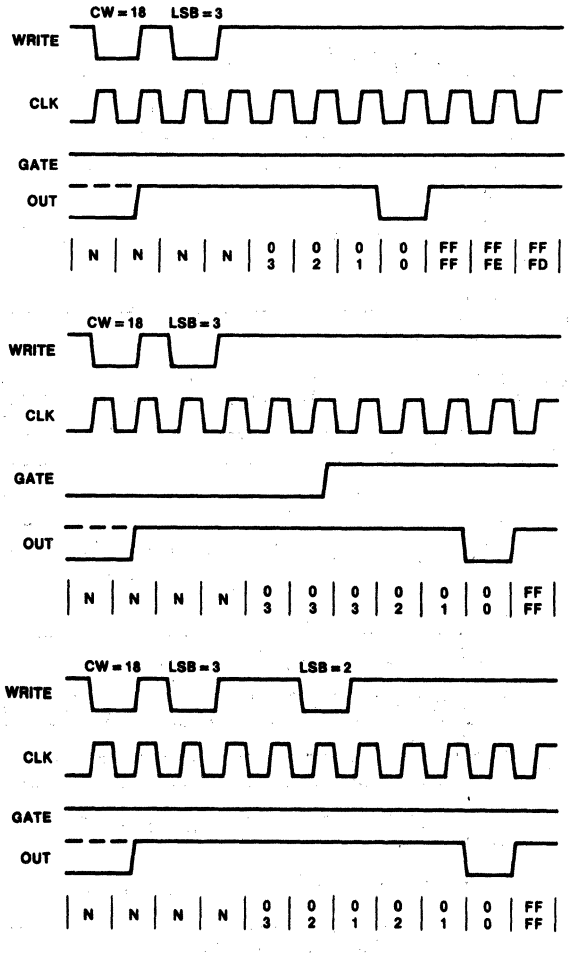
5.3.5 MODE 4—INITIAL COUNT TRIGGERED STROBE

This mode allows a strobe pulse to be generated by writing an initial count to the timer. Initially, OUT will be HIGH. When a new initial count is written into the timer, the counting sequence will begin. When the initial count expires (decremented to 1), OUT will go LOW for one CLK pulse and then go HIGH again.

Again, GATE = HIGH enables counting while GATE = LOW disables counting. GATE has no effect on OUT.

After writing the Control Word and initial count, the timer will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not strobe LOW until N + 1 CLK pulses after initial count is written.

If a new count is written during counting, it will be loaded in the next CLK pulse and counting will continue from the new count.



290128-72

Figure 5-7. Mode 4

If a two-byte count is written, the following will occur:

1. Writing the first byte has no effect on counting.
2. Writing the second byte allows the new count to be loaded on the next CLK pulse.

OUT will strobe LOW N + 1 CLK pulses after the new count of N is written. Therefore, when the strobe pulse will occur after a trigger depends on the value of the initial count loaded.

5.3.6 MODE 5—GATE RETRIGGERABLE STROBE

Mode 5 is very similar to Mode 4 except the count sequence is triggered by the GATE signal instead of by writing an initial count. Initially, OUT will be HIGH. Counting is triggered by a rising edge of GATE. When the initial count has expired (decremented to 1), OUT will go LOW for one CLK pulse and then go HIGH again.

After loading the Control Word and initial count, the Count Element will not be loaded until the CLK pulse after a trigger. This CLK pulse does not decrement the count. Therefore, for an initial count of N, OUT does not strobe LOW until N + 1 CLK pulses after a trigger.

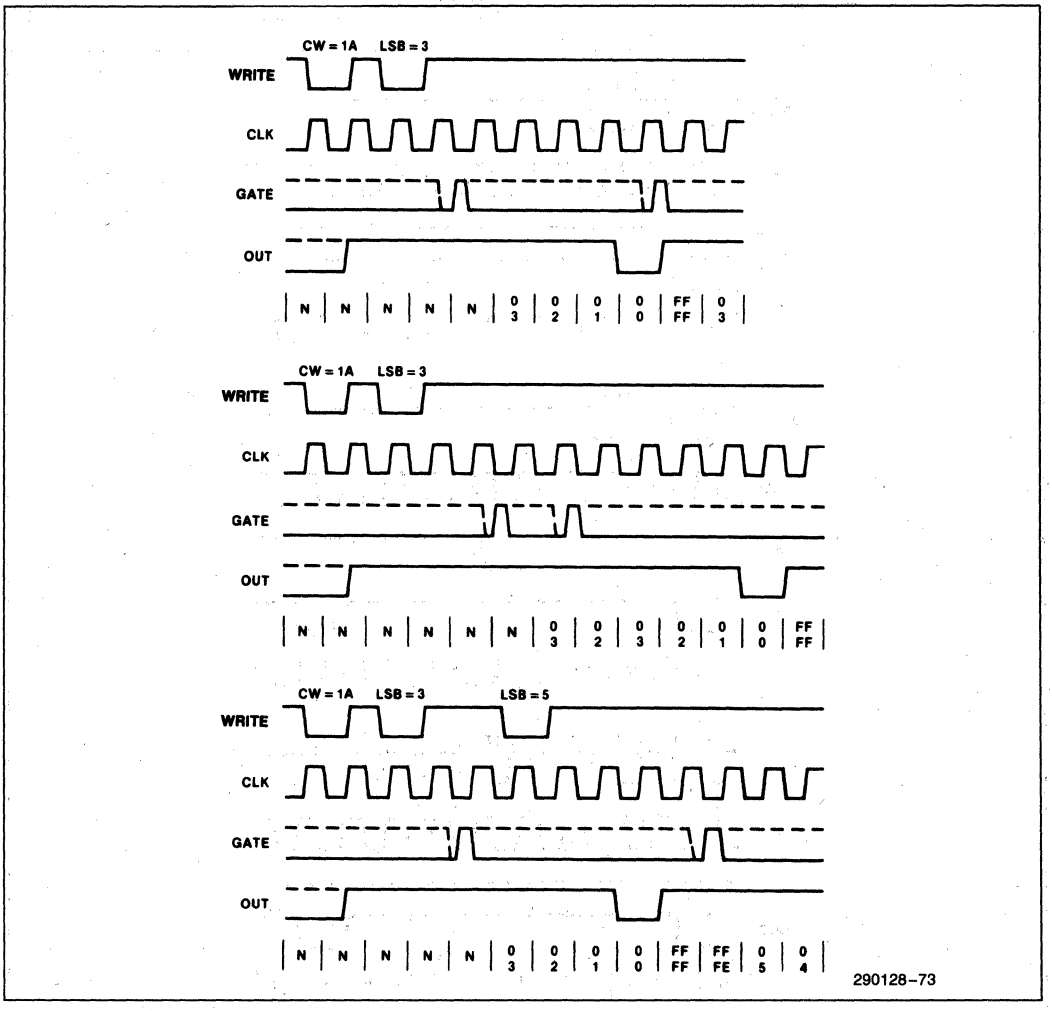


Figure 5-8. Mode 5

SUMMARY OF GATE OPERATIONS

Mode	GATE LOW or Going LOW	GATE Rising	GATE HIGH
0	Disable Count	No Effect	Enable Count
1	No Effect	1. Initiate Count 2. Reset Output After Next Clock	No Effect
2	1. Disable Count 2. Sets Output HIGH Immediately	Initiate Count	Enable Count
3	1. Disable Count 2. Sets Output HIGH Immediately	Initiate Count	Enable Count
4	Disable Count	No Effect	Enable Count
5	No Effect	Initiate Count	No Effect

The counting sequence is retriggerable. Every trigger will result in the timer being loaded with the initial count on the next CLK pulse.

If the new count is written during counting, the current counting sequence will not be affected. If a trigger occurs after the new count is written but before the current count expires, the timer will be loaded with the new count on the next CLK pulse and a new count sequence will start from there.

5.3.7 OPERATION COMMON TO ALL MODES

5.3.7.1 GATE

The GATE input is always sampled on the rising edge of CLKIN. In Modes 0, 2, 3 and 4, the GATE input is level sensitive. The logic level is sampled on the rising edge of CLKIN. In Modes 1, 2, 3 and 5, the GATE input is rising edge sensitive. In these modes, a rising edge of GATE (trigger) sets an edge sensitive flip-flop in the timer. The flip-flop is reset immediately after it is sampled. This way, a trigger will be detected no matter when it occurs; i.e., a HIGH logic level does not have to be maintained until the next rising edge of CLKIN. Note that in Modes 2 and 3, the GATE input is both edge and level sensitive.

5.3.7.2 Counter

New counts are loaded and counters are decremented on the falling edge of CLKIN. The largest possible initial count is 0. This is equivalent to 2**16 for binary counting and 10**4 for BCD counting.

Note that the counter does not stop when it reaches zero. In Modes 0, 1, 4, and 5, the counter 'wraps around' to the highest count: either FFFF Hex for binary counting or 9999 for BCD counting, and continues counting. Modes 2 and 3 are periodic. The counter reloads itself with the initial count and continues counting from there.

The minimum and maximum initial count in each counter depends on the mode of operation. They are summarized below.

Mode	Min	Max
0	1	0
1	1	0
2	2	0
3	2	0
4	1	0
5	1	0

5.4 Register Set Overview

The Programmable Interval Timer module of the 82380 contains a set of six registers. The port address map of these registers is shown in Table 5-2.

Table 5-2. Timer Register Port Address Map

Port Address	Description
40H	Counter 0 Register (read/write)
41H	Counter 1 Register (read/write)
42H	Counter 2 Register (read/write)
43H	Control Word Register I (Counter 0, 1 & 2) (write-only)
44H	Counter 3 Register (read/write)
45H	Reserved
46H	Reserved
47H	Control Word Register II (Counter 3) (write-only)

5.4.1 COUNTER 0, 1, 2, 3 REGISTERS

These four 8-bit registers are functionally identical. They are used to write the initial count value into the respective timer. Also, they can be used to read the latched count value of a timer. Since they are 8-bit registers, reading and writing of the 16-bit initial count must follow the count format specified in the Control Word Registers; i.e., least significant byte only, most significant byte only, or least significant byte then most significant byte (see Programming).

5.4.2 CONTROL WORD REGISTER I & II

There are two Control Word Registers associated with the Timer section. One of the two registers (Control Word Register I) is used to control the operations of Counters 0, 1, and 2 and the other (Control Word Register II) is for Counter 3. The major functions of both Control Word Registers are listed below:

— Select the timer to be programmed.

— Define which mode the selected timer is to operate in.

— Define the count sequence; i.e., if the selected timer is to count as a Binary Counter or a Binary Coded Decimal (BCD) Counter.

— Select the byte access sequence during timer read/write operations; i.e., least significant byte only, most significant byte only, or least significant byte first, then most significant byte.

Also, the Control Word Registers can be programmed to perform a Counter Latch Command or a Read Back Command which will be described later.

5.5 Programming

5.5.1 INITIALIZATION

Upon power-up or reset, the state of all timers is undefined. The mode, count value, and output of all timers are random. From this point on, how each timer operates is determined solely by how it is programmed. Each timer must be programmed before it can be used. Since the outputs of some timers can generate interrupt signals to the 82380, all timers should be initialized to a known state.

Timers are programmed by writing a Control Word into their respective Control Word Registers. Then, an Initial Count can be written into the corresponding Count Register. In general, the programming procedure is very flexible. Only two conventions need to be remembered:

1. For each timer, the Control Word must be written before the initial count is written.
2. The 16-bit initial count must follow the count format specified in the Control Word (least significant byte only, most significant byte only, or least significant byte first, followed by most significant byte).

Since the two Control Word Registers and the four Counter Registers have separate addresses, and each timer can be individually selected by the appropriate Control Word Register, no special instruction sequence is required. Any programming sequence that follows the conventions above is acceptable.

A new initial count may be written to a timer at any time without affecting the timer's programmed mode in any way. Count sequence will be affected as described in the Modes of Operation section. Note that the new count must follow the programmed count format.

If a timer is previously programmed to read/write two-byte counts, the following precaution applies. A program must not transfer control between writing the first and second byte to another routine which also writes into the same timer. Otherwise, the read/write will result in incorrect count.

Whenever a Control Word is written to a timer, all control logic for that timer(s) is immediately reset (i.e., no CLK pulse is required). Also, the corresponding output pin, TOUT(#), goes to a known initial state.

5.5.2 READ OPERATION

Three methods are available to read the current count as well as the status of each timer. They are: Read Counter Registers, Counter Latch Command and Read Back Command. Following is a description of these methods.

READ COUNTER REGISTERS

The current count of a timer can be read by performing a read operation on the corresponding Counter Register. The only restriction of this read operation is that the CLKIN of the timers must be inhibited by

using external logic. Otherwise, the count may be in the process of changing when it is read, giving an undefined result. Note that since all four timers are sharing the same CLKIN signal, inhibiting CLKIN to read a timer will unavoidably disable the other timers also. This may prove to be impractical. Therefore, it is suggested that either the Counter Latch Command or the Read Back Command be used to read the current count of a timer.

Another alternative is to temporarily disable a timer before reading its Counter Register by using the GATE input. Depending on the mode of operation, GATE = LOW will disable the counting operation. However, this option is available on Timer 2 and 3 only, since the GATE signals of the other two timers are internally enabled all the time.

COUNTER LATCH COMMAND

A Counter Latch Command will be executed whenever a special Control Word is written into a Control Word Register. Two bits written into the Control Word Register distinguish this command from a 'regular' Control Word (see Register Bit Definition). Also, two other bits in the Control Word will select which counter is to be latched.

Upon execution of this command, the selected counter's Output Latch (OL) latches the count at the time the Counter Latch Command is received. This count is held in the latch until it is read by the 80386, or until the timer is reprogrammed. The count is then unlatched automatically and the OL returns to 'following' the Counting Element (CE). This allows reading the contents of the counters 'on the fly' without affecting counting in progress. Multiple Counter Latch Commands may be used to latch more than one counter. Each latched count is held until it is read. Counter Latch Commands do not affect the programmed mode of the timer in any way.

If a counter is latched, and at some time later, it is latched again before the prior latched count is read, the second Counter Latch Command is ignored. The count read will then be the count at the time the first command was issued.

In any event, the latched count must be read according to the programmed format. Specifically, if the timer is programmed for two-byte counts, two bytes must be read. However, the two bytes do not have to be read right after the other. Read/write or programming operations of other timers may be performed between them.

Another feature of this Counter Latch Command is that read and write operations of the same timer may be interleaved. For example, if the timer is programmed for two-byte counts, the following sequence is valid.

1. Read least significant byte.
2. Write new least significant byte.
3. Read most significant byte.
4. Write new most significant byte.

If a timer is programmed to read/write two-byte counts, the following precaution applies. A program must not transfer control between reading the first and second byte to another routine which also reads from that same timer. Otherwise, an incorrect count will be read.

READ BACK COMMAND

The Read Back Command is another special Command Word operation which allows the user to read the current count value and/or the status of the selected timer(s). Like the Counter Latch Command, two bits in the Command Word identify this as a Read Back Command (see Register Bit Definition).

The Read Back Command may be used to latch multiple counter Output Latches (OL's) by selecting more than one timer within a Command Word. This single command is functionally equivalent to several Counter Latch Commands, one for each counter to be latched. Each counter's latched count will be held until it is read by the 80386 or until the timer is reprogrammed. The counter is automatically unlatched when read, but other counters remain latched until they are read. If multiple Read Back commands are issued to the same timer without reading the count, all but the first are ignored; i.e., the count read will correspond to the very first Read Back Command issued.

As mentioned previously, the Read Back Command may also be used to latch status information of the selected timer(s). When this function is enabled, the status of a timer can be read from the Counter Register after the Read Back Command is issued. The status information of a timer includes the following:

1. Mode of timer:

   This allows the user to check the mode of operation of the timer last programmed.

2. State of TOUT pin of the timer:

   This allows the user to monitor the counter's output pin via software, possibly eliminating some hardware from a system.

3. Null Count/Count available:

   The Null Count Bit in the status byte indicates if the last count written to the Count Register (CR) has been loaded into the Counting Element (CE). The exact time this happens depends on the mode of the timer and is described in the Programming section. Until the count is loaded into the Counting Element (CE), it cannot be read from the timer. If the count is latched or read before this occurs, the count value will not reflect the new count just written.

If multiple status latch operations of the timer(s) are performed without reading the status, all but the first command are ignored; i.e., the status read in will correspond to the first Read Back Command issued.

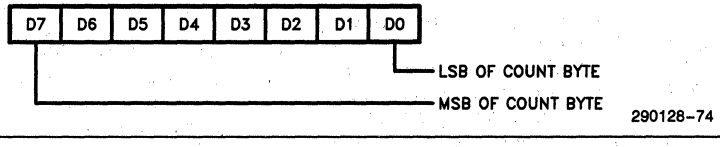
Both the current count and status of the selected timer(s) may be latched simultaneously by enabling both functions in a single Read Back Command. This is functionally the same as issuing two separate Read Back Commands at once. Once again, if multiple read commands are issued to latch both the count and status of a timer, all but the first command will be ignored.

If both count and status of a timer are latched, the first read operation of that timer will return the latched status, regardless of which was latched first. The next one or two (if two count bytes are to be read) read operations return the latched count. Note that subsequent read operations on the Counter Register will return the unlatched count (like the first read method discussed).

5.6 Register Bit Definitions

COUNTER 0, 1, 2, 3 REGISTER (READ/WRITE)

Port Address	Description
40H	Counter 0 Register (read/write)
41H	Counter 1 Register (read/write)
42H	Counter 2 Register (read/write)
44H	Counter 3 Register (read/write)
45H	Reserved
46H	Reserved

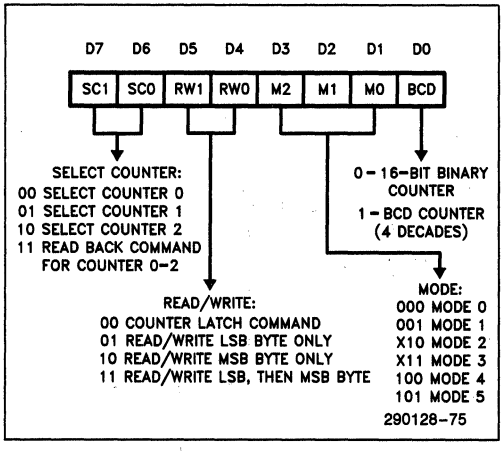


Note that these 8-bit registers are for writing and reading of one byte of the 16-bit count value, either the most significant or the least significant byte.

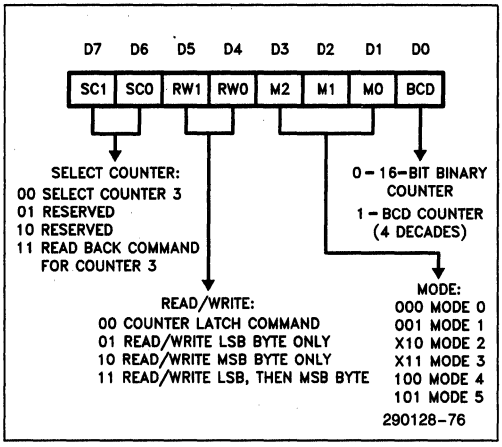
CONTROL WORD REGISTER I & II (WRITE-ONLY)

Port Address	Description
43H	Control Word Register I (Counter 0, 1, 2) (write-only)
47H	Control Word Register II (Counter 3) (write-only)

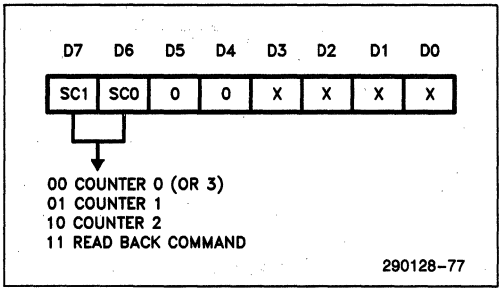
CONTROL WORD REGISTER I



CONTROL WORD REGISTER II



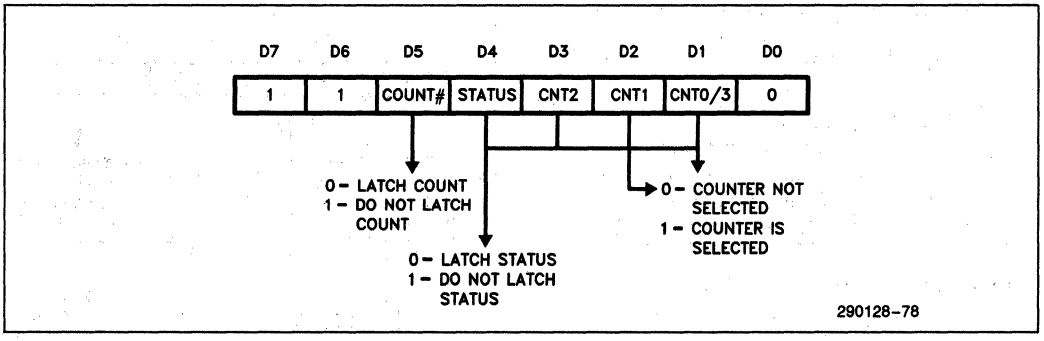
COUNTER LATCH COMMAND FORMAT
(Write to Control Word Register)



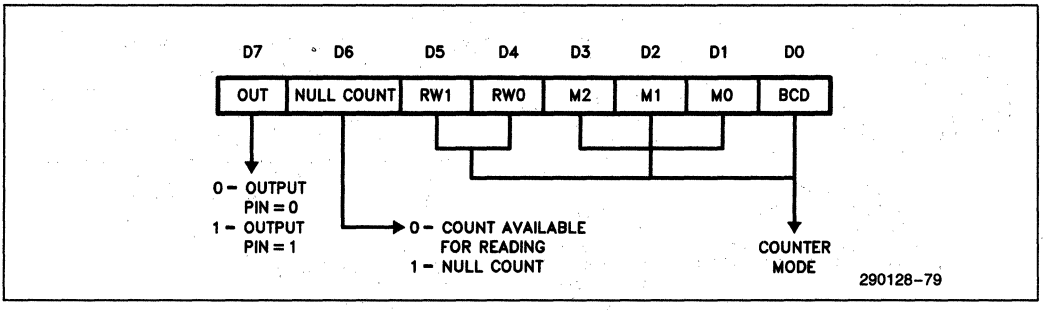
Mode	Timer				Gate Trigger		
	0	1	2	3	Edge	Level	
0						X	Interrupt on Terminal Count
1	NA	NA	①	①	X		Gate Retriggerable One Shot
2					X	X	Rate Generator
3					X	X	Square Wave Generator
4						X	Initial Count Triggered Strobe
5	NA	NA	①	①	X		Gate Retriggerable Strobe

① = Must use Port 61 to generate ⌐ edge.
NA = Not Applicable

READ BACK COMMAND FORMAT
(Write to Control Word Register)



STATUS FORMAT
(Returned from Read Back Command)



6.0 WAIT STATE GENERATOR

6.1 Functional Description

The 82380 contains a programmable Wait State Generator which can generate a pre-programmed number of wait states during both CPU and DMA initiated bus cycles. This Wait State Generator is capable of generating 1 to 16 wait states in non-pipelined mode, and 0 to 15 wait states in pipelined mode. Depending on the bus cycle type and the two Wait State Control inputs (WSC 0–1), a pre-programmed number of wait states in the selected Wait State Register will be generated.

The Wait State Generator can also be disabled to allow the use of devices capable of generating their own READY# signals. Figure 6-1 is a block diagram of the Wait State Generator.

6.2 Interface Signals

The following describes the interface signals which affect the operation of the Wait State Generator. The READY#, WSC0 and WSC1 signals are inputs. READYO# is the ready output signal to the host processor.

6.2.1 READY#

READY# is an active LOW input signal which indicates to the 82380 the completion of a bus cycle. In the Master mode (e.g., 82380 initiated DMA transfer), this signal is monitored to determine whether a peripheral or memory needs wait states inserted in the current bus cycle. In the Slave mode, it is used (together with the ADS# signal) to trace CPU bus cycles to determine if the current cycle is pipelined.

6.2.2 READYO#

READYO# (Ready Out#) is an active LOW output signal and is the output of the Wait State Generator. The number of wait states generated depends on the WSC(0-1) inputs. Note that special cases are handled for access to the 82380 internal registers and for the Refresh cycles. For 82380 internal register access, READYO# will be delayed to take into account the command recovery time of the register. One or more wait states will be generated in a pipelined cycle. During refresh, the number of wait states will be determined by the preprogrammed value in the Refresh Wait State Register.

In the simplest configuration, READYO# can be connected to the READY# input of the 82380 and the 80386 CPU. This is, however, not always the case. If external circuitry is to control the READY# inputs as well, additional logic will be required (see Application Issues).

6.2.3 WSC(0-1)

These two Wait State Control inputs select one of the three pre-programmed 8-bit Wait State Registers which determines the number of wait states to be generated. The most significant half of the three Wait State Registers corresponds to memory accesses, the least significant half to I/O accesses. The combination WSC(0-1) = 11 disables the Wait State Generator.

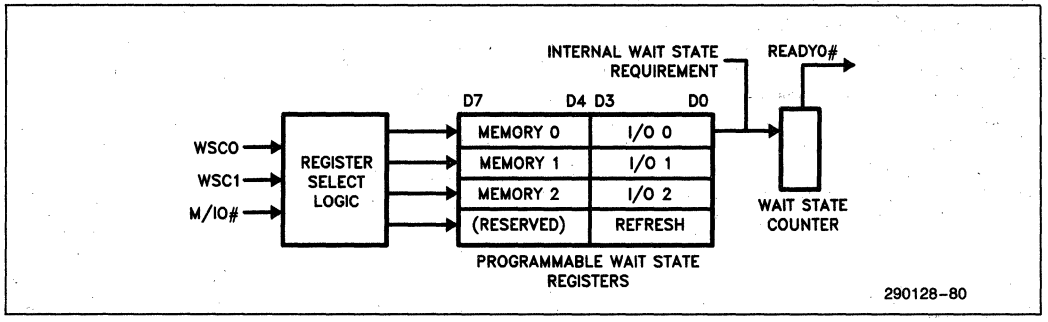


Figure 6-1. Wait State Generator Block Diagram

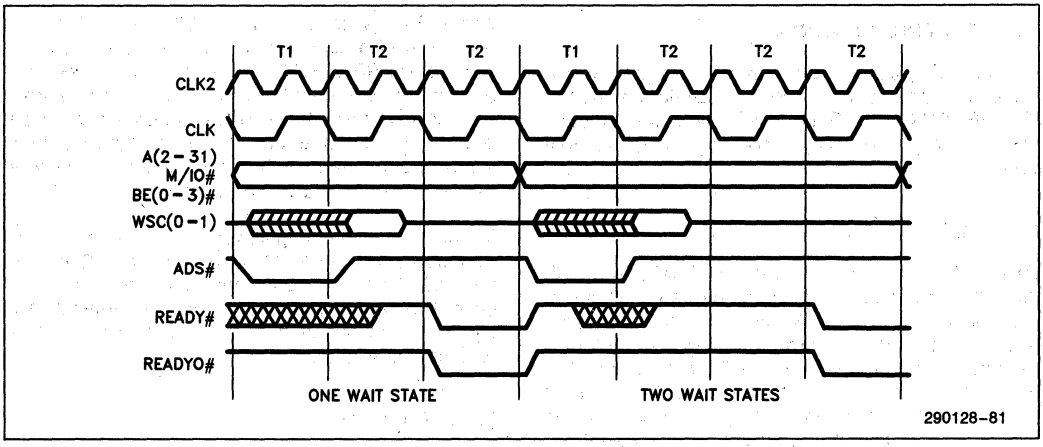


Figure 6-2. Wait States in Non-Pipelined Cycles

6.3 Bus Function

6.3.1 WAIT STATES IN NON-PIPELINED CYCLE

The timing diagram of two typical non-pipelined cycles with 82380 generated wait states is shown in Figure 6-2. In this diagram, it is assumed that the internal registers of the 82380 are not addressed. During the first T2 state of each bus cycle, the Wait State Control and the M/IO# inputs are sampled to determine which Wait State Register (if any) is selected. If the WSC inputs are active (i.e., not both are driven HIGH), the pre-programmed number of wait states corresponding to the selected Wait State Register will be requested. This is done by driving the READYO# output HIGH during the end of each T2 state.

The WSC(0-1) inputs need only be valid during the very first T2 state of each non-pipelined cycle. As a general rule, the WSC inputs are sampled on the rising edge of the next clock (82384 CLK) after the last state when ADS# (Address Status) is asserted.

The number of wait states generated depends on the type of bus cycle, and the number of wait states requested. The various combinations are discussed below.

1. Access the 82380 internal registers: 2 to 5 wait states, depending upon the specific register addressed. Some back-to-back sequences to the Interrupt Controller will require 7 wait states.
2. Interrupt Acknowledge to the 82380: 5 wait states.
3. Refresh: As programmed in the Refresh Wait State Register (see Register Set Overview). Note that if WSC(0-1) = 11, READYO# will stay inactive.
4. Other bus cycles: Depending on WSC(0-1) and M/IO# inputs, these inputs select a Wait State Register in which the number of wait states will be equal to the pre-programmed wait state count in the register plus 1. The Wait State Register selection is defined as follows (Table 6-1).

Table 6-1. Wait State Register Selection

M/IO#	WSC(1-0)	Register Selected
0	00	WAIT REG 0 (I/O half)
0	01	WAIT REG 1 (I/O half)
0	10	WAIT REG 2 (I/O half)
1	00	WAIT REG 0 (MEM half)
1	01	WAIT REG 1 (MEM half)
1	10	WAIT REG 2 (MEM half)
X	11	Wait State Gen. Disabled

The Wait State Control signals, WSC(0-1), can be generated with the address decode and the Read/Write control signals as shown in Figure 6-3.

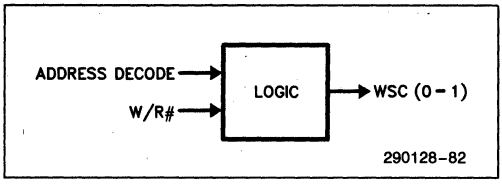


Figure 6-3. WSC(0-1) Generation

Note that during HALT and SHUTDOWN, the number of wait states will depend on the WSC(0-1) inputs, which will select the memory half of one of the Wait State Registers (see CPU Reset and Shutdown Detect).

6.3.2 WAIT STATES IN PIPELINED CYCLE

The timing diagram of two typical pipelined cycles with 82380 generated wait states is shown in Figure 6-4. Again, in this diagram, it is assumed that the 82380 internal registers are not addressed. As defined in the timing of the 80386 processor, the Address (A 2-31), Byte Enable (BE 0-3), and other control signals (M/IO#, ADS#) are asserted one T state earlier than in a non-pipelined cycle; i.e., they are asserted at T2P. Similar to the non-pipelined case, the Wait State Control (WSC) inputs are sampled in the middle of the state after the last state when the ADS# signal is asserted. Therefore, the WSC inputs should be asserted during the T1P state of each pipelined cycle (which is one T state earlier than in the non-pipelined cycle).

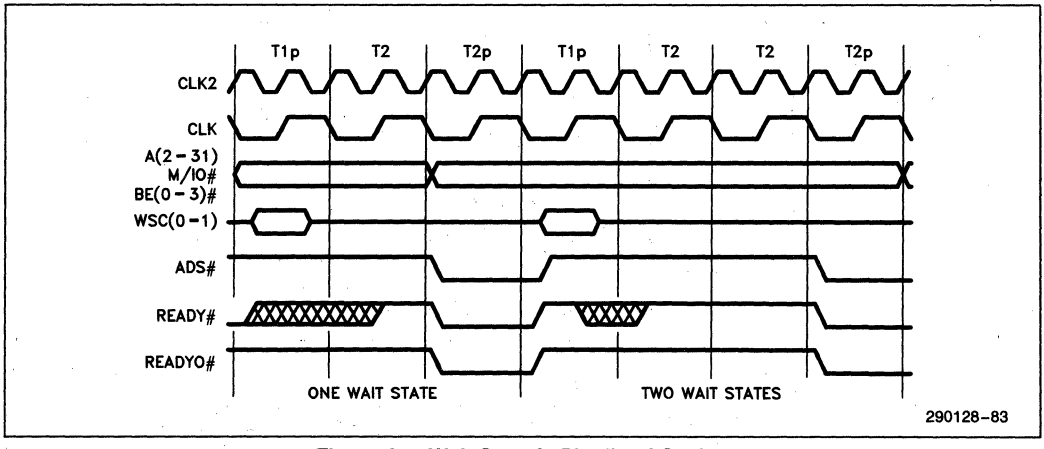


Figure 6-4. Wait State in Pipelined Cycles

The number of wait states generated in a pipelined cycle is selected in a similar manner as in the non-pipelined case discussed in the previous section. The only difference here is that the actual number of wait states generated will be one less than that of the non-pipelined cycle. This is done automatically by the Wait State Generator.

6.3.3 EXTENDING AND EARLY TERMINATING BUS CYCLE

The 82380 allows external logic to either add wait states or cause early termination of a bus cycle by controlling the READY# input to the 82380 and the host processor. A possible configuration is shown in Figure 6-5.

The EXT. RDY# (External Ready) signal of Figure 6-5 allows external devices to cause early termination of a bus cycle. When this signal is asserted LOW, the output of the circuit will also go LOW (even though the READYO# of the 82380 may still be HIGH). This output is fed to the READY# input of the 80386 and the 82380 to indicate the completion of the current bus cycle.

Similarly, the EXT. NOT READY (External Not Ready) signal is used to delay the READY# input of the processor and the 82380. As long as this signal is driven HIGH, the output of the circuit will drive the READY# input HIGH. This will effectively extend the duration of a bus cycle. However, it is important to note that if the two-level logic is not fast enough to satisfy the READY# setup time, the OR gate should be eliminated. Instead, the 82380 Wait State Generator can be disabled by driving both WSC(0–1) HIGH. In this case, the addressed memory or I/O device should activate the external READY# input whenever it is ready to terminate the current bus cycle.

Figure 6-6 and 6-7 show the timing relationships of the ready signals for the early termination and extension of the bus cycles. Section 6.7, Application Issues, contains a detailed timing analysis of the external circuit.

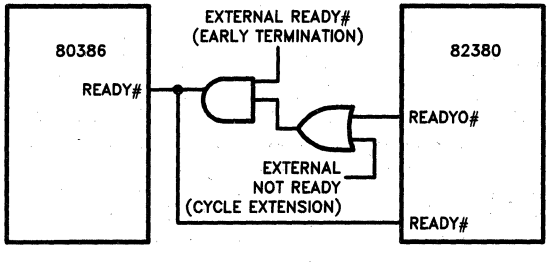


Figure 6-5. External 'READY' Control Logic

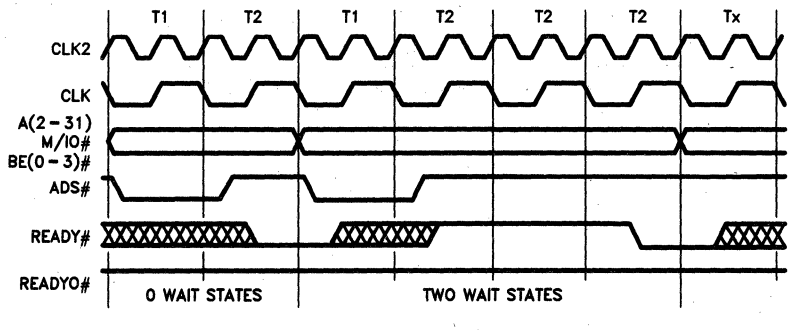


Figure 6-6. Early Termination of Bus Cycle By 'READY#'

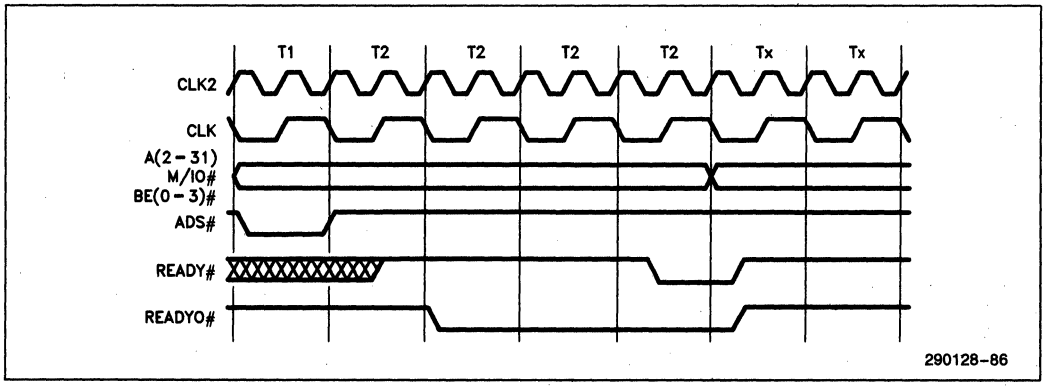


Figure 6-7. Extending Bus Cycle by 'READY#'

Due to the following implications, it should be noted that early termination of bus cycles in which 82380 internal registers are accessed is not recommended.

1. Erroneous data may be read from or written into the addressed register.
2. The 82380 must be allowed to recover either before HLDA (Hold Acknowledge) is asserted or before another bus cycle into an 82380 internal register is initiated.

   The recovery time, in bus periods, equals the remaining wait states that were avoided plus 4.

6.4 Register Set Overview

Altogether, there are four 8-bit internal registers associated with the Wait State Generator. The port address map of these registers is shown below in Table 6-2. A detailed description of each follows.

Table 6-2. Register Address Map

Port Address	Description
72H	Wait State Reg 0 (read/write)
73H	Wait State Reg 1 (read/write)
74H	Wait State Reg 2 (read/write)
75H	Ref. Wait State Reg (read/write)

WAIT STATE REGISTER 0, 1, 2

These three 8-bit read/write registers are functionally identical. They are used to store the pre-programmed wait state count. One half of each register contains the wait state count for I/O accesses while the other half contains the count for memory accesses. The total number of wait states generated will depend on the type of bus cycle. For a non-pipelined cycle, the actual number of wait states requested is equal to the wait state count plus 1. For a pipelined cycle, the number of wait states will be equal to the wait state count in the selected register. Therefore, the Wait State Generator is capable of generating 1 to 16 wait states in non-pipelined mode, and 0 to 15 wait states in pipelined mode.

Note that the minimum wait state count in each register is 0. This is equivalent to 0 wait states for a pipelined cycle and 1 wait state for a non-pipelined cycle.

REFRESH WAIT STATE REGISTER

Similar to the Wait State Registers discussed above, this 4-bit register is used to store the number of wait states to be generated during the DRAM refresh cycle. Note that the Refresh Wait State Register is not selected by the WSC inputs. It will automatically be

chosen whenever a DRAM refresh cycle occurs. If the Wait State Generator is disabled during the refresh cycle (WSC(0-1) = 11), READYO# will stay inactive and the Refresh Wait State Register is ignored.

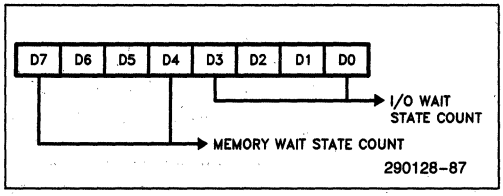
6.5 Programming

Using the Wait State Generator is relatively straightforward. No special programming sequence is required. In order to ensure the expected number of wait states will be generated when a register is selected, the registers to be used must be programmed after power-up by writing the appropriate wait state count into each register. Note that upon hardware reset, all Wait State Registers are initialized with the value FFH, giving the maximum number of wait states possible. Also, each register can be read to check the wait state count previously stored in the register.

6.6 Register Bit Definition

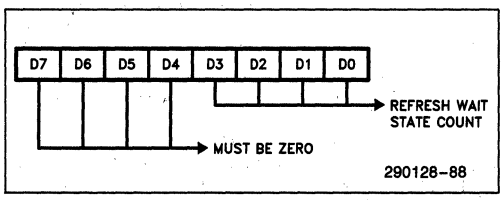
WAIT STATE REGISTER 0, 1, 2

Port Address	Description
72H	Wait State Register 0 (read/write)
73H	Wait State Register 1 (read/write)
74H	Wait State Register 2 (read/write)



REFRESH WAIT STATE REGISTER

Port Address: 75H (Read/Write)



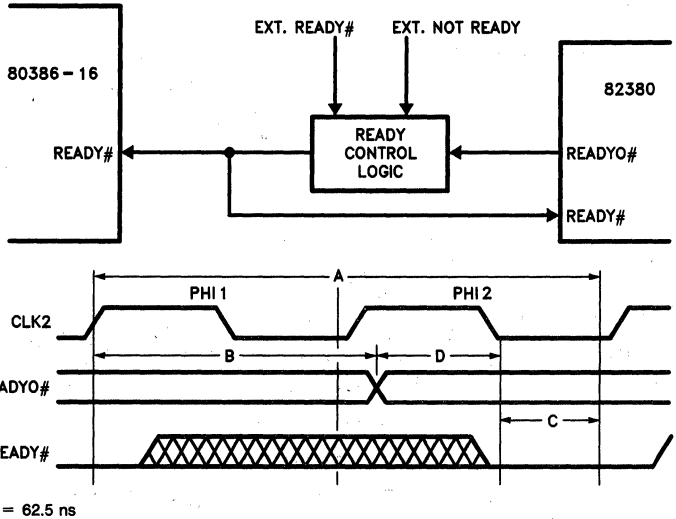
6.7 Application Issues

6.7.1 EXTERNAL 'READY' CONTROL LOGIC

As mentioned in section 6.3.3, wait state cycles generated by the 82380 can be terminated early or extended longer by means of additional external logic (see Figure 6-5). In order to ensure that the READY# input timing requirement of the 80386 and the 82380 is satisfied, special care must be taken when designing this external control logic. This section addresses the design requirements.

A simplified block diagram of the external logic along with the READY# tiiming diagram is shown in Figure 6-8. The purpose is to determine the maximum delay time allowed in the external control logic in order to satisfy the READY# setup time.

First, it will be assumed that the 80386 is running at 16 MHz (i.e., CLK2 and 32 MHz). Therefore, one bus state (two CLK2 periods) will be equivalent to 62.5 nsec. According to the AC specifications of the 82380, the maximum delay time for valid READYO# signal is 31 ns after the rising edge of CLK2 in the beginning of T2 (for non-pipelined cycle) or T2P (for pipelined cycle). Also, the minimum READY# setup time of the 80386 and the 82380 should be 20 ns before the rising edge of CLK2 at the beginning of the next bus state. This limits the total delay time for the external READY# control logic to be 11 ns (62.5-31-21) in order to meet the READY# setup timing requirement.



A = PHI1 + PHI2 = 62.5 ns
B = Maximum READYO# Valid Delay = 31 ns
C = READY# Set-up Time = 21 ns
D = Maximum Ready Control Logic Delay = A – B – C = 11 ns

Figure 6-8. 'READY' Timing Consideration

7.0 DRAM REFRESH CONTROLLER

7.1 Functional Description

The 82380 DRAM Refresh Controller consists of a 24-bit Refresh Address Counter and Refresh Request logic for DRAM refresh operations (see Figure 7-1). TIMER 1 can be used as a trigger signal to the DRAM Refresh Request logic. The Refresh Bus Size can be programmed to be 8-, 16-, or 32-bit wide. Depending on the Refresh Bus Size, the Refresh Address Counter will be incremented with the appropriate value after every refresh cycle. The internal logic of the 82380 will give the Refresh operation the highest priority in the bus control arbitration process. Bus control is not released and re-requested if the 82380 is already a bus master.

7.2 Interface Signals

7.2.1 TOUT1/REF#

The dual function output pin of TIMER 1 (TOUT1/REF#) can be programmed to generate DRAM Refresh signal. If this feature is enabled, the rising edge of TIMER 1 output (TOUT1) will trigger the DRAM Refresh Request logic. After some delay for gaining access of the bus, the 82380 DRAM Controller will generate a DRAM Refresh signal by driving REF# output LOW. This signal is cleared after the refresh cycle has taken place, or by a hardware reset.

If the DRAM Refresh feature is disabled, the TOUT1/REF# output pin is simply the TIMER 1 output. Detailed information of how TIMER 1 operates is discussed in section 6—Programmable Interval Timer, and will not be repeated here.

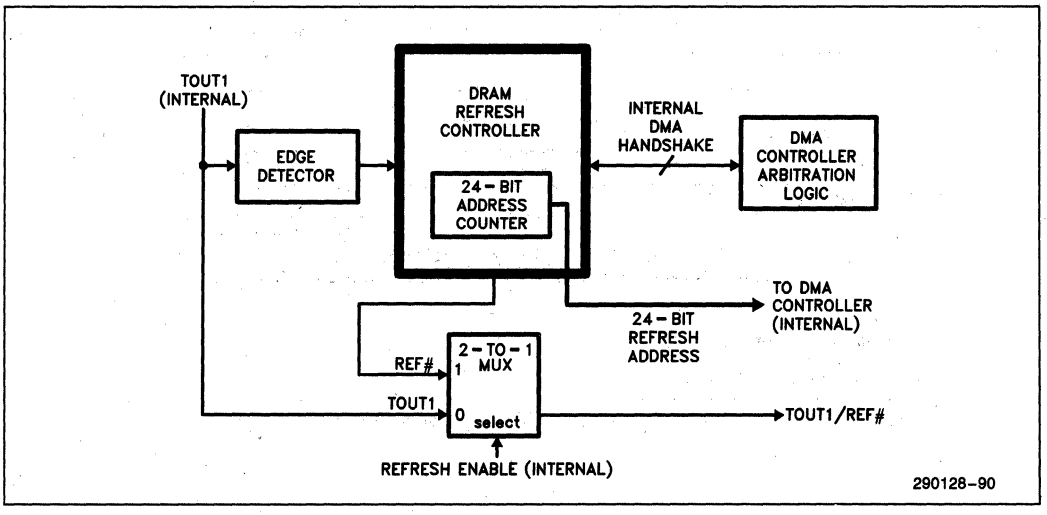


Figure 7-1. DRAM Refresh Controller

7.3 Bus Function

7.3.1 ARBITRATION

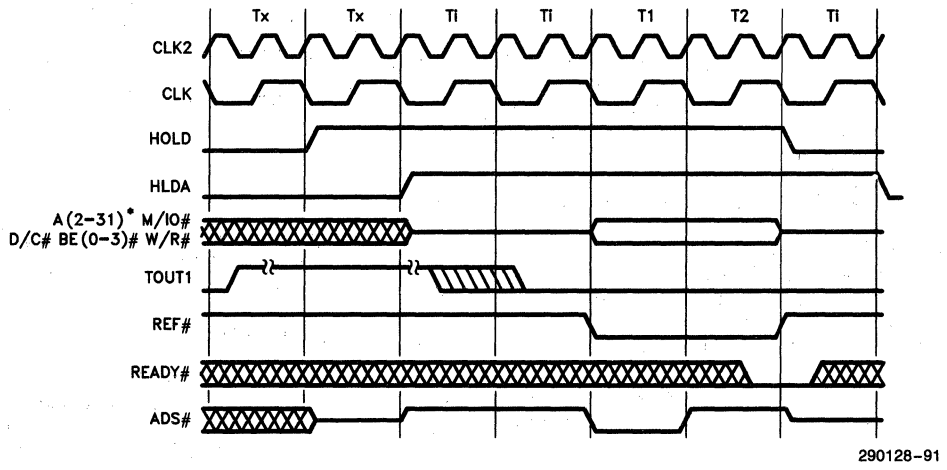
In order to ensure data integrity of the DRAMs, the 82380 gives the DRAM Refresh signal the highest priority in the arbitration logic. It allows DRAM Refresh to interrupt a DMA in progress in order to perform the DRAM Refresh cycle. The DMA service will be resumed after the refresh is done.

In case of a DRAM Refresh during a DMA process, the cascaded device will be requested to get off the bus. This is done by deasserting the EDACK signal. Once DREQn goes inactive, the 82380 will perform the refresh operation. Note that the DMA controller does not completely relinquish the system bus during refresh. The Refresh Generator simply 'steals' a bus cycle between DMA accesses.

Figure 7-2 shows the timing diagram of a Refresh Cycle. Upon expiration of TIMER 1, the 82380 will try to take control of the system bus by asserting HOLD. As soon as the 82380 see HLDA go active, the DRAM Refresh Cycle will be carried out by activating the REF# signal as well as the refresh address and control signals on the system bus (Note that REF# will not be active until two CLK periods after HLDA is asserted). The address bus will contain the 24-bit address currently in the Refresh Address Counter. The control signals are driven the same way as in a Memory Read cycle. This 'read' operation is complete when the READY# signal is driven LOW. Then, the 82380 will relinquish the bus by de-asserting HOLD. Typically, a Refresh Cycle without wait states will take five bus states to execute. If 'n' wait states are added, the Refresh Cycle will last for five plus 'n' bus states.

How often the Refresh Generation will initiate a refresh cycle depends on the frequency of CLKIN as well as TIMER1's programmed mode of operation. For this specific application, TIMER1 should be programmed to operate in Mode 2 or 3 to generate a constant clock rate. See section 6—Programmable Interval Timer for more information on programming the timer. One DRAM Refresh Cycle will be generated each time TIMER 1 expires (when TOUT1 changes to LOW to HIGH).

The Wait State Generator can be used to insert wait states during a refresh cycle. The 82380 will automatically insert the desired number of wait states as programmed in the Refresh Wait State Register (see Wait State Generator).



*NOTE:
A24–A31 = 1 during Refresh cycle.

Figure 7-2. 82380 Refresh Cycle

7.4 Modes of Operation

7.4.1 WORD SIZE AND REFRESH ADDRESS COUNTER

The 82380 supports 8-, 16- and 32-bit refresh cycle. The bus width during a refresh cycle is programmable (see Programming). The bus size can be programmed via the Refresh Control Register (see Register Overview). If the DRAM bus size is 8-, 16-, or 32-bits, the Refresh Address Counter will be incremented by 1, 2, or 4, respectively.

The Refresh Address Counter is cleared by a hardware reset.

7.5 Register Set Overview

The Refresh Generator has two internal registers to control its operation. They are the Refresh Control Register and the Refresh Wait State Register. Their port address map is shown in Table 7-1 below.

Port Address	Description
1CH	Refresh Control Reg. (read/write)
75H	Ref. Wait State Reg. (read/write)

Table 7-1. Register Address Map

The Refresh Wait State Register is not part of the Refresh Generator. It is only used to program the number of wait states to be inserted during a refresh cycle. This register is discussed in detail in section 7 (Wait State Generator) and will not be repeated here.

REFRESH CONTROL REGISTER

This 2-bit register serves two functions. First, it is used to enable/disable the DRAM Refresh function output. If disabled, the output of TIMER 1 is simply used as a general purpose timer. The second function of this register is to program the DRAM bus size for the refresh operation. The programmed bus size also determines how the Refresh Address Counter will be incremented after each refresh operation.

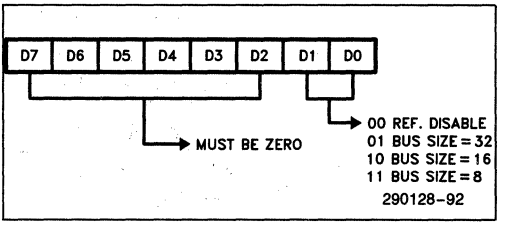
7.6 Programming

Upon hardware reset, the DRAM Refresh function is disabled (the Refresh Control Register is cleared). The following programming steps are needed before the Refresh Generator can be used. Since the rate of refresh cycles depends on how TIMER 1 is programmed, this timer must be initialized with the desired mode of operation as well as the correct refresh interval (see Programming Interval Timer). Whether or not wait states are to be generated during a refresh cycle, the Refresh Wait State Register must also be programmed with the appropriate value. Then, the DRAM Refresh feature must be enabled and the DRAM bus width should be defined. These can be done in one step by writing the appropriate control word into the Refresh Control Register (see Register Bit Definition). After these steps are done, the refresh operation will automatically be invoked by the Refresh Generator upon expiration of Timer 1.

In addition to the above programming steps, it should be noted that after reset, although the TOUT1/REF# becomes the Timer 1 output, the state of this pin is undefined. This is because the Timer module has not been initialized yet. Therefore, if this output is used as a DRAM Refresh signal, this pin should be disqualified by external logic until the Refresh function is enabled. One simple solution is to logically AND this output with HLDA, since HLDA should not be active after reset.

7.7 Register Bit Definition

REFRESH CONTROL REGISTER
Port Address: 1CH (Read/Write)



8.0 RELOCATION REGISTER AND ADDRESS DECODE

8.1 Relocation Register

All the integrated peripheral devices in the 82380 are controlled by a set of internal registers. These registers span a total of 256 consecutive address locations (although not all the 256 locations are used). The 82380 provides a Relocation Register which allows the user to map this set of internal registers into either the memory or I/O address space. The function of the Relocation Register is to define the base address of the internal register set of the 82380 as well as if the registers are to be memory- or I/O-mapped. The format of the Relocation Register is depicted in Figure 8-1.

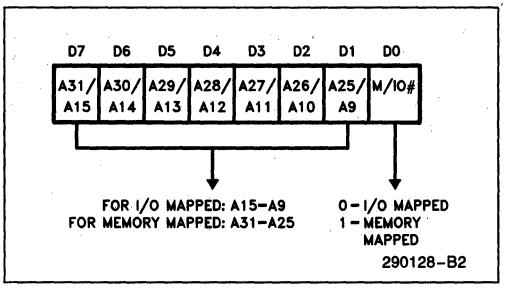


Figure 8-1. Relocation Register

Note that the Relocation Register is part of the internal register set of the 82380. It has a port address of 7FH. Therefore, any time the content of the Relocation Register is changed, the physical location of this register will also be moved. Upon reset of the 82380, the content of the Relocation Register will be cleared. This implies that the 82380 will respond to its I/O addresses in the range of 0000H to 00FFH.

8.1.1 I/O-MAPPED 82380

As shown in the figure, Bit 0 of the Relocation Register determines whether the 82380 registers are to be memory-mapped or I/O-mapped. When Bit 0 is set to '0', the 82380 will respond to I/O Addresses. Address signals BE0#–BE3#, A2–A7 will be used to select one of the internal registers to be accessed. Bit 1 to Bit 7 of the Relocation Register will correspond to A9 to A15 of the Address bus, respectively. Together with A8 implied to be '0', A15 to A8 will be fully decoded by the 82380. The following shows how the 82380 is mapped into the I/O address space.

Example

Relocation Register = 11001110 (0CEH)

82380 will respond to I/O address range from 0CE00H to 0CEFFH.

Therefore, this I/O mapping mechanism allows the 82380 internal registers to be located on any even, contiguous, 256 byte boundary of the system I/O space.

Port Address: 7FH (Read/Write)

8.1.2 MEMORY-MAPPED 82380

When Bit 0 of the Relocation Register is set to '1', the 82380 will respond to memory addresses. Again, Address signals BE0#–BE3#, A2–A7 will be used to select one of the internal registers to be accessed. Bit 1 to Bit 7 of the Relocation Register will correspond to A25–A31, respectively. A24 is assumed to be '0', and A8–A23 are ignored. Consider the following example.

Example

Relocation Register = 10100111 (0A7H)

The 82380 will respond to memory addresses in the range of 0A6XXXX00H to 0A6XXXXFFH (where 'X' is don't care).

This scheme implies that the internal register can be located in any even, contiguous, 2**24 byte page of the memory space.

8.2 Address Decoding

As mentioned previously, the 82380 internal registers do not occupy the entire contiguous 256 address locations. Some of the locations are 'unoccupied'. The 82380 always decodes the lower 8 address bits (A0–A7) to determine if any one of its registers is being accessed. If the address does not correspond to any of its registers, the 82380 will not respond. This allows external devices to be located within the 'holes' in the 82380 address space. Note that there are several unused addresses reserved for future Intel peripheral devices.

9.0 CPU RESET AND SHUTDOWN DETECT

The 82380 will activate the CPURST signal to reset the host processor when one of the following conditions occurs:

— 82380 RESET is active;

— 82380 detects a 80386 Shutdown cycle (this feature can be disabled);

— CPURST software command is issued to 80386.

Whenever the CPURST signal is activated, the 82380 will reset its own internal Slave-Bus state machine.

9.1 Hardware Reset

Following a hardware reset, the 82380 will assert its CPURST output to reset the host processor. This output will stay active for as long as the RESET input is active. During a hardware reset, the 82380 internal registers will be initialized as defined in the corresponding functional descriptions.

9.2 Software Reset

CPURST can be generated by writing the following bit pattern into 82380 register location 64H.

D7							D0
1	1	1	1	X	X	X	0

X = Don't Care

The Write operation into this port is considered as an 82380 access and the internal Wait State Generator will automatically determine the required number of wait states. The CPURST will be active following the completion of the Write cycle to this port. This signal will last for 62 CLK2 periods. The 82380 should not be accessed until the CPURST is deactivated.

This internal port is Write-Only and the 82380 will not respond to a Read operation to this location. Also, during a CPU software reset command, the 82380 will reset its Slave-Bus state machine. However, its internal registers remain unchanged. This allows the operating system to distinguish a 'warm' reset by reading any 82380 internal register previously programmed for an non-default value. The Diagnostic registers can be used or this purpose (see Internal Control and Diagnostic Ports).

9.3 Shutdown Detect

The 82380 is constantly monitoring the Bus Cycle Definition signals (M/IO#, D/C#, R/W#) and is able to detect when the 80386 executes a Shutdown bus cycle. Upon detection of a processor shutdown, the 82380 will activate the CPURST output for 62 CLK2 periods to reset the host processor. This signal is generated after the Shutdown cycle is terminated by the READY# signal.

Although the 82380 Wait State Generator will not automatically respond to a Shutdown (or Halt) cycle, the Wait State Control inputs (WSC0, WSC1) can be used to determine the number of wait states in the same manner as other non-82380 bus cycle.

This Shutdown Detect feature can be enabled or disabled by writing a control bit in the Internal Control Port at address 61H (see Internal Control and Diagnostic Ports). This feature is disabled upon a hardware reset of the 82380. As in the case of Software Reset, the 82380 will reset its Slave-Bus state machine but will not change any of its internal register contents.

10.0 INTERNAL CONTROL AND DIAGNOSTIC PORTS

10.1 Internal Control Port

The format of the Internal Control Port of the 82380 is shown in Figure 10.1. This Control Port is used to enable/disable the Processor Shutdown Detect mechanism as well as controlling the Gate inputs of the Timer 2 and 3. Note that this is a Write-Only port. Therefore, the 82380 will not respond to a read operation to this port. Upon hardware reset, this port will be cleared; i.e., the Shutdown Detect feature and the Gate inputs of Timer 2 and 3 are disabled.

10.2 Diagnostic Ports

Two 8-bit read/write Diagnostic Ports are provided in the 82380. These are two storage registers and have no effect on the operation of the 82380. They can be used to store checkpoint data or error codes in the power-on sequence and in the diagnostic service routines. As mentioned in CPU RESET AND SHUTDOWN DETECT section, these Diagnostic Ports can be used to distinguish between 'cold' and 'warm' reset. Upon hardware reset, both Diagnostic Ports are cleared. The address map of these Diagnostic Ports is shown in Figure 10-2.

Port	Address
Diagnostic Port 1 (Read/Write)	80H
Diagnostic Port 2 (Read/Write)	88H

Figure 10-2. Address Map of Diagnostic Ports

Port Address: 61H (Write Only)

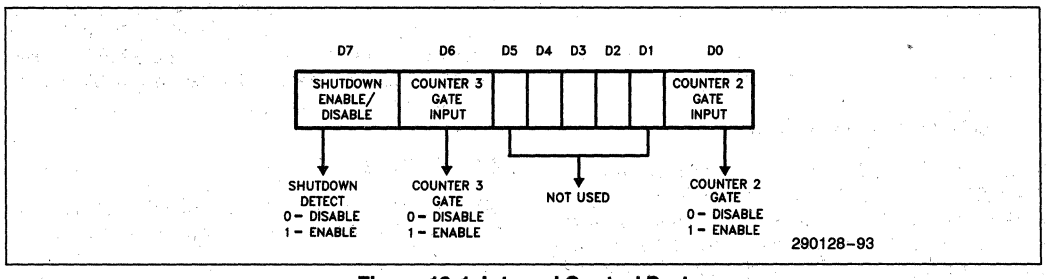


Figure 10-1. Internal Control Port

11.0 INTEL RESERVED I/O PORTS

There are eleven I/O ports in the 82380 address space which are reserved for Intel future peripheral device use only. Their address locations are: 2AH, 3DH, 3EH, 45H, 46H, 76H, 77H, 7DH, 7EH, CCH and CDH. These addresses should not be used in the system since the 82380 may respond to read/write operations to these locations and bus contention may occur if any peripheral is assigned to the same address location.

12.0 MECHANICAL DATA

12.1 Introduction

In this section, the physical package and its connections are described in detail.

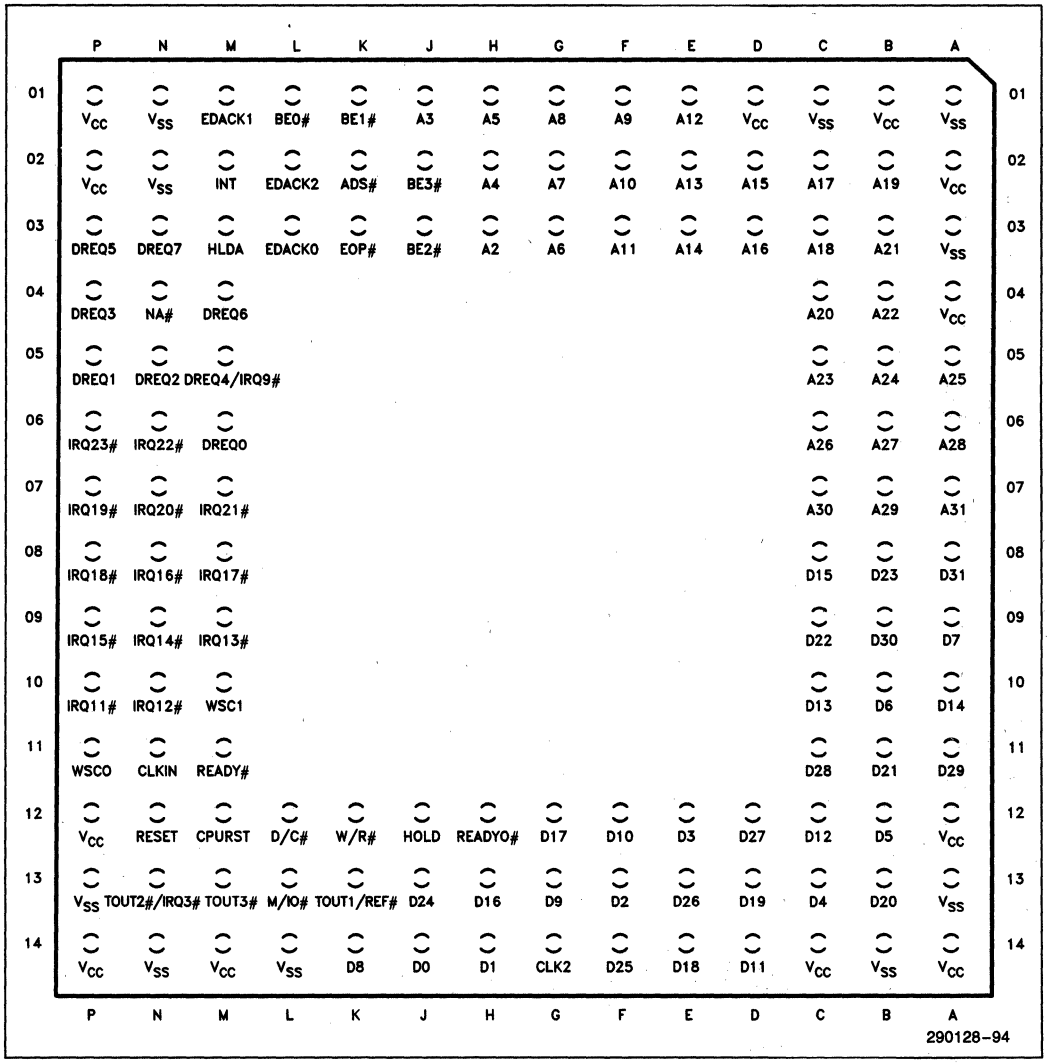


Figure 12.1. 82380 PGA Pinout—View from TOP side

12.2 Pin Assignment

The 82380 pinout as viewed from the top side of the component is shown in Figure 12.1. Its pinout as viewed from the pin side of the component is shown in Figure 12.2.

V_{CC} and GND connections must be made to multiple V_{CC} and V_{SS} (GND) pins. Each V_{CC} and V_{SS} MUST be connected to the appropriate voltage level. The circuit board should include V_{CC} and GND planes for power distribution and all V_{CC} pins must be connected to the appropriate plane.

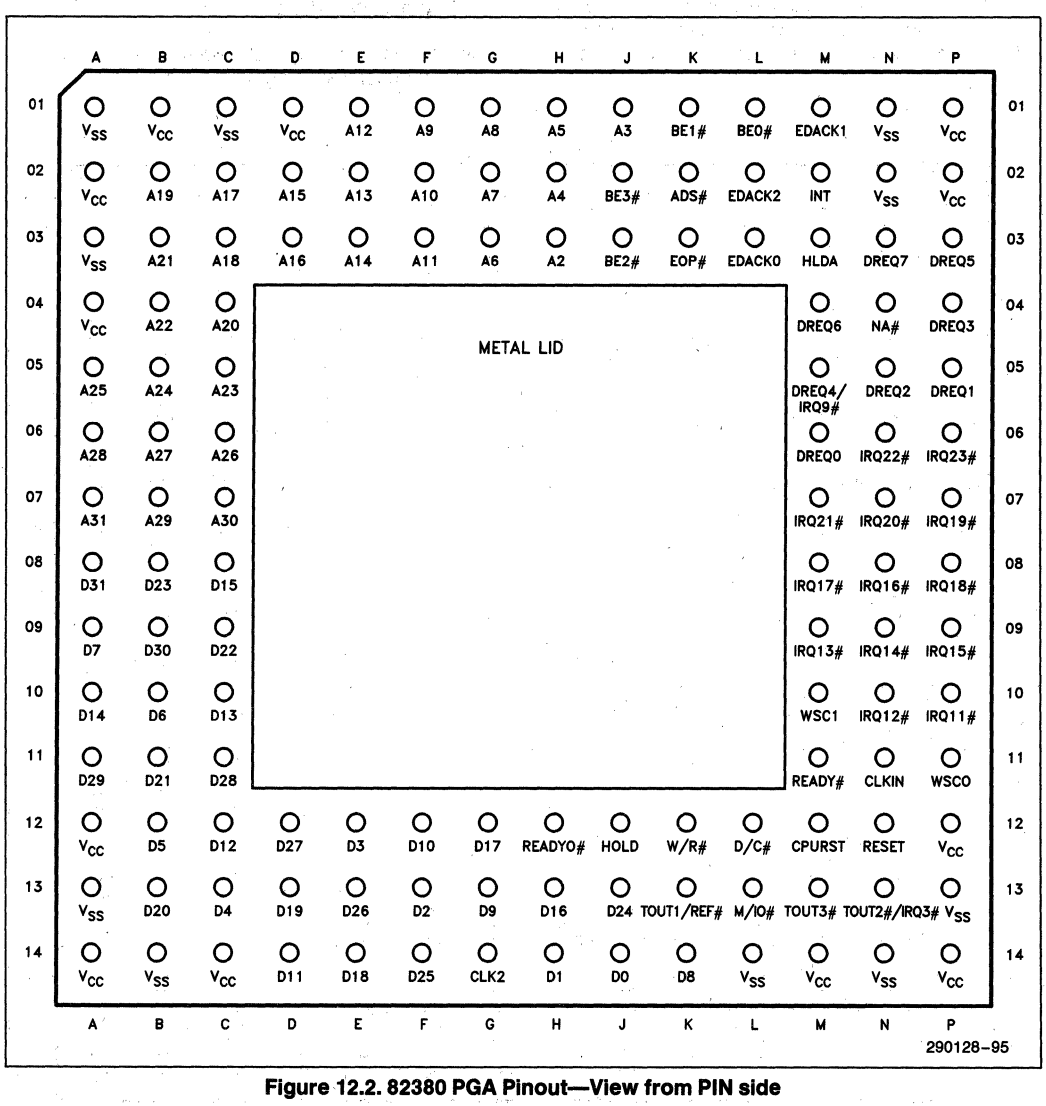


Figure 12.2. 82380 PGA Pinout—View from PIN side

Table 12-1. 82380 PGA Pinout—Functional Grouping

Pin/Signal		Pin/Signal		Pin/Signal		Pin/Signal	
A7	A31	A8	D31	P12	V_{CC}	L14	V_{SS}
C7	A30	B9	D30	M14	V_{CC}	A1	V_{SS}
B7	A29	A11	D29	P1	V_{CC}	P13	V_{SS}
A6	A28	C11	D28	P2	V_{CC}	N1	V_{SS}
B6	A27	D12	D27	P14	V_{CC}	N2	V_{SS}
C6	A26	E13	D26	D1	V_{CC}	C1	V_{SS}
A5	A25	F14	D25	C14	V_{CC}	A3	V_{SS}
B5	A24	J13	D24	B1	V_{CC}	B14	V_{SS}
C5	A23	B8	D23	A2	V_{CC}	A13	V_{SS}
B4	A22	C9	D22	A4	V_{CC}	N14	V_{SS}
B3	A21	B11	D21	A12	V_{CC}		
C4	A20	B13	D20	A14	V_{CC}	P6	IRQ23#
B2	A19	D13	D19			N6	IRQ22#
C3	A18	E14	D18	G14	CLK2	M7	IRQ21#
C2	A17	G12	D17	L12	D/C#	N7	IRQ20#
D3	A16	H13	D16	K12	W/R#	P7	IRQ19#
D2	A15	C8	D15	L13	M/IO#	P8	IRQ18#
E3	A14	A10	D14	K2	ADS#	M8	IRQ17#
E2	A13	C10	D13	N4	NA#	N8	IRQ16#
E1	A12	C12	D12	J12	HOLD	P9	IRQ15#
F3	A11	D14	D11	M3	HLDA	N9	IRQ14#
F2	A10	F12	D10	M6	DREQ0	M9	IRQ13#
F1	A9	G13	D9	P5	DREQ1	N10	IRQ12#
G1	A8	K14	D8	N5	DREQ2	P10	IRQ11#
G2	A7	A9	D7	P4	DREQ3	M2	INT
G3	A6	B10	D6	M5	DREQ4/IRQ9#		
H1	A5	B12	D5	P3	DREQ5	N11	CLKIN
H2	A4	C13	D4	M4	DREQ6	K13	TOUT1/REF#
J1	A3	E12	D3	N3	DREQ7	N13	TOUT2#/IRQ3#
H3	A2	F13	D2			M13	TOUT3#
J2	BE3#	H14	D1	K3	EOP#	M11	READY#
J3	BE2#	J14	D0	L3	EDACK0	H12	READYO#
K1	BE1#			M1	EDACK1	P11	WSC0
L1	BE0#	N12	RESET	L2	EDACK2	M10	WSC1
		M12	CPURST				

12.3 Package Dimensions and Mounting

The 82380 package is a 132-pin ceramic Pin Grid Array (PGA). The pins are arranged 0.100 inch (2.54 mm) center-to-center, in a 14 x 14 matrix, three rows around.

A wide variety of available sockets allow low insertion force or zero insertion force mountings, and a choice of terminals such as soldertail, surface mount, or wire wrap. Several applicable sockets are listed in Figure 12-4.

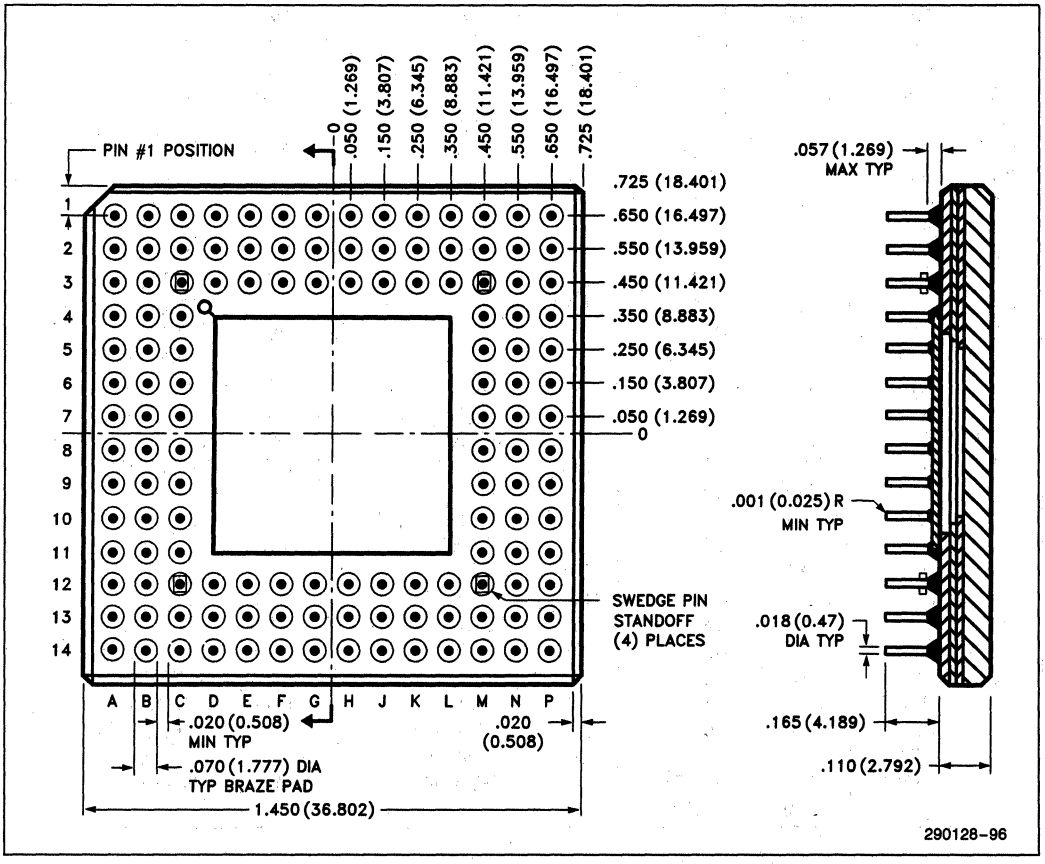


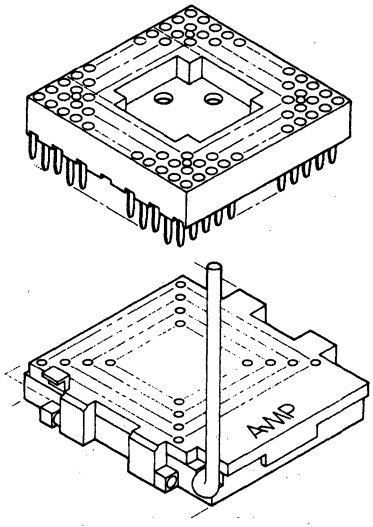
Figure 12.3. 132-Pin Ceramic PGA Package Dimensions

- Low insertion force (LIF) soldertail 55274-1
- Amp tests indicate 50% reduction in insertion force compared to machined sockets

Other socket options

- Zero insertion force (ZIF) soldertail 55583-1
- Zero insertion force (ZIF) Burn-in version 55573-2

Amp Incorporated
(Harrisburg, PA 17105 U.S.A.
Phone 717-564-0100)



290128-97

Cam handle locks in low profile position when substrate is installed (handle UP for open and DOWN for closed positions)

courtesy Amp Incorporated

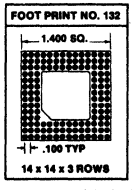
Peel-A-Way™ Mylar and Kapton Socket Terminal Carriers

- Low insertion force surface mount CS132-37TG
- Low insertion force soldertail CS132-01TG
- Low insertion force wire-wrap CS132-02TG (two level) CS132-03TG (three-level)
- Low insertion force press-fit CS132-05TG

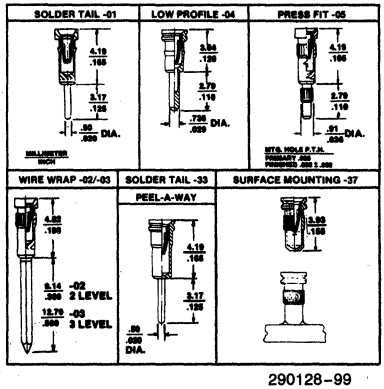
Advanced Interconnections
(5 Division Street
Warwick, RI 02818 U.S.A.
Phone 401-885-0485)

Peel-A-Way Carrier No. 132;
Kapton Carrier is KS132
Mylar Carrier is MS132

Molded Plastic Body KS132 is shown below:



290128-98



290128-99

courtesy Advanced Interconnections
(Peel-A-Way Terminal Carriers
U.S. Patent No. 4442938)

Figure 12-4. Several Socket Options for 132-pin PGA

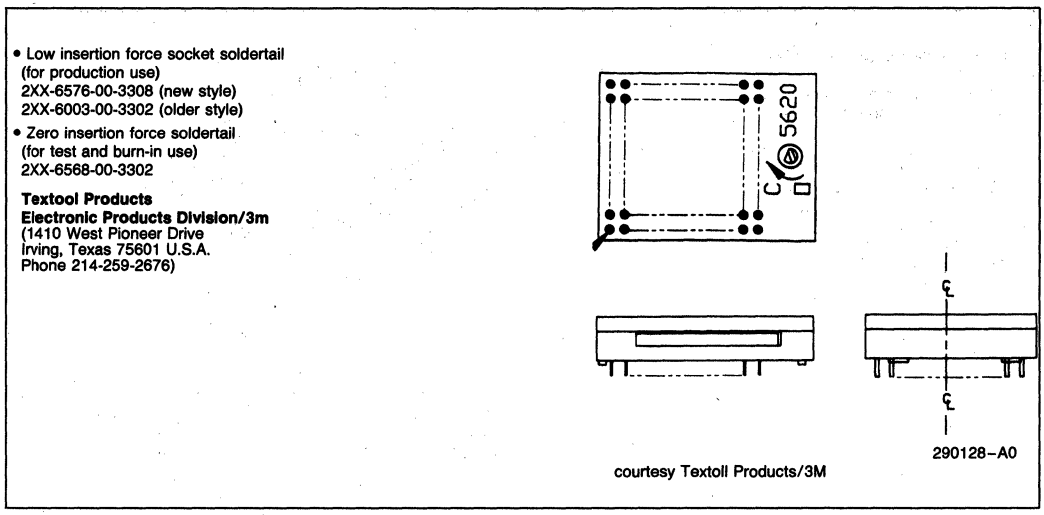


Figure 12-4. Several Socket Options for 132-pin PGA (Continued)

12.4 Package Thermal Specification

The 82380 is specified for operation when case temperature is within the range of 0°C −85°C. The case temperature may be measured in any environment, to determine whether the 82380 is within the specified operating range.

The PGA case temperature should be measured at the center of the top surface opposite the pins, as in Figure 12.5.

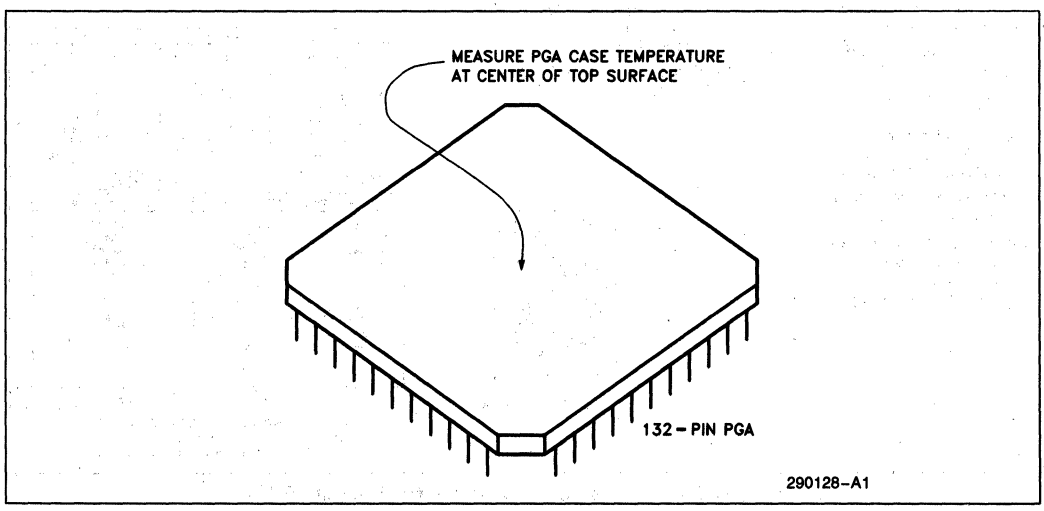
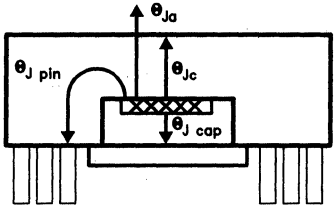


Figure 12.5. Measuring 82380 PGA Case Temperature

Thermal Resistance—°C/Watt							
Parameter	Airflow—ft³/min (m³/sec)						
	0 (0)	50 (0.25)	100 (0.50)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)
θ Junction-to-Case (case measured as Fig. 6.4)	2	2	2	2	2	2	2
θ Case-to-Ambient (no heatsink)	19	18	17	15	12	10	9
θ Case-to-Ambient (with omnidirectional heatsink)	16	15	14	12	9	7	6
θ Case-to-Ambient (with unidirectional heatsink)	15	14	13	11	8	6	5



290128-A2

NOTES:
1. Table 12-6 applies to 82380 PGA plugged into socket or soldered directly into board.
2. $\theta_{JA} = \theta_{JC} + \theta_{CA}$.
3. $\theta_{J\text{-}CAP} = 4°C/W$ (approx.)
   $\theta_{J\text{-}PIN} = 4°C/W$ (inner pins) (approx.)
   $\theta_{J\text{-}PIN} = 8°C/W$ (outer pins) (approx.)

Figure 12-6. 82380 PGA Package Typical Thermal Characteristics

13.0 ELECTRICAL DATA

13.1 Power and Grounding

The large number of output buffers (address, data and control) can cause power surges as multiple output buffers drive new signal levels simultaneously. The 22 V_{CC} and V_{SS} pins of the 82380 each feed separate functional units to minimize switching induced noise effects. All V_{CC} pins of the 82380 must be connected on the circuit board.

13.2 Power Decoupling

Liberal decoupling capacitance should be placed close to the 82380. The 82380 driving its 32-bit parallel address and data buses at high frequencies can cause transient power surges when driving large capacitive loads. Low inductance capacitors and interconnects are recommended for the best reliability at high frequencies. Low inductance capacitors are available specifically for Pin Grid Array packages.

13.3 Unused Pin Recommendations

For reliable operation, ALWAYS connect unused inputs to a valid logic level. As is the case with most other CMOS processes, a floating input will increase the current consumption of the component and give an indeterminate state to the component.

13.4 ICE-386 Support

The 82380 specifications provide sufficient drive capability to support the ICE386. On the pins that are generally shared between the 80386 and the 82380, the additional loading represented by the ICE386 was allowed for in the design of the 82380.

13.5 Maximum Ratings

Storage Temperature ..... −65°C to +150°C
Case temperature Under Bias ... −65°C to +110°C
Supply Voltage with Respect to V_{SS} ..... −0.5V to +6.5V
Voltage on any other Pin ..... −0.5V to V_{CC} +0.5V

NOTE:
Stress above those listed above may cause permanent damage to the device. This is a stress rating only and functional operation at these or any other conditions above those listed in the operational sections of this specification is not implied.

Exposure to absolute maximum rating conditions for extended periods may affect device reliability. Although the 82380 contains protective circuitry to reset damage from static electric discharges, always take precautions against high static voltages or electric fields.

13.6 D.C. Specifications

T_{CASE} = 0°C to 85°C; V_{CC} = 5V ±5%; V_{SS} = 0V.

Table 13-1.

Symbol	Parameter	Min	Max	Unit	Notes
V_{IL}	Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{ILC}	CLK2 Input Low Voltage	−0.3	0.8		(Note 1)
V_{IHC}	CLK2 Input High Voltage	V_{CC} − 0.8	V_{CC} + 0.3	V	
V_{OL}	Output Low Voltage I_{OL} = 4 mA: A2–A31, D0–D31 I_{OL} = 5 mA: All Others		 0.45 0.45	 V V	
V_{OH}	Output High Voltage I_{OH} = −1 mA: A2–A31, D0–D31 I_{OH} = −0.9 mA: All Others	 2.4 2.4		 V V	
I_{LI}	Input Leakage Current for all ins except: IRQ11#–IRQ23#, TOUT2/IRQ3#, EOP#, DREQ4		±15	μA	$0V < V_{IN} < V_{CC}$
I_{LI1}	Input Leakage Current for pins: IRQ11#–IRQ23#, TOUT2#/IRQ3#, EOP#, DREQ4	10	−300	μA	$0V < V_{IN} < V_{CC}$ (Note 3)
I_{LO}	Output Leakage Current		±15	μA	$0.45 < V_{OUT} < V_{CC}$
I_{CC}	Supply Current		300 325	mA mA	CLK2 = 32 MHz = 40 MHz (Note 4)
(CAP)	Capacitance (Input/IO)		12	pF	f_c = 1 MHz (Note 2)
CCLK	CLK2 Capacitance		20	pF	f_c = 1 MHz (Note 2)

NOTES:
1. Minimum value is not 100% tested.
2. Sampled only.
3. These pins have internal pullups on them.
4. I_{CC} is specified with inputs driven to CMOS levels. I_{CC} may be higher if driven to TTL levels.

13.6 D.C. Specifications (Continued)

T_{CASE} = 0°C to 85°C; V_{CC} = 5V ±5%; V_{SS} = 0V.

Table 13-2. 82380-25 D.C. Specifications

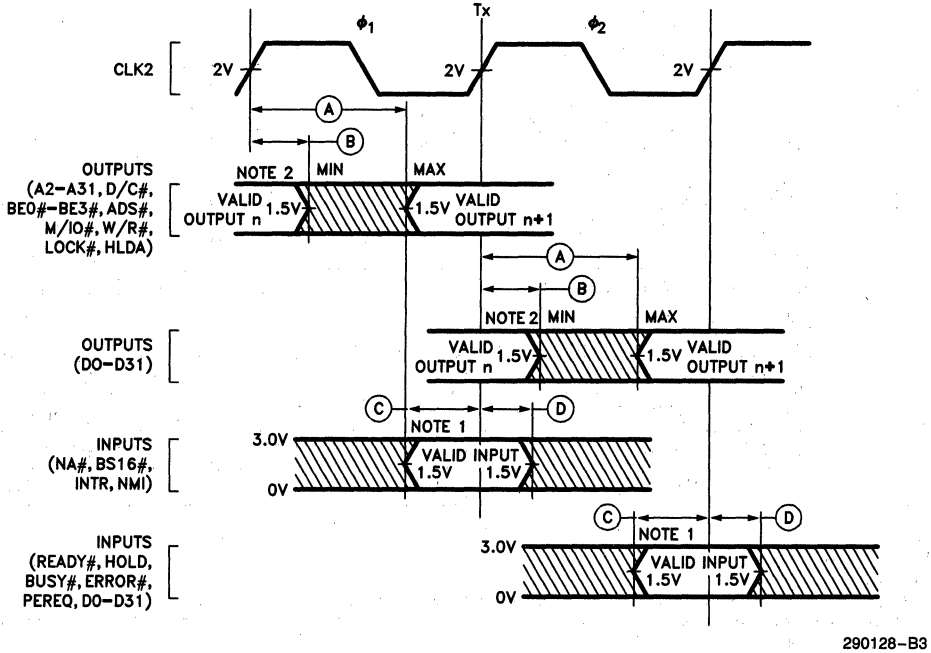
Symbol	Parameter	Min	Max	Unit	Notes
V_{IL}	Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{ILC}	CLK2 Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IHC}	CLK2 Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{OL}	Output Low Voltage I_{OL} = 4 mA: A_2–A_{31}, D_0–D_{31} I_{OL} = 5 mA: All Others		 0.45 0.45	 V V	
V_{OH}	Output High Voltage I_{OH} = −1 mA: A_2–A_{31}, D_0–D_{31} I_{OH} = −0.9 mA: All Others	 2.4 2.4		 V V	
I_{LI}	Input Leakage Current All Inputs except: IRQ11# – IRQ23#, EOP#, TOUT2/IRQ3#, DREQ4		±15	µA	
I_{LI1}	Input Leakage Current Inputs: IRQ11# – IRQ23#, EOP#, TOUT2/IRQ3#, DREQ4	10	−300	µA	$0 < V_{IN} < V_{CC}$ (Note 3)
I_{LO}	Output Leakage Current		±15	µA	$0 < V_{IN} < V_{CC}$
I_{CC}	Supply Current (CLK2 = 50 MHz)		375	mA	(Note 4)
C_I	Input Capacitance		12	pF	(Note 2)
C_{CLK}	CLK2 Input Capacitance		20	pF	(Note 2)

NOTES:
1. Minimum value is not 100% tested.
2. f_C = 1 MHz; Sampled only.
3. These pins have weak internal pullups. They should not be left floating.
4. I_{CC} is specified with inputs driven to CMOS levels, and outputs driving CMOS loads. I_{CC} may be higher if inputs are driven to TTL levels, or if outputs are driving TTL loads.

13.7 A.C. Specifications

The A.C. specifications given in the following tables consist of output delays and input setup requirements. The A.C. diagram's purpose is to illustrate the clock edges from which the timing parameters are measured. The reader should not infer any other timing relationships from them. For specific information on timing relationships between signals, refer to the appropriate functional section.

A.C. spec measurement is defined in Figure 13.1. Inputs must be driven to the levels shown when A.C. specifications are measured. 82380 output delays are specified with minimum and maximum limits, which are measured as shown. The minimum 82380 output delay times are hold times for external circuitry. 82380 input setup and hold times are specified as minimums and define the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct 82380 operation.



290128-B3

LEGEND:

Ⓐ—maximum output delay spec
Ⓑ—minimum output delay spec
Ⓒ—minimum input setup spec
Ⓓ—minimum input hold spec

NOTES:

1. Input waveforms have tr ≤ 2.0 ns from 0.8V to 2.0V.
2. Under rated loading (120 pF) 386 output tr, tf is typically ≤ 4.0 ns from 0.8V to 2.0V.

Figure 13-1. Drive Levels and Measurement Points for A.C. Specification

A.C. SPECIFICATION TABLES

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C

Table 13-3. 82380 A.C. Characteristics

Symbol	Parameter	82380-16		82380-20		Notes
		Min	Max	Min	Max	
	Operating Frequency	4 MHz	16 MHz	4 MHz	20 MHz	Half CLK2 Frequency
t1	CLK2 Period	31 ns	125 ns	25 ns	125 ns	
t2a	CLK2 High Time	9		8		at 2.0V
t2b	CLK2 High Time	5		5		at (V_{CC}–0.8)V
t3a	CLK2 Low Time	9		8		at 2.0V
t3b	CLK2 Low Time	7		6		at 0.8V
t4	CLK2 Fall Time		8		8	(V_{CC}–0.8)V to 0.8V
t5	CLK2 Rise Time		8		8	0.8V to (V_{CC}–0.8)V
	A (2–31), BE (0–3) #, EDACK (0–2)					
t6	Valid Delay	4	36	4	30	CL = 120 pF
t7	Float Delay	4	40	4	32	(Note 1)
	A (2–31), BE (0–3) #					
t8	Setup Time	6		6		
t9	Hold Time	4		4		
	W/R#, M/IO#, D/C#,					
t10	Valid Delay	6	33	6	28	CL = 75 pF
t11	Float Delay	4	35	4	30	(Note 1)
t12	Setup Time	6		6		
t13	Hold Time	4		4		
t14	ADS# Valid Delay	6	33	6	28	CL = 75 pF
t15	Float Delay	4	35	4	30	
t16	Setup Time	21		15		
t17	Hold Time	4		4		
	Slave Mode— D(0–31) Read					
t18	Valid Delay	3	46	4	46	CL = 120 pF
t19	Float Delay	6	35	6	29	(Note 1)
	Slave Mode— D(0–31) Write					
t20	Setup Time	31		29		
t21	Hold Time	26		26		

A.C. SPECIFICATION TABLES (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C.

Table 13-3. 82380 A.C. Characteristics (Continued)

Symbol	Parameter	82380-16		82380-20		Notes
		Min	Max	Min	Max	
	Master Mode— D(0–31) Write					
t22	Valid Delay	4	48	4	38	CL = 120 pF
t23	Float Delay	4	35	4	27	(Note 1)
	Master Mode— D(0–31) Read					
t24	Setup Time	11		11		
t25	Hold Time	6		6		
t26	READY# Setup Time	21		12		
t27	Hold Time	4		4		
t28	WSC (0–1) Setup	6		6		
t29	Hold	21		21		
t31	RESET Setup Time	13		12		
t30	Hold Time	4		4		
t32	READYO# Valid Delay	4	31	4	28	CL = 25 pF
t33	CPU Reset From CLK2	2	18	2	16	CL = 50 pF
t34	HOLD Valid Delay	5	33	5	30	CL = 100 pF
t35	HLDA Setup Time	21		17		
t36	Hold Time	6		6		
t37a	EOP# Setup Time	21		17		Synch. EOP
t38a	EOP# Hold Time	4		4		
t37b	EOP# Setup Time	11		11		Asynch. EOP
t38b	EOP# Hold Time	11		11		
t39	EOP# Valid Delay	5	38	5	30	CL = 100 pF ('1'->'0')
t40	EOP# Float Delay	5	40	5	32	
t41a	DREQ Setup Time	21		19		Synchronous DREQ
t42a	Hold Time	4		4		
t41b	DREQ Setup Time	11		11		Asynchronous DREQ
t42b	Hold Time	11		11		
t43	INT Valid Delay		500		500	From IRQ Input CL = 75 pF
t44	NA# Setup Time	11		10		
t45	Hold Time	15		15		

A.C. SPECIFICATION TABLES (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C.

Table 13-3. 82380 A.C. Characteristics (Continued)

Symbol	Parameter	82380-16		82380-20		Notes
		Min	Max	Min	Max	
t46	CLKIN Frequency	0 MHz	10 MHz	0 MHz	10 MHz	
t47	CLKIN High Time	30		30		At 1.5V
t48	CLKIN Low Time	50		50		At 1.5V
t49	CLKIN Rise Time		10		10	0.8V to 2.0V
t50	CLKIN Fall Time		10		10	2.0V to 0.8V
t51	TOUT1/REF# Valid	4	36	4	30	From CLK2, CL = 25 pF
t52	TOUT1/REF# Valid	3	93	3	93	From CLKIN, CL = 120 pF
t53	TOUT2# Valid Delay	3	93	3	93	From CLKIN, CL = 120 pF (Falling Edge Only)
t54	TOUT2# Float Delay	3	40	3	40	From CLKIN (Note 1)
t55	TOUT3# Valid Delay	3	93	3	93	From CLKIN, CL = 120 pF

NOTE:
1. Float condition occurs when the maximum output current becomes less than ILO in magnitude. Float delay is not tested. For testing purposes, the float condition occurs when the dynamic output driven voltage changes with current loads.

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C.
A.C. timings are tested at 1.5V thresholds; except as noted.

Table 13-4. 82380-25 A.C. Characteristics

Symbol	Parameter	82380-25		Unit	Notes
		Min	Max		
	Operating Frequency 1/(t1a × 2)	4	25	MHz	
t1	CLK2 Period	20	125	ns	
t2a	CLK2 High Time	7		ns	at 2.0V
t2b	CLK2 High Time	4		ns	at 3.7V
t3a	CLK2 Low Time	7		ns	at 2.0V
t3b	CLK2 Low Time	4		ns	at 0.8V
t4	CLK2 Fall Time		7	ns	3.7V to 0.8V
t5	CLK2 Rise Time		7	ns	0.8V to 3.7V
t6	A2–A31, BE0#–BE3# EDACK0–EDACK3 Valid Delay	4	20	ns	50 pF Load
t7	A2–A31, BE0#–BE3# EDACK0–EDACK3 Float Delay	4	27	ns	50 pF Load
t8	A2–A31, BE0#–BE3# Setup Time	6		ns	
t9	A2–A31, BE0#–BE3# Hold Time	4		ns	
t10	W/R#, M/IO#, D/C# Valid Delay	4	20	ns	50 pF Load
t11	W/R#, M/IO#, D/C# Float Delay	4	29	ns	50 pF Load

A.C. SPECIFICATION TABLES (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C.
A.C. timings are tested at 1.5V thresholds; except as noted.

Table 13-4. 82380-25 A.C. Characteristics (Continued)

Symbol	Parameter	82380-25		Unit	Notes
		Min	Max		
t12	W/R#, M/IO#, D/C# Setup Time	6		ns	
t13	W/R#, M/IO#, D/C# Hold Time	4		ns	
t14	ADS# Valid Delay	4	19	ns	50 pF Load
t15	ADS# Float Delay	4	29	ns	50 pF Load
t16	ADS# Setup Time	12		ns	
t17	ADS# Hold Time	4		ns	
t18	Slave Mode D0–D31 Read Valid	4	31	ns	50 pF Load
t19	Slave Mode D0–D31 Read Float	6	21	ns	50 pF Load
t20	Slave Mode D0–D31 Write Setup	20		ns	
t21	Slave Mode D0–D31 Write Hold	20		ns	
t22	Master Mode D0–D31 Write Valid	8	27	ns	50 pF Load
t23	Master Mode D0–D31 Write Float	4	19	ns	50 pF Load
t24	Master Mode D0–D31 Read Setup	7		ns	
t25	Master Mode D0–D31 Read Hold	4		ns	
t26	READY# Setup Time	9		ns	
t27	READY# Hold Time	4		ns	
t28	WSC0–WSC1 Setup Time	6		ns	
t29	WSC0–WSC1 Hold Time	15		ns	
t30	RESET Hold Time	4		ns	
t31	RESET Setup Time	9		ns	
t32	READYO# Valid Delay	3	21	ns	25 pF Load
t33	CPURST Valid Delay	2	14	ns	50 pF Load
t34	HOLD Valid Delay	4	22	ns	50 pF Load
t35	HLDA Setup Time	17		ns	
t36	HLDA Hold Time	4		ns	
t37a	EOP# Setup (Synchronous)	13		ns	
t38a	EOP# Hold (Synchronous)	4		ns	
t37b	EOP# Setup (Asynchronous)	10		ns	
t38b	EOP# Hold (Asynchronous)	10		ns	
t39	EOP# Valid Delay	4	21	ns	50 pF Load
t40	EOP# Float Delay	4	21	ns	50 pF Load
t41a	DREQ Setup (Synchronous)	17		ns	
t42a	DREQ Hold (Synchronous)	4		ns	
t41b	DREQ Setup (Asynchronous)	10		ns	
t42b	DREQ Hold (Asynchronous)	10		ns	
t43	INT Valid Delay from IRQn		500	ns	50 pF Load

A.C. SPECIFICATION TABLES (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C.
A.C. timings are tested at 1.5V thresholds; except as noted.

Table 13-4. 82380-25 A.C. Characteristics (Continued)

Symbol	Parameter	82380-25		Unit	Notes
		Min	Max		
t44	NA# Setup Time	7		ns	
t45	NA# Hold Time	8		ns	
t46	CLKIN Frequency	0	10	MHz	
t47	CLKIN High Time	30		ns	2.0V
t48	CLKIN Low Time	50		ns	0.8V
t49	CLKIN Rise Time		10	ns	0.8V to 3.7V
t50	CLKIN Fall Time		10	ns	3.7V to 0.8V
t51	TOUT1/REF# Valid Delay from CLK2 (Refresh)	4	20	ns	50 pF Load
t52	TOUT1/REF# Valid Delay from CLKIN (Timer)	3	90	ns	50 pF Load
t53	TOUT2# Valid Delay (Falling Edge Only)	3	90	ns	50 pF Load
t54	TOUT2# Float Delay	3	37	ns	50 pF Load
t55	TOUT3# Valid Delay	3	90	ns	50 pF Load

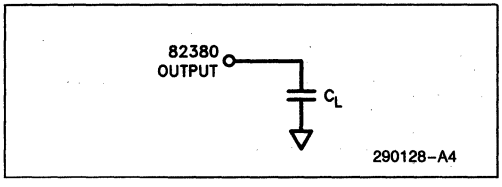


Figure 13-2. A.C. Test Load

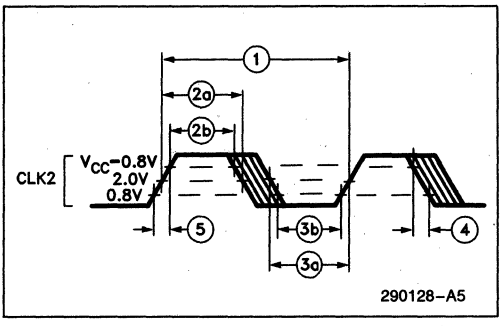


Figure 13-3. CLK2 Timing

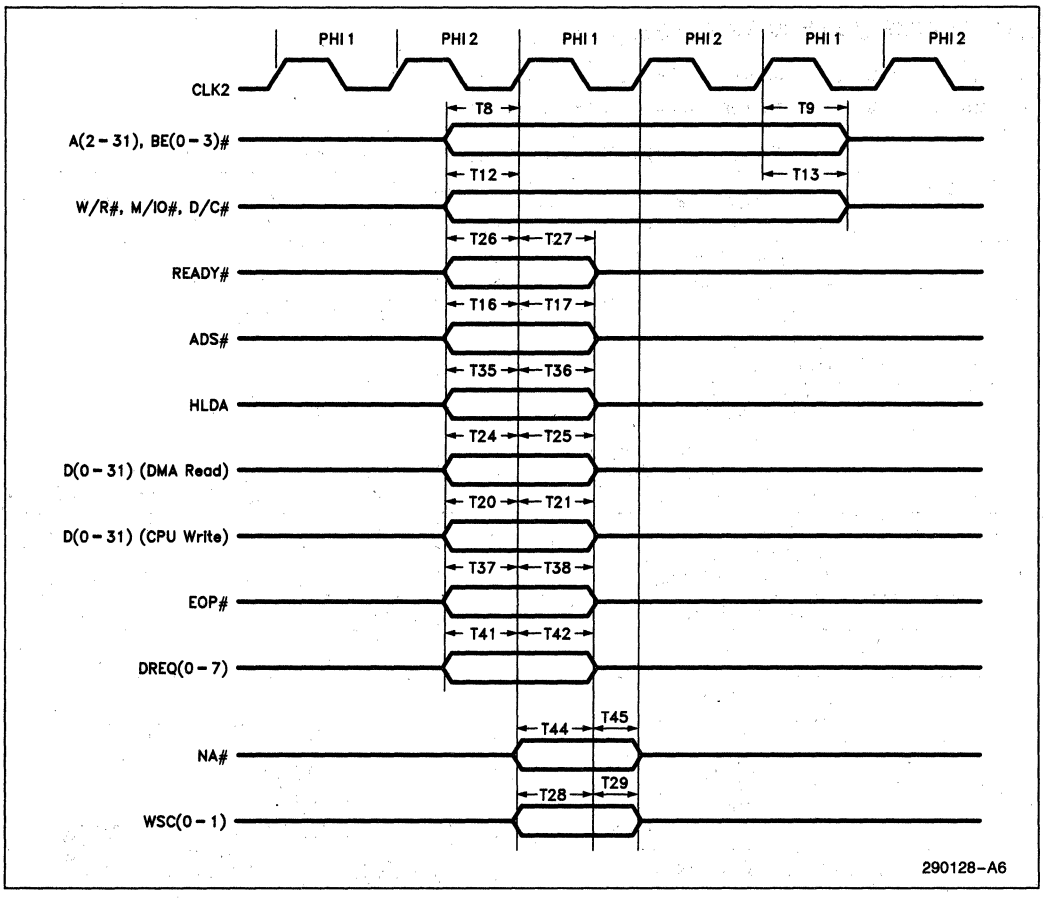


Figure 13-4. Input Setup and Hold Timing

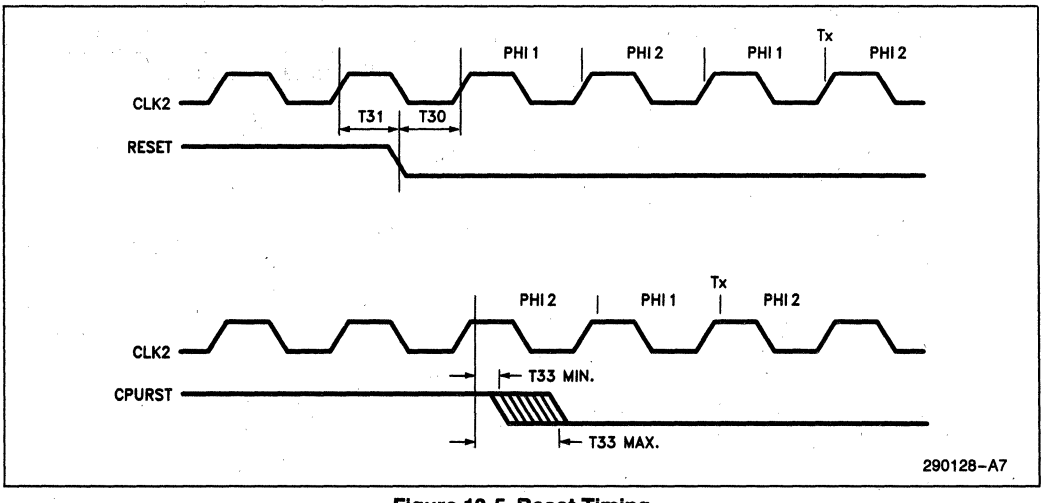


Figure 13-5. Reset Timing

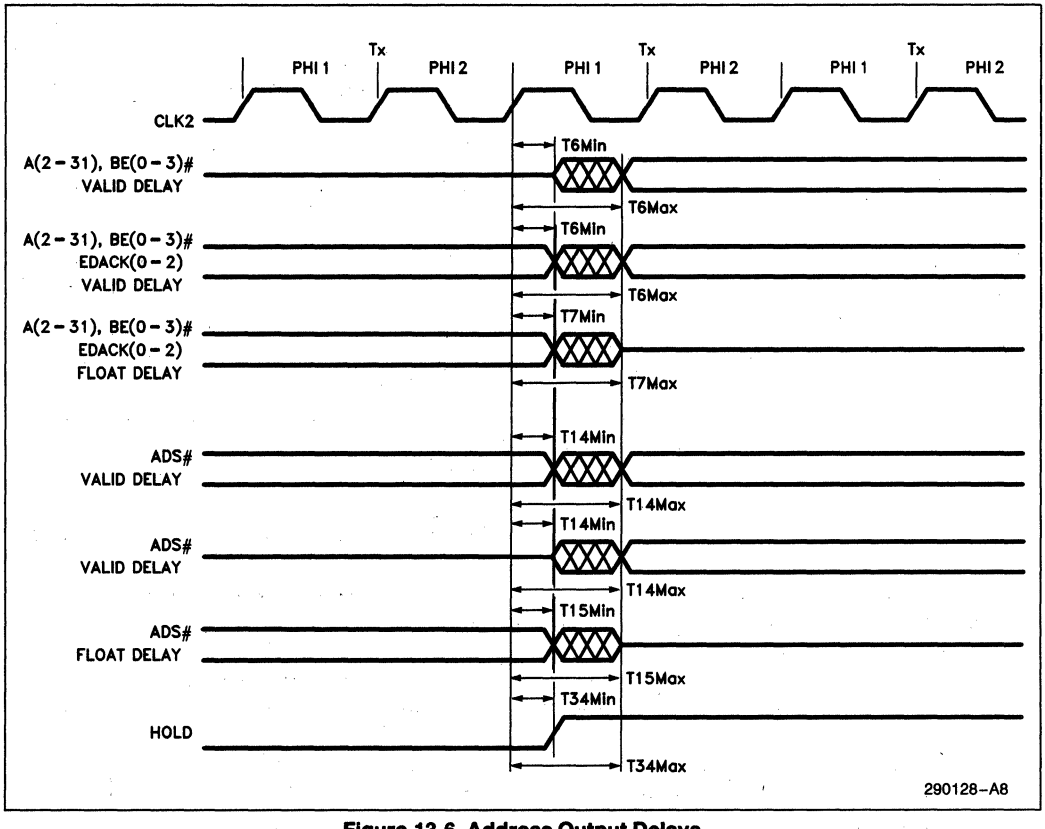


Figure 13-6. Address Output Delays

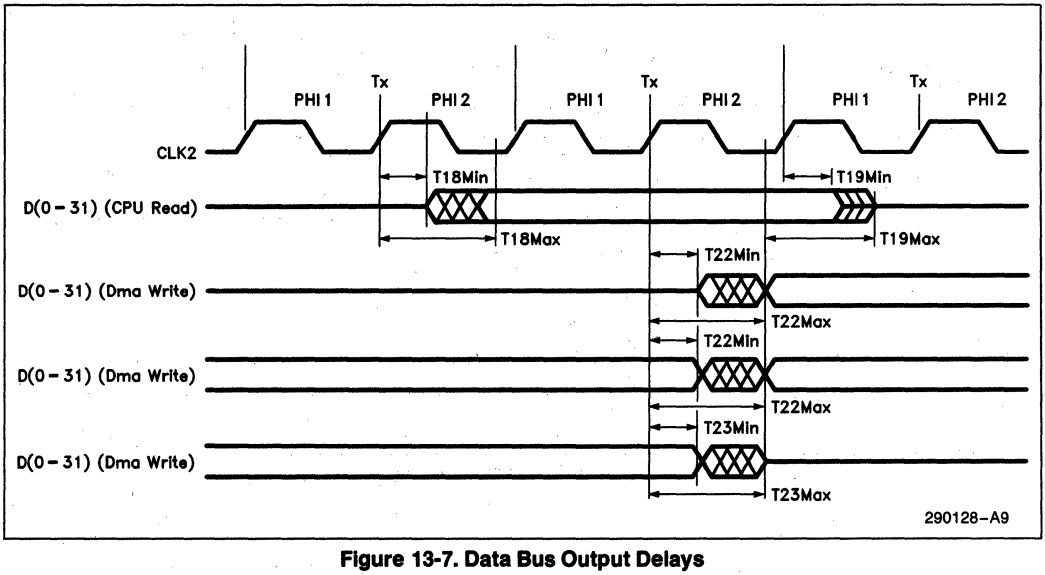


Figure 13-7. Data Bus Output Delays

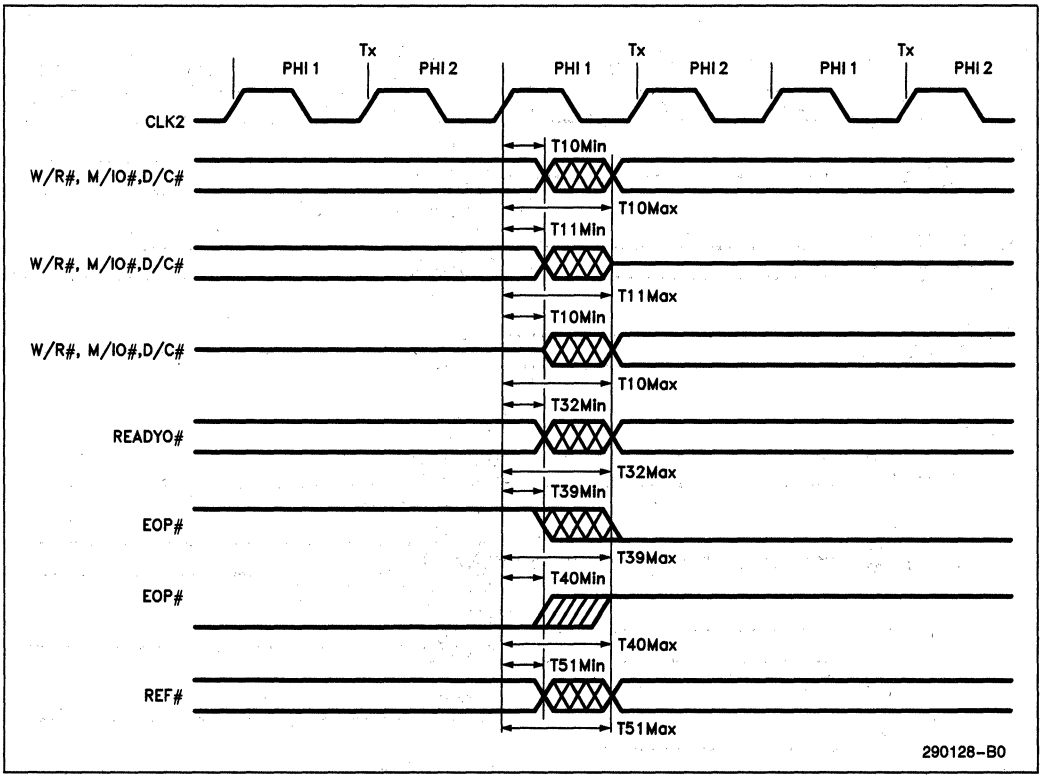


Figure 13-8. Control Output Delays

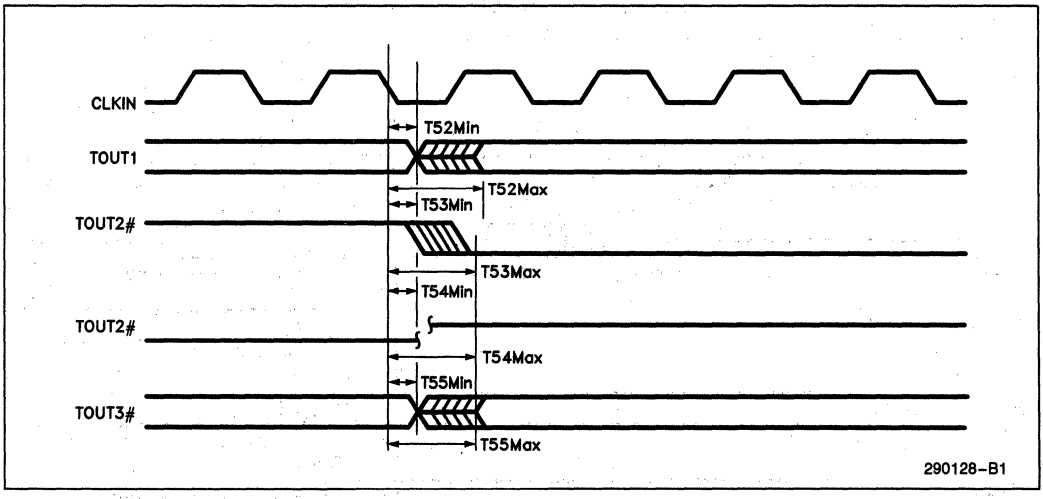


Figure 13-9. Timer Output Delays

APPENDIX A
Ports Listed by Address

Port Address (HEX)	Description
00	Read/Write DMA Channel 0 Target Address, A0–A15
01	Read/Write DMA Channel 0 Byte Count, B0–B15
02	Read/Write DMA Channel 1 Target Address, A0–A15
03	Read/Write DMA Channel 1 Byte Count, B0–B15
04	Read/Write DMA Channel 2 Target Address, A0–A15
05	Read/Write DMA Channel 2 Byte Count, B0–B15
06	Read/Write DMA Channel 3 Target Address, A0–A15
07	Read/Write DMA Channel 3 Byte Count, B0–B15
08	Read/Write DMA Channel 0–3 Status/Command I Register
09	Read/Write DMA Channel 0–3 Software Request Register
0A	Write DMA Channel 0–3 Set-Reset Mask Register
0B	Write DMA Channel 0–3 Mode Register I
0C	Write Clear Byte-Pointer FF
0D	Write DMA Master-Clear
0E	Write DMA Channel 0–3 Clear Mask Register
0F	Read/Write DMA Channel 0–3 Mask Register
10	Read/Write DMA Channel 0 Target Address, A24–A31
11	Read/Write DMA Channel 0 Byte Count, B16–B23
12	Read/Write DMA Channel 1 Target Address, A24–A31
13	Read/Write DMA Channel 1 Byte Count, B16–B23
14	Read/Write DMA Channel 2 Target Address, A24–A31
15	Read/Write DMA Channel 2 Byte Count, B16–B23
16	Read/Write DMA Channel 3 Target Address, A24–A31
17	Read/Write DMA Channel 3 Byte Count, B16–B23
18	Write DMA Channel 0–3 Bus Size Register
19	Read/Write DMA Channel 0–3 Chaining Register
1A	Write DMA Channel 0–3 Command Register II
1B	Write DMA Channel 0–3 Mode Register II
1C	Read/Write Refresh Control Register
1E	Reset Software Request Interrupt
20	Write Bank B ICW1, OCW2, or OCW3 Read Bank B Poll, Interrupt Request or In-Service Status Register
21	Write Bank B ICW2, ICW3, ICW4 or OCW1 Read Bank B Interrupt Mask Register
22	Read Bank B ICW2
28	Read/Write IRQ8 Vector Register
29	Read/Write IRQ9 Vector Register
2A	Reserved
2B	Read/Write IRQ11 Vector Register
2C	Read/Write IRQ12 Vector Register
2D	Read/Write IRQ13 Vector Register
2E	Read/Write IRQ14 Vector Register
2F	Read/Write IRQ15 Vector Register

APPENDIX A—Ports Listed by Address (Continued)

Port Address (HEX)	Description
30	Write Bank A ICW1, OCW2 or OCW3 Read Bank A Poll, Interrupt Request or In-Service Status Register
31	Write Bank A ICW2, ICW3, ICW4 or OCW1 Read Bank A Interrupt Mask Register
32	Read Bank A ICW2
38	Read/Write IRQ0 Vector Register
39	Read/Write IRQ1 Vector Register
3A	Read/Write IRQ1.5 Vector Register
3B	Read/Write IRQ3 Vector Register
3C	Read/Write IRQ4 Vector Register
3D	Reserved
3E	Reserved
3F	Read/Write IRQ7 Vector Register
40	Read/Write Counter 0 Register
41	Read/Write Counter 1 Register
42	Read/Write Counter 2 Register
43	Write Control Word Register I—Counter 0, 1, 2
44	Read/Write Counter 3 Register
45	Reserved
46	Reserved
47	Write Word Register II—Counter 3
61	Write Internal Control Port
64	Write CPU Reset Register (Data-1111XXX0H)
72	Read/Write Wait State Register 0
73	Read/Write Wait State Register 1
74	Read/Write Wait State Register 2
75	Read/Write Refresh Wait State Register
76	Reserved
77	Reserved
7D	Reserved
7E	Reserved
7F	Read/Write Relocation Register
80	Read/Write Internal Diagnostic Port 0
81	Read/Write DMA Channel 2 Target Address, A16–A23
82	Read/Write DMA Channel 3 Target Address, A16–A23
83	Read/Write DMA Channel 1 Target Address, A16–A23
87	Read/Write DMA Channel 0 Target Address, A16–A23
88	Read/Write Internal Diagnostic Port 1
89	Read/Write DMA Channel 6 Target Address, A16–A23
8A	Read/Write DMA Channel 7 Target Address, A16–A23
8B	Read/Write DMA Channel 5 Target Address, A16–A23
8F	Read/Write DMA Channel 4 Target Address, A16–A23

APPENDIX A—Ports Listed by Address (Continued)

Port Address (HEX)	Description
90	Read/Write DMA Channel 0 Requester Address, A0–A15
91	Read/Write DMA Channel 0 Requester Address, A16–A31
92	Read/Write DMA Channel 1 Requester Address, A0–A15
93	Read/Write DMA Channel 1 Requester Address, A16–A31
94	Read/Write DMA Channel 2 Requester Address, A0–A15
95	Read/Write DMA Channel 2 Requester Address, A16–A31
96	Read/Write DMA Channel 3 Requester Address, A0–A15
97	Read/Write DMA Channel 3 Requester Address, A16–A31
98	Read/Write DMA Channel 4 Requester Address, A0–A15
99	Read/Write DMA Channel 4 Requester Address, A16–A31
9A	Read/Write DMA Channel 5 Requester Address, A0–A15
9B	Read/Write DMA Channel 5 Requester Address, A16–A31
9C	Read/Write DMA Channel 6 Requester Address, A0–A15
9D	Read/Write DMA Channel 6 Requester Address, A16–A31
9E	Read/Write DMA Channel 7 Requester Address, A0–A15
9F	Read/Write DMA Channel 7 Requester Address, A16–A31
A0	Write Bank C ICW1, OCW2 or OCW3 Read Bank C Poll, Interrupt Request or In-Service Status Register
A1	Write Bank C ICW2, ICW3, ICW4 or OCW1 Read Bank C Interrupt Mask Register
A2	Read Bank C ICW2
A8	Read/Write IRQ16 Vector Register
A9	Read/Write IRQ17 Vector Register
AA	Read/Write IRQ18 Vector Register
AB	Read/Write IRQ19 Vector Register
AC	Read/Write IRQ20 Vector Register
AD	Read/Write IRQ21 Vector Register
AE	Read/Write IRQ22 Vector Register
AF	Read/Write IRQ23 Vector Register
C0	Read/Write DMA Channel 4 Target Address, A0–A15
C1	Read/Write DMA Channel 4 Byte Count, B0–B15
C2	Read/Write DMA Channel 5 Target Address, A0–A15
C3	Read/Write DMA Channel 5 Byte Count, B0–B15
C4	Read/Write DMA Channel 6 Target Address, A0–A15
C5	Read/Write DMA Channel 6 Byte Count, B0–B15
C6	Read/Write DMA Channel 7 Target Address, A0–A15
C7	Read/Write DMA Channel 7 Byte Count, B0–B15
C8	Read DMA Channel 4–7 Status/Command I Register
C9	Read/Write DMA Channel 4–7 Software Request Register
CA	Write DMA Channel 4–7 Set—Reset Mask Register
CB	Write DMA Channel 4–7 Mode Register I
CC	Reserved
CD	Reserved
CE	Write DMA Channel 4–7 Clear Mask Register
CF	Read/Write DMA Channel 4–7 Mask Register

APPENDIX A—Ports Listed by Address (Continued)

Port Address (HEX)	Description
D0	Read/Write DMA Channel 4 Target Address, A24–A31
D1	Read/Write DMA Channel 4 Byte Count, B16–B23
D2	Read/Write DMA Channel 5 Target Address, A24–A31
D3	Read/Write DMA Channel 5 Byte Count, B16–B23
D4	Read/Write DMA Channel 6 Target Address, A24–A31
D5	Read/Write DMA Channel 6 Byte Count, B16–B23
D6	Read/Write DMA Channel 7 Target Address, A24–A31
D7	Read/Write DMA Channel 7 Byte Count, B16–B23
D8	Write DMA Channel 4–7 Bus Size Register
D9	Read/Write DMA Channel 4–7 Chaining Register
DA	Write DMA Channel 4–7 Command Register II
DB	Write DMA Channel 4–7 Mode Register II

APPENDIX B
Ports Listed by Function

Port Address (HEX)	Description
	DMA CONTROLLER
0D	Write DMA Master-Clear
0C	Write DMA Clear Byte-Pointer FF
08	Read/Write DMA Channel 0–3 Status/Command I Register
C8	Read/Write DMA Channel 4–7 Status/Command I Register
1A	Write DMA Channel 0–3 Command Register II
DA	Write DMA Channel 4–7 Command Register II
0B	Write DMA Channel 0–3 Mode Register I
CB	Write DMA Channel 4–7 Mode Register I
1B	Write DMA Channel 0–3 Mode Register II
DB	Write DMA Channel 4–7 Mode Register II
09	Read/Write DMA Channel 0–3 Software Request Register
C9	Read/Write DMA Channel 4–7 Software Request Register
1E	Reset Software Request Interrupt
0E	Write DMA Channel 0–3 Clear Mask Register
CE	Write DMA Channel 4–7 Clear Mask Register
0F	Read/Write DMA Channel 0–3 Mask Register
CF	Read/Write DMA Channel 4–7 Mask Register
0A	Write DMA Channel 0–3 Set-Reset Mask Register
CA	Write DMA Channel 4–7 Set-Reset Mask Register
18	Write DMA Channel 0–3 Bus Size Register
D8	Write DMA Channel 4–7 Bus Size Register
19	Read/Write DMA Channel 0–3 Chaining Register
D9	Read/Write DMA Channel 4–7 Chaining Register
00	Read/Write DMA Channel 0 Target Address, A0–A15
87	Read/Write DMA Channel 0 Target Address, A16–A23
10	Read/Write DMA Channel 0 Target Address, A24–A31
01	Read/Write DMA Channel 0 Byte Count, B0–B15
11	Read/Write DMA Channel 0 Byte Count, B16–B23
90	Read/Write DMA Channel 0 Requester Address, A0–A15
91	Read/Write DMA Channel 0 Requester Address, A16–A31
02	Read/Write DMA Channel 1 Target Address, A0–A15
83	Read/Write DMA Channel 1 Target Address, A16–A23
12	Read/Write DMA Channel 1 Target Address, A24–A31
03	Read/Write DMA Channel 1 Byte Count, B0–B15
13	Read/Write DMA Channel 1 Byte Count, B16–B23
92	Read/Write DMA Channel 1 Requester Address, A0–A15
93	Read/Write DMA Channel 1 Requester Address, A16–A31

APPENDIX B—Ports Listed by Function (Continued)

Port Address (HEX)	Description
	DMA CONTROLLER
04	Read/Write DMA Channel 2 Target Address, A0–A15
81	Read/Write DMA Channel 2 Target Address, A16–A23
14	Read/Write DMA Channel 2 Target Address, A24–A31
05	Read/Write DMA Channel 2 Byte Count, B0–B15
15	Read/Write DMA Channel 2 Byte Count, B16–B23
94	Read/Write DMA Channel 2 Requester Address, A0–A15
95	Read/Write DMA Channel 2 Requester Address, A16–A31
06	Read/Write DMA Channel 3 Target Address, A0–A15
82	Read/Write DMA Channel 3 Target Address, A16–A23
16	Read/Write DMA Channel 3 Target Address, A24–A31
07	Read/Write DMA Channel 3 Byte Count, B0–B15
17	Read/Write DMA Channel 3 Byte Count, B16–B23
96	Read/Write DMA Channel 3 Requester Address, A0–A15
97	Read/Write DMA Channel 3 Requester Address, A16–A31
C0	Read/Write DMA Channel 4 Target Address, A0–A15
8F	Read/Write DMA Channel 4 Target Address, A16–A23
D0	Read/Write DMA Channel 4 Target Address, A24–A31
C1	Read/Write DMA Channel 4 Byte Count, B0–B15
D1	Read/Write DMA Channel 4 Byte Count, B16–B23
98	Read/Write DMA Channel 4 Requester Address, A0–A15
99	Read/Write DMA Channel 4 Requester Address, A16–A31
C2	Read/Write DMA Channel 5 Target Address, A0–A15
8B	Read/Write DMA Channel 5 Target Address, A16–A23
D2	Read/Write DMA Channel 5 Target Address, A24–A31
C3	Read/Write DMA Channel 5 Byte Count, B0–B15
D3	Read/Write DMA Channel 5 Byte Count, B16–B23
9A	Read/Write DMA Channel 5 Requester Address, A0–A15
9B	Read/Write DMA Channel 5 Requester Address, A16–A31
C4	Read/Write DMA Channel 6 Target Address, A0–A15
89	Read/Write DMA Channel 6 Target Address, A16–A23
D4	Read/Write DMA Channel 6 Target Address, A24–A31
C5	Read/Write DMA Channel 6 Byte Count, B0–B15
D5	Read/Write DMA Channel 6 Byte Count, B16–B23
9C	Read/Write DMA Channel 6 Requester Address, A0–A15
9D	Read/Write DMA Channel 6 Requester Address, A16–A31
C6	Read/Write DMA Channel 7 Target Address, A0–A15
8A	Read/Write DMA Channel 7 Target Address, A16–A23
D6	Read/Write DMA Channel 7 Target Address, A24–A31
C7	Read/Write DMA Channel 7 Byte Count, B0–B15
D7	Read/Write DMA Channel 7 Byte Count, B16–B23
9E	Read/Write DMA Channel 7 Requester Address, A0–A15
9F	Read/Write DMA Channel 7 Requester Address, A16–A31

APPENDIX B—Ports Listed by Function (Continued)

Port Address (HEX)	Description
	INTERRUPT CONTROLLER
20	Write Bank B ICW1, OCW2, or OCW3 Read Bank B Poll, Interrupt Request or In-Service Status Register
21	Write Bank B ICW2, ICW3, ICW4 or OCW1 Read Bank B Interrupt Mask Register
22	Read Bank B ICW2
28	Read/Write IRQ8 Vector Register
29	Read/Write IRQ9 Vector Register
2A	Reserved
2B	Read/Write IRQ11 Vector Register
2C	Read/Write IRQ12 Vector Register
2D	Read/Write IRQ13 Vector Register
2E	Read/Write IRQ14 Vector Register
2F	Read/Write IRQ15 Vector Register
A0	Write Bank C ICW1, OCW2 or OCW3 Read Bank C Poll, Interrupt Request or In-Service Status Register
A1	Write Bank C ICW2, ICW3, ICW4 or OCW1 Read Bank C Interrupt Mask Register
A2	Read Bank C ICW2
A8	Read/Write IRQ16 Vector Register
A9	Read/Write IRQ17 Vector Register
AA	Read/Write IRQ18 Vector Register
AB	Read/Write IRQ19 Vector Register
AC	Read/Write IRQ20 Vector Register
AD	Read/Write IRQ21 Vector Register
AE	Read/Write IRQ22 Vector Register
AF	Read/Write IRQ23 Vector Register
30	Write Bank A ICW1, OCW2 or OCW3 Read Bank A Poll, Interrupt Request oor In-Service Status Register
31	Write Bank A ICW2, ICW3, ICW4 or OCW1 Read Bank A Interrupt Mask Register
32	Read Bank A ICW2
38	Read/Write IRQ0 Vector Register
39	Read/Write IRQ1 Vector Register
3A	Read/Write IRQ1.5 Vector Register
3B	Read/Write IRQ3 Vector Register
3C	Read/Write IRQ4 Vector Register
3D	Reserved
3E	Reserved
3F	Read/Write IRQ7 Vector Register

APPENDIX B—Ports Listed by Function (Continued)

Port Address (HEX)	Description
	PROGRAMMABLE INTERVAL TIMER
40	Read/Write Counter 0 Register
41	Read/Write Counter 1 Register
42	Read/Write Counter 2 Register
43	Write Control Word Register I—Counter 0, 1, 2
44	Read/Write Counter 3 Register
47	Write Word Register II—Counter 3
	CPU RESET
64	Write CPU Reset Register (Data-1111XXX0H)
	WAIT STATE GENERATOR
72	Read/Write Wait State Register 0
73	Read/Write Wait State Register 1
74	Read/Write Wait State Register 2
75	Read/Write Refresh Wait State Register
	DRAM REFRESH CONTROLLER
1C	Read/Write Refresh Control Register
	INTERNAL CONTROL AND DIAGNOSTIC PORTS
61	Write Internal Control Port
80	Read/Write Internal Diagnostic Port 0
88	Read/Write Internal Diagnostic Port 1
	RELOCATION REGISTER
7F	Read/Write Relocation Register
	INTEL RESERVED PORTS
2A	Reserved
3D	Reserved
3E	Reserved
45	Reserved
46	Reserved
76	Reserved
77	Reserved
7D	Reserved
7E	Reserved
CC	Reserved
CD	Reserved

APPENDIX C
Pin Descriptions

The 82380 provides all of the signals necessary to interface it to an 80386 processor. It has separate 32-bit address and data buses. It also has a set of control signals to support operation as a bus master or a bus slave. Several special function signals exist on the 82380 for interfacing the system support peripherals to their respective system counterparts. Following are the definitions of the individual pins of the 82380. These brief descriptions are provided as a reference. Each signal is further defined within the sections which describe the associated 82380 function.

A2-A31 I/O ADDRESS BUS

This is the 32-bit address bus. The addresses are doubleword memory and I/O addresses. These are three-state signals which are active only during Master mode. The address lines should be connected directly to the 80386's local bus.

BE0# I/O BYTE-ENABLE 0

BE0# active indicates that data bits D0–D7 are being accessed or are valid. It is connected directly to the 80386's BE0#. The byte enable signals are active outputs when the 82380 is in the Master mode.

BE1# I/O BYTE-ENABLE 1

BE1# active indicates that data bits D8–D15 are being accessed or are valid. It is connected directly to the 80386's BE1#. The byte enable signals are active only when the 82380 is in the Master mode.

BE2# I/O BYTE-ENABLE 2

BE2# active indicates that data bits D15–D23 are being accessed or are valid. It is connected directly to the 80386's BE2#. The byte enable signals are active only when the 82380 is in the Master mode.

BE3# I/O BYTE-ENABLE 3

BE3# active indicates that data bits D24–D31 are being accessed or are valid. The byte enable signals are active only when the 82380 is in the Master mode. This pin should be connected directly to the 80386's BE3#. This pin is used for factory testing and must be low during reset. The 80386 drives BE3# low during reset.

D0–D31 I/O DATA BUS

This is the 32-bit data bus. These pins are active outputs during interrupt acknowledges, during Slave accesses, and when the 82380 is in the Master mode.

CLK2 I PROCESSOR CLOCK

This pin must be connected to CLK2. The 82380 monitors the phase of this clock in order to remain synchronized with the 80386. This clock drives all of the internal synchronous circuitry.

D/C# I/O DATA/CONTROL

D/C# is used to distinguish between 80386 control cycles and DMA or 80386 data access cycles. It is active as an output only in the Master mode.

W/R# I/O WRITE/READ

W/R# is used to distinguish between write and read cycles. It is active as an output only in the Master mode.

M/IO# I/O MEMORY/IO

M/IO# is used to distinguish between memory and IO accesses. It is active as an output only in the Master mode.

ADS# I/O ADDRESS STATUS

This signal indicates presence of a valid address on the address bus. It is active as output only in the Master mode. ADS# is active during the first T-state where addresses and control signals are valid.

NA# I NEXT ADDRESS

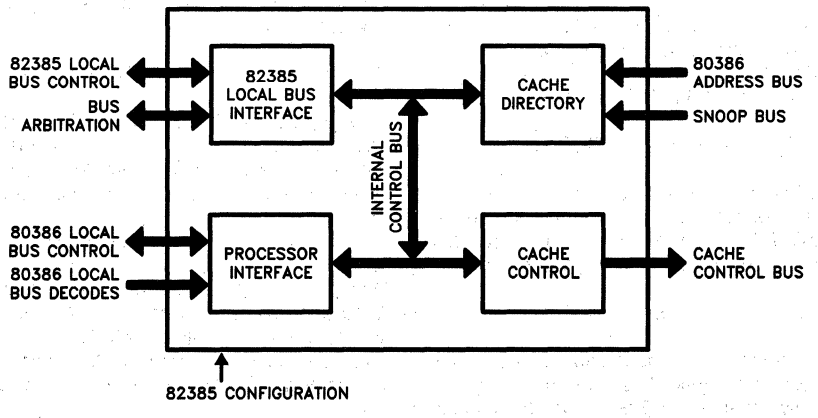
Asserted by a peripheral or memory to begin a pipelined address cycle. This pin is monitored only while the 82380 is in the Master mode. In the Slave mode, pipelining is determined by the current and past status of the ADS# and READY# signals.

82385
HIGH PERFORMANCE
32-BIT CACHE CONTROLLER

- **Improves 80386 System Performance**
  - **— Reduces Average CPU Wait States to Nearly Zero**
  - **— Zero Wait State Read Hit**
  - **— Zero Wait State Posted Memory Writes**
  - **— Allows Other Masters to Access the System Bus More Readily**
- **Hit Rates up to 99%**
- **Optimized as 80386 Companion**
  - **— Simple 80386 Interface**
  - **— Part of 386-Based Compute Engine Including 80387 Numerics Coprocessor and 82380 Integrated System Peripheral**
  - **— 16 MHz, 20 MHz, and 25 MHz Operation**
- **Software Transparent**
- **Synchronous Dual Bus Architecture**
  - **— Bus Watching Maintains Cache Coherency**
- **Maps Full 80386 Address Space (4 Gigabytes)**
- **Flexible Cache Mapping Policies**
  - **— Direct Mapped or 2-Way Set Associative Cache Organization**
  - **— Supports Non-Cacheable Memory Space**
  - **— Unified Cache for Code and Data**
- **Integrates Cache Directory and Cache Management Logic**
- **High Speed CHMOS III Technology**
- **132-Pin PGA Package**

The 82385 Cache Controller is a high performance 32-bit peripheral for Intel's 80386 Microprocessor. It stores a copy of frequently accessed code and data from main memory in a zero wait state local cache memory. The 82385 enables the 80386 to run at its full potential by reducing the average number of CPU wait states to nearly zero. The dual bus architecture of the 82385 allows other masters to access system resources while the 80386 operates locally out of its cache. In this situation, the 82385's "bus watching" mechanism preserves cache coherency by monitoring the system bus address lines at no cost to system or local throughput.

The 82385 is completely software transparent, protecting the integrity of system software. High performance and board savings are achieved because the 82385 integrates a cache directory and all cache management logic on one chip.



82385 Internal Block Diagram

October 1988
Order Number: 290143-003

1.0 82385 FUNCTIONAL OVERVIEW

The 82385 Cache Controller is a high performance 32-bit peripheral for Intel's 80386 microprocessor. This chapter provides an overview of the 82385, and of the basic architecture and operation of an 80386/82385 system.

1.1 82385 OVERVIEW

The main function of a cache memory system is to provide fast local storage for frequently accessed code and data. The cache system intercepts 80386 memory references to see if the required data resides in the cache. If the data resides in the cache (a hit), it is returned to the 80386 without incurring wait states. If the data is not cached (a miss), the reference is forwarded to the system and the data retrieved from main memory. An efficient cache will yield a high "hit rate" (the ratio of cache hits to total 80386 accesses), such that the majority of accesses are serviced with zero wait states. The net effect is that the wait states incurred in a relatively infrequent miss are averaged over a large number of accesses, resulting in an average of nearly zero wait states per access. Since cache hits are serviced locally, a processor operating out of its local cache has a much lower "bus utilization" which reduces system bus bandwidth requirements, making more bandwidth available to other bus masters.

The 82385 Cache Controller integrates a cache directory and all cache management logic required to support an external 32 Kbyte cache. The cache directory structure is such that the entire physical address range of the 80386 (4 Gigabytes) is mapped into the cache. Provision is made to allow areas of memory to be set aside a non-cacheable. The user has two cache organization options: direct mapped and 2-way set associative. Both provide the high hit rates necessary to make a large, relatively slow main memory array look like a fast, zero wait state memory to the 80386.

A good hit rate is an essential ingredient of a successful cache implementation. Hit rate is the measure of how efficient a cache is in maintaining a copy of the most frequently requested code and data. However, efficiency is not the only factor for performance consideration. Just as essential are sound cache management policies. These policies refer to the handling of 80386 writes, preservation of cache coherency, and ease of system design. The 82385's "posted write" capability allows 80386 memory writes, including non-cacheable, to run with zero wait states, and the 82385's "bus watching" mechanism preserves cache coherency with no impact on system performance. Physically, the 82385 ties directly to the 80386 with virtually no external logic.

1.2 SYSTEM OVERVIEW I: BUS STRUCTURE

A good grasp of the bus structure of an 80386/82385 system is essential in understanding both the 82385 and its role in an 80386 system. The following is a description of this structure.

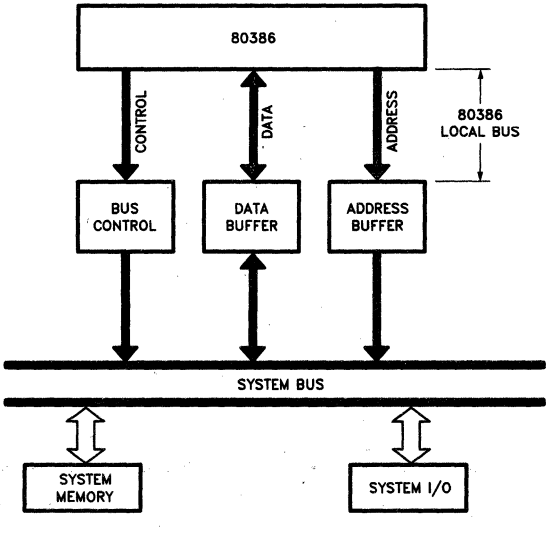


Figure 1-1. 80386 System Bus Structure

1.2.1 80386 Local Bus/82385 Local Bus/System Bus

Figure 1-1 depicts the bus structure of a typical 80386 system. The "80386 Local Bus" consists of the physical 80386 address, data, and control busses. The local address and data busses are buffered and/or latched to become the "system" address and data busses. The local control bus is decoded by bus control logic to generate the various system bus read and write commands.

The addition of an 82385 Cache Controller causes a separation of the 80386 bus into two distinct busses: the actual 80386 local bus and the "82385 Local Bus" (Figure 1-2). The 82385 local bus is designed to look like the front end of an 80386 by providing 82385 local bus equivalents to all appropriate 80386 signals. The system ties to this "80386-like" front end just as it would to an actual 80386. The 80386 simply sees a fast system bus, and the system sees an 80386 front end with low bus bandwidth requirements. The cache subsystem is transparent to both. Note that the 82385 local bus is not simply a buffered version of the 80386 bus, but rather is distinct from, and able to operate in parallel with the 80386 bus. Other masters residing on either the 82385 local bus or system bus are free to manage system resources while the 80386 operates out of its cache.

1.2.2 Bus Arbitration

The 82385 presents the "80386-like" interface which is called the 82385 local bus. Whereas the 80386 provides a Hold Request/Hold Acknowledge bus arbitration mechanism via its HOLD and HLDA pins, the 82385 provides an equivalent mechanism via its BHOLD and BHLDA pins. (These signals are described in section 3.7.) When another master requests the 82385 local bus, it issues the request to the 82385 via BHOLD. Typically, at the end of the current 82385 local bus cycle, the 82385 will release the 82385 local bus and acknowledge the request via BHLDA. The 80386 is of course free to continue operating on the 80386 local bus while another master owns the 82385 local bus.

1.2.3 Master/Slave Operation

The above 82385 local bus arbitration discussion is strictly true only when the 82385 is programmed for "Master" mode operation. The user can, however, configure the 82385 for "Slave" mode operation. (Programming is done via a hardware strap option.) The roles of BHOLD and BHLDA are reversed for an 82385 in slave mode; BHOLD is now an output indicating a request to control the bus, and BHLDA is an input indicating that a request has been granted. An 82385 programmed in slave mode drives the 82385 local bus only when it has requested and subsequently been granted bus control. This allows multiple 80386/82385 subsystems to reside on the same 82385 local bus (Figure 1-3).

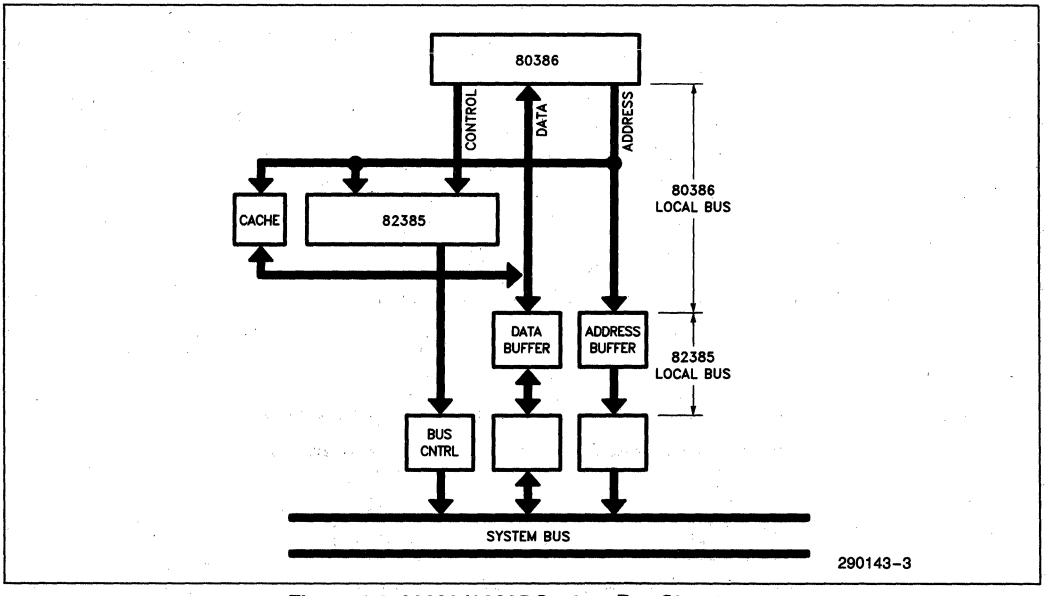


Figure 1-2. 80386/82385 System Bus Structure

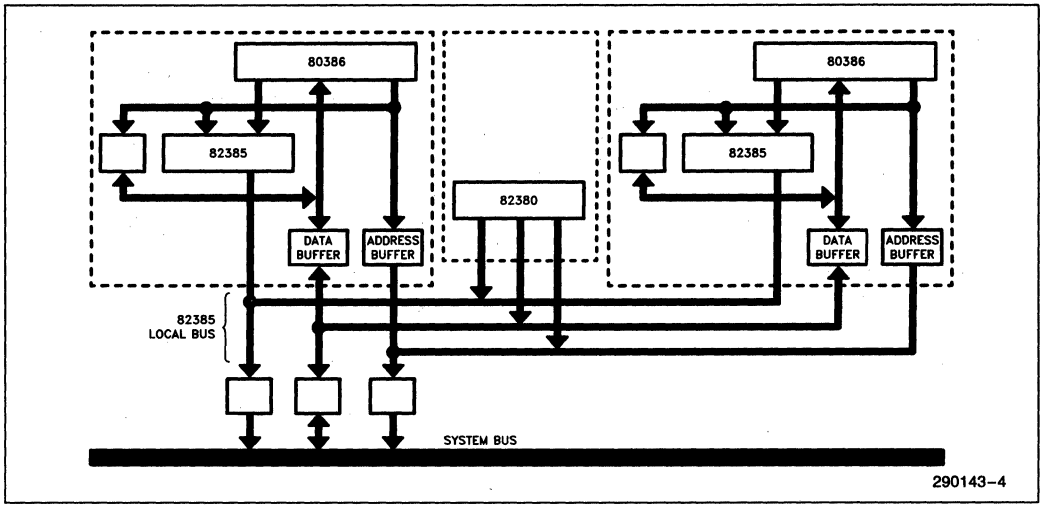


Figure 1-3. Multi-Master/Multi-Cache Environment

1.2.4 Cache Coherency

Ideally, a cache contains a copy of the most heavily used portions of main memory. To maintain cache "coherency" is to make sure that this local copy is identical to main memory. In a system where multiple masters can access the same memory, there is always a risk that one master will alter the contents of a memory location that is duplicated in the local cache of another master. (The cache is said to contain "stale" data.) One rather restrictive solution is to not allow cache subsystems to cache shared memory. Another simple solution is to flush the cache anytime another master writes to system memory. However, this can seriously degrade system performance as excessive cache flushing will reduce the hit rate of what may otherwise be a highly efficient cache.

The 82385 preserves cache coherency via "bus watching" (also called snooping), a technique that neither impacts performance nor restricts memory mapping. An 82385 that is not currently bus master monitors system bus cycles, and when a write cycle by another master is detected (a snoop), the system address is sampled and used to see if the referenced location is duplicated in the cache. If so (a snoop hit), the corresponding cache entry is invalidated, which will force the 80386 to fetch the up-to-date data from main memory the next time it accesses this modified location. Figure 1-4 depicts the general form of bus watching.

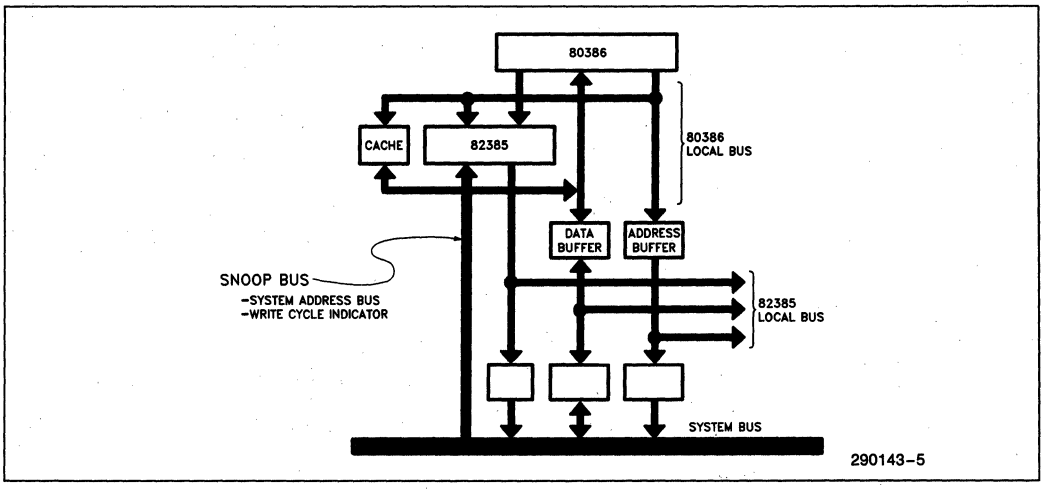


Figure 1-4. 82385 Bus Watching—Monitor System Bus Write Cycles

1.3 SYSTEM OVERVIEW II: BASIC OPERATION

This discussion is an overview of the basic operation of an 80386/82385 system. Items discussed include the 82385's response to all 80386 cycles, including interrupt acknowledges, halts, and shutdowns. Also discussed are non-cacheable and local accesses.

1.3.1 80386 Memory Code and Data Read Cycles

1.3.1.1 READ HITS

When the 80386 initiates a memory code or data read cycle, the 82385 compares the high order bits of the 80386 address bus with the appropriate addresses (tags) stored in its on-chip directory. (The directory structure is described in chapter 2.) If the 82385 determines that the requested data is in the cache, it issues the appropriate control signals that direct the cache to drive the requested data onto the 80386 data bus, where it is read by the 80386. The 82385 terminates the 80386 cycle without inserting any wait states.

1.3.1.2 READ MISSES

If the 82385 determines that the requested data is not in the cache, the request is forwarded to the 82385 local bus and the data retrieved from main memory. As the data returns from main memory, it is directed to the 80386 and also written into the cache. Concurrently, the 82385 updates the cache directory such that the next time this particular piece of information is requested by the 80386, the 82385 will find it in the cache and return it with zero wait states.

The basic unit of transfer between main memory and cache memory in a cache subsystem is called the line size. In an 82385 system, the line size is one 32-bit aligned doubleword. During a read miss, all four 82385 local bus byte enables are active. This ensures that a full 32-bit entry is written into the cache. (The 80386 simply ignores what it did not request.) In any other type of 80386 cycle that is forwarded to the 82385 local bus, the logic levels of the 80386 byte enables are duplicated on the 82385 local bus.

The 82385 does not actively fetch main memory data independently of the 80386. The 82385 is essentially a passive device which only monitors the address bus and activates control signals. The read miss is the only mechanism by which main memory data is copied into the cache and validated in the cache directory.

In an isolated read miss, the number of wait states seen by the 80386 is that required by the system memory to respond with data plus the cache comparison cycle (hit/miss decision). The cache system must determine that the cycle is a miss before it can begin the system memory access. However, since misses most often occur consecutively, the 82385 will begin 80386 address pipelined cycles to effectively "hide" the comparison cycle beyond the first miss (refer to section 4.1.3).

The 82385 can execute a main memory access on the 82385 local bus only if it currently owns the bus. If not, an 82385 in master mode will run the cycle after the current master releases the bus. An 82385 in slave mode will issue a hold request, and will run the cycle as soon as the request is acknowledged. (This is true for any read or write cycle that needs to run on the 82385 local bus.)

1.3.2 80386 Memory Write Cycles

The 82385's "posted write" capability allows the majority of 80386 memory write cycles to run with zero wait states. The primary memory update policy implemented in a posted write is the traditional cache "write through" technique, which implies that main memory is always updated in any memory write cycle. If the referenced location also happens to reside in the cache (a write hit), the cache is updated as well.

Beyond this, a posted write latches the 80386 address, data, and cycle definition signals, and the 80386 local bus cycle is terminated without any wait states, even though the corresponding 82385 local bus cycle is not yet completed, or perhaps not even started. A posted write is possible because the 82385's bus state machine, which is almost identical to the 80386 bus state machine, is able to run 82385 local bus cycles independently of the 80386. The only time the 80386 sees wait states in a write cycle is when a previously latched write has not yet been completed on the 82385 local bus. An 80386 write can be posted even if the 82385 does not currently own the 82385 local bus. In this case, an 82385 in master mode will run the cycle as soon as the current master releases the bus, and an 82385 in slave mode will request the bus and run the cycle when the request is acknowledged. The 80386 is free to continue operating out of its cache (on the 80386 local bus) during this time.

1.3.3 Non-Cacheable Cycles

Non-cacheable cycles fall into one of two categories: cycles decoded as non-cacheable, and cycles

that are by default non-cacheable according to the 82385's design. All non-cacheable cycles are forwarded to the 82385 local bus. Non-cacheable cycles have no effect on the cache or cache directory.

The 82385 allows the system designer to define areas of main memory as non-cacheable. The 80386 address bus is decoded and the decode output is connected to the 82385's non-cacheable access (NCA#) input. This decoding is done in the first 80386 bus state in which the non-cacheable cycle address becomes available. Non-cacheable read cycles resemble cacheable read miss cycles, except that the cache and cache directory are unaffected. Non-cacheable writes, like all writes, are posted.

The 82385 defines certain cycles as non-cacheable without using its non-cacheable access input. These include I/O cycles, interrupt acknowledge cycles, and halt/shutdown cycles. I/O reads and interrupt acknowledge cycles execute as any other non-cacheable read. I/O write cycles and Halt/Shutdown cycles, as with other non-cacheable writes, are posted. During a halt/shutdown condition, the 82385 local bus duplicates the behavior of the 80386, including the ability to recognize and respond to a BHOLD request. (The 82385's bus watching mechanism is functional in this condition.)

1.3.3.1 16-BIT MEMORY SPACE

The 82385 does not cache 16-bit memory space (as decoded by the 80386 BS16# input), but does make provisions to handle 16-bit space as non-cacheable. (There is no 82385 equivalent to the 80386 BS16# input.) In a system without an 82385, the 80386 BS16# input need not be asserted until the last state of a 16-bit cycle for the 80386 to recognize it as such (unless NA# is sampled active earlier in the cycle.) The 82385, however, needs this information earlier, specifically at the end of the first 80386 bus state in which the address of the 16-bit cycle becomes available. The result is that in a system without an 82385, 16-bit devices can inform the 80386 that they are 16-bit devices "on the fly," while in a system with an 82385, devices decoded as 16-bit (using the 80386 BS16#) must be located in address space set aside for 16-bit devices. If 16-bit space is decoded according to 82385 guidelines (as described later in the data sheet), then the 82385 will handle 16-bit cycles just like the 80386 does, including effectively locking the two halves of a non-aligned 16-bit transfer from interruption by another master.

1.3.4 80386 Local Bus Cycles

80386 Local Bus Cycles are accesses to resources on the 80386 local bus rather than to the 82385 itself. The 82385 simply ignores these accesses: they are neither forwarded to the system nor do they affect the cache. The designer sets aside memory and/or I/O space for local resources by decoding the 80386 address bus and feeding the decode to the 82385's local bus access (LBA#) input. The designer can also decode the 80386 cycle definition signals to keep specific 80386 cycles from being forwarded to the system. For example, a multi-processor design may wish to capture and remedy an 80386 shutdown locally without having it detected by the rest of the system. Note that in such a design, the local shutdown cycle must be terminated by local bus control logic. The 80387 Numerics Coprocessor is considered an 80386 local bus resource, but it need not be decoded as such by the user since the 82385 is able to internally recognize 80387 accesses via the M/IO# and A31 pins.

1.3.5 Summary of 82385 Response to All 80386 Cycles

Table 1-1 summarizes the 82385 response to all 80386 bus cycles, as conditioned by whether or not the cycle is decoded as local or non-cacheable. The table describes the impact of each cycle on the cache and on the cache directory, and whether or not the cycle is forwarded to the 82385 local bus. Whenever the 82385 local bus is marked "IDLE", it implies that this bus is available to other masters.

Table 1-1. 82385 Response to 80386 Cycles

80386 Bus Cycle Definition					82385 Response when Decoded as Cacheable			82385 Response when Decoded as Non-Cacheable			82385 Response when Decoded as an 80386 Local Bus Access		
M/IO#	D/C#	W/R#	80386 Cycle		Cache	Cache Directory	82385 Local Bus	Cache	Cache Directory	82385 Local Bus	Cache	Cache Directory	82385 Local Bus
0	0	0	INT ACK	N/A	—	—	INT ACK	—	—	INT ACK	—	—	IDLE
0	0	1	UNDEFINED	N/A			UNDEFINED			UNDEFINED			IDLE
0	1	0	I/O READ	N/A	—	—	I/O READ	—	—	I/O READ	—	—	IDLE
0	1	1	I/O WRITE	N/A	—	—	I/O WRITE	—	—	I/O WRITE	—	—	IDLE
1	0	0	MEM CODE READ	HIT	CACHE READ	—	IDLE	—	—	MEM CODE READ	—	—	IDLE
				MISS	CACHE WRITE	DATA VALIDATION	MEM CODE READ						
1	0	1	HALT/ SHUTDOWN	N/A	—	—	HALT/ SHUTDOWN	—	—	HALT/ SHUTDOWN	—	—	IDLE
1	1	0	MEM DATA READ	HIT	CACHE READ	—	IDLE	—	—	MEM DATA READ	—	—	IDLE
				MISS	CACHE WRITE	DATA VALIDATION	MEM DATA READ						
1	1	1	MEM DATA WRITE	HIT	CACHE WRITE	—	MEM DATA WRITE	—	—	MEM DATA WRITE	—	—	IDLE
				MISS	—	—	MEM DATA WRITE						

NOTES:
- A dash (—) indicates that the cache and cache directory are unaffected. This table does not reflect how an access affects the LRU bit.
- An "IDLE" 82385 Local Bus implies that this bus is available to other masters.
- The 82385's response to 80387 accesses is the same as when decoded as an 80386 Local Bus access.
- The only other operations that affect the cache directory are:
  1. RESET or Cache Flush—all tag valid bits cleared.
  2. Snoop Hit—corresponding line valid bit cleared.

1.3.6 Bus Watching

As previously discussed, the 82385 "qualifies" an 80386 bus cycle in the first bus state in which the address and cycle definition signals of the cycle become available. The cycle is qualified as read or write, cacheable or non-cacheable, etc. Cacheable cycles are further classified as hit or miss according to the results of the cache comparison, which accesses the 82385 directory and compares the appropriate directory location (tag) to the current 80386 address. If the cycle turns out to be non-cacheable or a 80386 local bus access, the hit/miss decision is ignored. The cycle qualification requires one 80386 state. Since the fastest 80386 access is two states, the second state can be used for bus watching.

When the 82385 does not own the system bus, it monitors system bus cycles. If another master writes into main memory, the 82385 latches the system address and executes a cache look-up to see if the altered main memory location resides in the cache. If so (a snoop hit), the cache entry is marked invalid in the cache directory. Since the directory is at most only being used every other state to qualify 80386 accesses, snoop look-ups are interleaved between 80386 local bus look-ups. The cache directory is time multiplexed between the 80386 address and the latched system address. The result is that all snoops are caught and serviced without slowing down the 80386, even when running zero wait state hits on the 80386 local bus.

1.3.7 Cache Flush

The 82385 offers a cache flush input. When activated, this signal causes the 82385 to invalidate all data which had previously been cached. Specifically, all tag valid bits are cleared. (Refer to the 82385 directory structure in chapter 2.) Therefore, the cache is effectively empty and subsequent cycles are misses until the 80386 begins repeating the new accesses (hits). The primary use of the FLUSH input is for diagnostics and multi-processor support.

NOTE:
The use of this pin as a coherency mechanism may impact software transparency.

2.0 82385 CACHE ORGANIZATION

The 82385 supports two cache organizations: a simple direct mapped organization and a slightly more complex, higher performance two way set associative organization. The choice is made by strapping an 82385 input (2W/D#) either high or low. This chapter describes the structure and operation of both organizations.

2.1 DIRECT MAPPED CACHE

2.1.1 Direct Mapped Cache Structure and Terminology

Figure 2-1 depicts the relationship between the 82385's internal cache directory, the external cache memory, and the 80386's 4 Gigabyte physical address space. The 4 Gigabytes can conceptually be thought of as cache "pages" each being 8K doublewords (32 Kbytes) deep. The page size matches the cache size. The cache can be further divided into 1024 (0 thru 1023) sets of eight doublewords (8 x 32 bits). Each 32-bit doubleword is called a "line." The unit of transfer between the main memory and cache is one line.

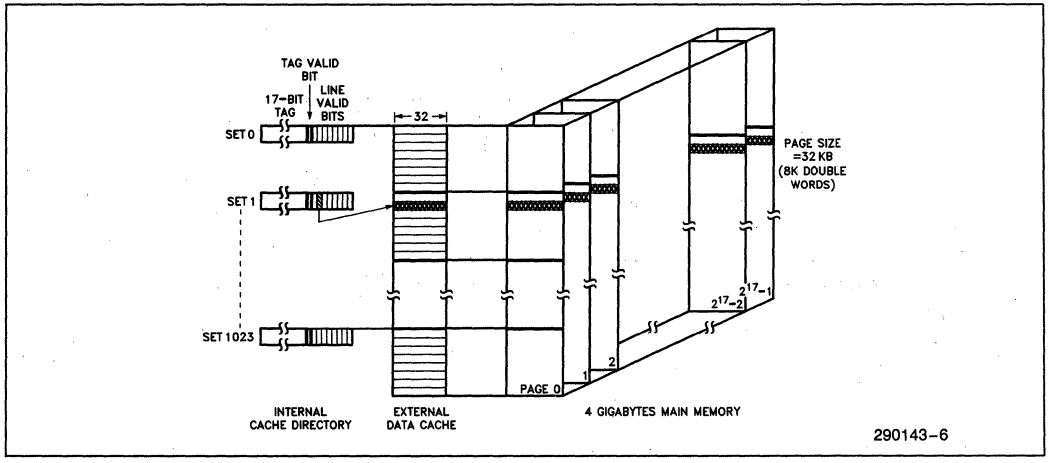


Figure 2-1. Direct Mapped Cache Organization

Each block in the external cache has an associated 26-bit entry in the 82385's internal cache directory. This entry has three components: a 17-bit "tag," a "tag valid" bit, and eight "line valid" bits. The tag acts as a main memory page number (17 tag bits support 2^{17} pages). For example, if line 9 of page 2 currently resides in the cache, then a binary 2 is stored in the Set 1 tag field. (For any 82385 direct mapped cache page in main memory, Set 0 consists of lines 0–7, Set 1 consists of lines 8–15, etc. Line 9 is shaded in Figure 2-1.) An important characteristic of a direct mapped cache is that line 9 of any page can only reside in line 9 of the cache. All identical page offsets map to a single cache location.

The data in a cache set is considered valid or invalid depending on the status of its tag valid bit. If clear, the entire set is considered invalid. If true, an individual line within the set is considered valid or invalid depending on the status of its line valid bit.

The 82385 sees the 80386 address bus (A2–A31) as partitioned into three fields: a 17-bit "tag" field (A15–A31), a 10-bit "set-address" field (A5–A14), and a 3-bit "line select" field (A2–A4). (See Figure 2-2.) The lower 13 address bits (A2–A14) also serve as the "cache address" which directly selects one of 8K doublewords in the external cache.

2.1.2 Direct Mapped Cache Operation

The following is a description of the interaction between the 80386, cache, and cache directory.

2.1.2.1 READ HITS

When the 80386 initiates a memory read cycle, the 82385 uses the 10-bit set address to select one of 1024 directory entries, and the 3-bit line select field to select one of eight line valid bits within the entry. The 13-bit cache address selects the corresponding doubleword in the cache. The 82385 compares the 17-bit tag field (A15–A31 of the 80386 access) with the tag stored in the selected directory entry. If the tag and upper address bits match, and if both the tag and appropriate line valid bits are set, the result is a hit, and the 82385 directs the cache to drive the selected doubleword onto the 80386 data bus. A read hit does not alter the contents of the cache or directory.

2.1.2.2 READ MISSES

A read miss can occur in two ways. The first is known as a "line" miss, and occurs when the tag and upper address bits match and the tag valid bit is set, but the line valid bit is clear. The second is called a "tag" miss, and occurs when either the tag and upper address bits do not match, or the tag valid bit is clear. (The line valid bit is a "don't care" in a tag miss.) In both cases, the 82385 forwards the 80386 reference to the system, and as the returning data is fed to the 80386, it is written into the cache and validated in the cache directory.

In a line miss, the incoming data is validated simply by setting the previously clear line valid bit. In a tag miss, the upper address bits overwrite the previously stored tag, the tag valid bit is set, the appropriate line valid bit is set, and the other seven line valid bits are cleared. Subsequent tag hits with line misses will only set the appropriate line valid bit. (Any data associated with the previous tag is no longer considered resident in the cache.)

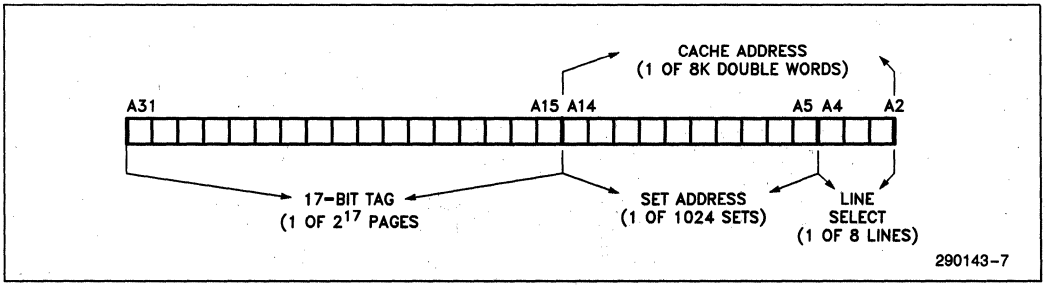


Figure 2-2. 80386 Address Bus Bit Fields—Direct Mapped Organization

2.1.2.3 OTHER OPERATIONS THAT AFFECT THE CACHE AND CACHE DIRECTORY

The other operations that affect the cache and/or directory are write hits, snoop hits, cache flushes, and 82385 resets. In a write hit, the cache is updated along with main memory, but the directory is unaffected. In a snoop hit, the cache is unaffected, but the affected line is invalidated by clearing its line valid bit in the directory. Both an 82385 reset and cache flush clear all tag valid bits.

When an 80386/82385 system "wakes up" upon reset, all tag valid bits are clear. At this point, a read miss is the only mechanism by which main memory data is copied into the cache and validated in the cache directory. Assume an early 80386 code access seeks (for the first time) line 9 of page 2. Since the tag valid bit is clear, the access is a tag miss, and the data is fetched from main memory. Upon return, the data is fed to the 80386 and simultaneously written into line 9 of the cache. The set directory entry is updated to show this line as valid. Specifically, the tag and appropriate line valid bits are set, the remaining seven line valid bits cleared, and a binary 2 written into the tag. Since code is sequential in nature, the 80386 will likely next want line 10 of page 2, then line 11, and so on. If the 80386 sequentially fetches the next six lines, these fetches will be line misses, and as each is fetched from main memory and written into the cache, its corresponding line valid bit is set. This is the basic flow of events that fills the cache with valid data. Only after a piece of data has been copied into the cache and validated can it be accessed in a zero wait state read hit. Also, a cache entry must have been validated before it can be subsequently altered by a write hit, or invalidated by a snoop hit.

An extreme example of "thrashing" is if line 9 of page two is an instruction to jump to line 9 of page one, which is an instruction to jump back to line 9 of page two. Thrashing results from the direct mapped cache characteristic that all identical page offsets map to a single cache location. In this example, the page one access overwrites the cached page two data, and the page two access overwrites the cached page one data. As long as the code jumps back and forth the hit rate is zero. This is of course an extreme case. The effect of thrashing is that a direct mapped cache exhibits a slightly reduced overall hit rate as compared to a set associative cache of the same size.

2.2 TWO WAY SET ASSOCIATIVE CACHE

2.2.1 Two Way Set Associative Cache Structure and Terminology

Figure 2-3 illustrates the relationship between the directory, cache, and 4 Gigabyte address space.

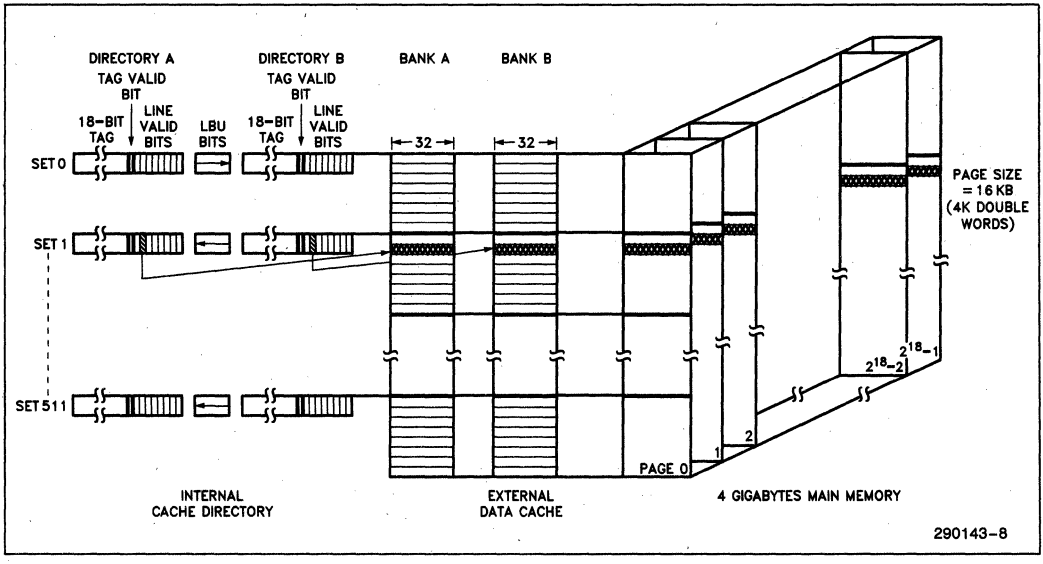


Figure 2-3. Two-Way Set Associative Cache Organization

Whereas the direct mapped cache is organized as one bank of 8K doublewords, the two way set associative cache is organized as two banks (A and B) of 4K doublewords each. The page size is halved, and the number of pages doubled. (Note the extra tag bit.) The cache now has 512 sets in each bank. (Two banks times 512 sets gives a total of 1024. The structure can be thought of as two half-sized direct mapped caches in parallel.) The performance advantage over a direct mapped cache is that all identical page offsets map to two cache locations instead of one, reducing the potential for thrashing. The 82385's partitioning of the 80386 address bus is depicted in Figure 2-4.

2.2.2 LRU Replacement Algorithm

The two way set associative directory has an additional feature: the "least recently used" or LRU bit. In the event of a read miss, either bank A or bank B will be updated with new data. The LRU bit flags the candidate for replacement. Statistically, of two blocks of data, the block most recently used is the block most likely to be needed again in the near future. By flagging the least recently used block, the 82385 ensures that the cache block replaced is the least likely to have data needed by the CPU.

2.2.3 Two Way Set Associative Cache Operation

2.2.3.1 READ HITS

When the 80386 initiates a memory read cycle, the 82385 uses the 9-bit set address to select one of 512 sets. The two tags of this set are simultaneously compared with A14–A31, both tag valid bits checked, and both appropriate line valid bits checked. If either comparison produces a hit, the corresponding cache bank is directed to drive the selected doubleword onto the 80386 data bus. (Note that both banks will never concurrently cache the same main memory location.) If the requested data resides in bank A, the LRU bit is pointed toward B. If B produces the hit, the LRU bit is pointed toward A.

2.2.3.2 READ MISSES

As in direct mapped operation, a read miss can be either a line or tag miss. Let's start with a tag miss example. Assume the 80386 seeks line 9 of page 2, and that neither the A nor B directory produces a tag match. Assume also, as indicated in Figure 2-3, that the LRU bit points to A. As the data returns from main memory, it is loaded into offset 9 of bank A. Concurrently, this data is validated by updating the set 1 directory entry for bank A. Specifically, the upper address bits overwrite the previous tag, the tag valid bit is set, the appropriate line valid bit is set, and the other seven line valid bits cleared. Since this data is the most recently used, the LRU bit is turned toward B. No change to bank B occurs.

If the next 80386 request is line 10 of page two, the result will be a line miss. As the data returns from main memory, it will be written into offset 10 of bank A (tag hit/line miss in bank A), and the appropriate line valid bit will be set. A line miss in one bank will cause the LRU bit to point to the other bank. In this example, however, the LRU bit has already been turned toward B.

2.2.3.3 OTHER OPERATIONS THAT AFFECT THE CACHE AND CACHE DIRECTORY

Other operations that affect the cache and cache directory are write hits, snoop hits, cache flushes, and 82385 resets. A write hit updates the cache along with main memory. If directory A detects the hit, bank A is updated. If directory B detects the hit, bank B is updated. If one bank is updated, the LRU bit is pointed toward the other.

If a snoop hit invalidates an entry, for example, in cache bank A, the corresponding LRU bit is pointed toward A. This ensures that invalid data is the prime candidate for replacement in a read miss. Finally, resets and flushes behave just as they do in a direct mapped cache, clearing all tag valid bits.

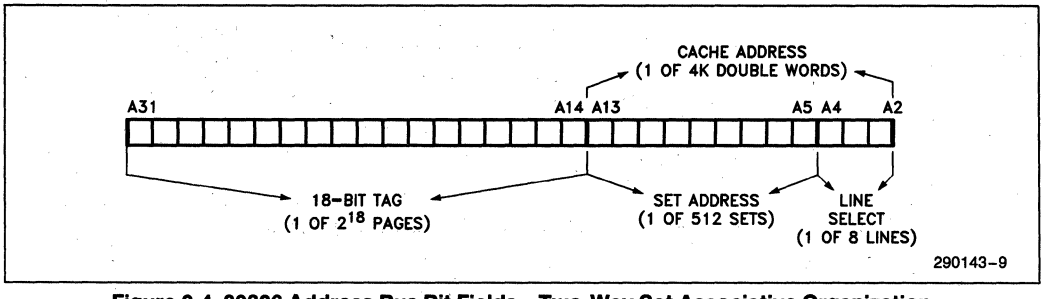


Figure 2-4. 80386 Address Bus Bit Fields—Two-Way Set Associative Organization

3.0 82385 PIN DESCRIPTION

The 82385 creates the 82385 local bus, which is a functional 80386 interface. To facilitate understanding, 82385 local bus signals go by the same name as their 80386 equivalents, except that they are preceded by the letter "B". The 82385 local bus equivalent to ADS# is BADS#, the equivalent to NA# is BNA#, etc. This convention applies to bus states as well. For example, BT1P is the 82385 local bus state equivalent to the 80386 T1P state.

3.1 80386/82385 INTERFACE SIGNALS

These signals form the direct interface between the 80386 and 82385.

3.1.1 80386/82385 Clock (CLK2)

CLK2 provides the fundamental timing for an 80386/82385 system, and is driven by the same source that drives the 80386 CLK2 input. The 82385, like the 80386, divides CLK2 by two to generate an internal "phase indication" clock. (See Figure 3-1.) The CLK2 period whose rising edge drives the internal clock low is called PHI1, and the CLK2 period that drives the internal clock high is called PHI2. A PHI1-PHI2 combination (in that order) is known as a "T" state, and is the basis for 80386 bus cycles.

3.1.2 80386/82385 Reset (RESET)

This input resets the 82385, bringing it to an initial known state, and is driven by the same source that drives the 80386 RESET input. A reset effectively flushes the cache by clearing all cache directory tag valid bits. The falling edge of RESET is synchronized to CLK2, and used by the 82385 to properly establish the phase of its internal clock. (See Figure 3-2.) Specifically, the second internal phase following the falling edge of RESET is PHI2.

3.1.3 80386/82385 Address Bus (A2–A31), Byte Enables (BE0#–BE3#), and Cycle Definition Signals (M/IO#, D/C#, W/R#, LOCK#)

The 82385 directly connects to these 80386 outputs. The 80386 address bus is used in the cache directory comparison to see if data referenced by 80386 resides in the cache, and the byte enables inform the 82385 as to which portions of the data bus are involved in an 80386 cycle. The cycle definition signals are decoded by the 82385 to determine the type of cycle the 80386 is executing.

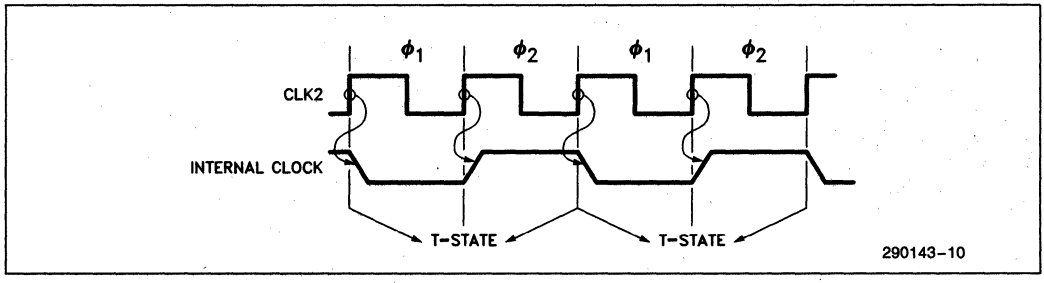


Figure 3-1. CLK2 and Internal Clock

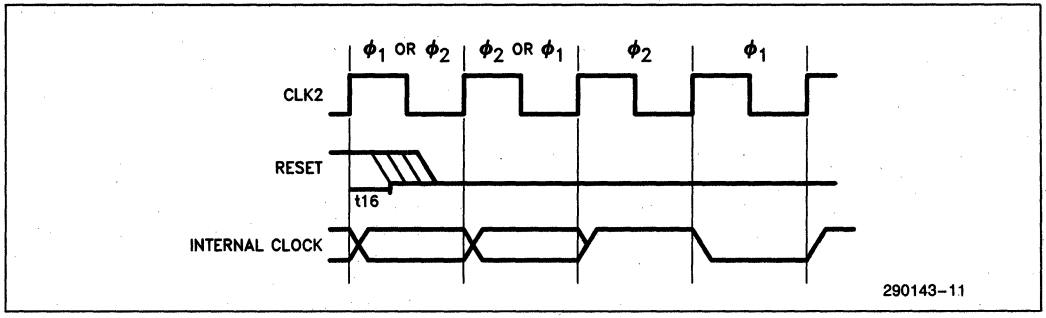


Figure 3-2. Reset/Internal Phase Relationship

3.1.4 80386/82385 Address Status (ADS#) and Ready Input (READYI#)

ADS# is an 80386 output, and tells the 82385 that new address and cycle definition information is available. READYI# is an input to both the 80386 (via the 80386 READY# input pin) and 82385, that indicates the completion of an 80386 bus cycle. ADS# and READYI# are used to keep track of the 80386 bus state.

3.1.5 80386 Next Address Request (NA#)

This 82385 output controls the pipelining of the 80386. It can be tied directly to the 80386 NA# input, or it can be logically "AND"ed with other 80386 local bus next address requests.

3.1.6 Ready Output (READYO#) and Bus Ready Enable (BRDYEN#)

The 82385 directly terminates all but two types of 80386 bus cycles with its READYO# output. 80386 local bus cycles must be terminated by the local device being accessed. This includes devices decoded using the 82385 LBA# signal and 80387 accesses.

The other cycles not directly terminated by the 82385 are 82385 local bus reads, specifically cache read misses and non-cacheable reads. (Recall that the 82385 forwards and runs such cycles on the 82385 bus.) In these cycles the signal that terminates the 82385 local bus access is BREADY#, which is gated through to the 80386 local bus such that the 80386 and 82385 local bus cycles are concurrently terminated. BRDYEN# is used to gate the BREADY# signal to the 80386.

3.2 CACHE CONTROL SIGNALS

These 82385 outputs control the external 32KB cache data memory.

3.2.1 Cache Address Latch Enable (CALEN)

This signal controls the latch (typically an F or AS series 74373) that resides between the low order 80386 address bits and the cache SRAM address inputs. (The outputs of this latch are the "cache address" described in the previous chapter.) When CALEN is high the latch is transparent. The falling edge of CALEN latches the current inputs which remain applied to the cache data memory until CALEN returns to an active high state.

3.2.2 Cache Transmit/Receive (CT/R#)

This signal defines the direction of an optional data transceiver (typically an F or AS series 74245) between the cache and 80386 data bus. When high, the transceiver is pointed towards the 80386 local data bus (the SRAMs are output enabled). When low, the transceiver points towards the cache data memory. A transceiver is required if the cache is designed with SRAMs that lack an output enable control. A transceiver may also be desirable in a system that has a heavily loaded 80386 local data bus. These devices are not necessary when using SRAMs which incorporate an output enable.

3.2.3 Cache Chip Selects (CS0# – CS3#)

These active low signals tie to the cache SRAM chip selects, and individually enable the four bytes of the 32-bit wide cache. CS0# enables D0–D7, CS1# enables D8–D15, CS2# enables D16–D23, and CS3# enables D24–D31. During read hits, all four bytes are enabled regardless of whether or not all four 80386 byte enables are active. (The 80386 ignores what it did not request.) Also, all four cache bytes are enabled in a read miss so as to update the cache with a complete line (double word). In a write hit, only those cache bytes that correspond to active byte enables are selected. This prevents cache data from being corrupted in a partial doubleword write.

3.2.4 Cache Output Enables (COEA#, COEB#) and Write Enables (CWEA#, CWEB#)

COEA# and COEB# are active low signals which tie to the cache SRAM output enables and respectively enable cache bank A or B to drive the 80386 data bus. In a two-way set associative cache, either COEA# or COEB# is active during a read hit, depending on which bank is selected. In a direct mapped cache, both are activated, so the designer is free to use either one.

CWEA# and CWEB# are active low signals which tie to the cache SRAM write enables, and respectively enable cache bank A or B to receive data from the 80386 data bus (80386 write hit or read miss update). In a two-way set associative cache, one or the other is enabled in a read miss or write hit. In a direct mapped cache, both are activated, so the designer is free to use either one.

If the cache is implemented with SRAMs that do not have output enables, then a transceiver between the cache memory and 80386 data bus is required. In this case, the output enable of each bank must be "AND"ed with the corresponding write enable to provide the transceiver enable signal. For example, COEA# and CWEA# are "AND"ed to enable the transceiver between cache bank A and the 80386 data bus (chapter 4, Figures 4-4B and 4-4D). The various cache configurations supported by the 82385 are described in chapter 4.

3.3 80386 LOCAL BUS DECODE INPUTS

These 82385 inputs are generated by decoding the 80386 address and cycle definition lines. These active low inputs are sampled at the end of the first state in which the address of a new 80386 cycle becomes available (T1 or first T2P). The signals must be kept stable during the entire time the address is valid. They are not internally latched by the 82385.

3.3.1 80386 Local Bus Access (LBA#)

This input identifies an 80386 access as directed to a resource (other than the cache) on the 80386 local bus. (The 80387 Numerics Coprocessor is considered an 80386 local bus resource, but LBA# need not be generated as the 82385 internally decodes 80387 accesses.) The 82385 simply ignores these cycles. They are neither forwarded to the system nor do they affect the cache or cache directory. Note that LBA# has priority over all other types of cycles. If LBA# is asserted, the cycle is interpreted as an 80386 local bus access, regardless of the cycle type or status of NCA# or X16#. This allows any 80386 cycle (memory, I/O, interrupt acknowledge, etc.) to be kept on the 80386 local bus if desired.

3.3.2 Non-Cacheable Access (NCA#)

This active low input identifies an 80386 cycle as non-cacheable. The 82385 forwards non-cacheable cycles to the 82385 local bus and runs them. The cache and cache directory are unaffected.

NCA# allows a designer to set aside a portion of main memory as non-cacheable. Potential applications include memory-mapped I/O and systems where multiple masters access dual ported memory via different busses.

3.3.3 16-Bit Access (X16#)

X16# is an active low input which identifies 16-bit memory and/or I/O space, and the decoded signal that drives X16# should also drive the 80386 BS16# input. 16-bit accesses are treated like non-cacheable accesses: they are forwarded to and executed on the 82385 local bus with no impact on the cache or cache directory. In addition, the 82385 locks the two halves of a non-aligned 16-bit transfer from interruption by another master, as does the 80386.

3.4 82385 LOCAL BUS INTERFACE SIGNALS

The 82385 presents an "80386-like" front end to the system, and the signals discussed in this section are 82385 local bus equivalents to actual 80386 signals. These signals are named with respect to their 80386 counterparts, but with the letter "B" appended to the front.

Note that the 82385 itself does not have equivalent output signals to the 80386 data bus (D0–D31), address bus (A2–A31), and cycle definition signals (M/IO#, D/C#, W/R#). The 82385 data bus (BD0–BD31) is actually the system side of a latching transceiver, and the 82385 address bus and cycle definition signals (BA2–BA31, BM/IO#, BD/C#, BW/R#) are the outputs of an edge-triggered latch. The signals that control this data transceiver and address latch are discussed in section 3.5.

3.4.1 82385 Bus Byte Enables (BBE0#–BBE3#)

BBE0#–BBE3# are the 82385 local bus equivalents to the 80386 byte enables. In a cache read miss, the 82385 drives all four signals low, regardless of whether or not all four 80386 byte enables are active. This ensures that a complete line (doubleword) is fetched from main memory for the cache update. In all other 82385 local bus cycles, the 82385 duplicates the logic levels of the 80386 byte enables. The 82385 tri-states these outputs when it is not the current bus master.

3.4.2 82385 Bus Lock (BLOCK#)

BLOCK# is the 82385 local bus equivalent to the 80386 LOCK# output, and distinguishes between locked and unlocked cycles. When the 80386 runs a locked sequence of cycles (and LBA# is negated), the 82385 forwards and runs the sequence on the 82385 local bus, regardless of whether any locations

referenced in the sequence reside in the cache. A read hit will be run as if it is a read miss, but a write hit will update the cache as well as being completed to system memory. In keeping with 80386 behavior, the 82385 does not allow another master to interrupt the sequence. BLOCK# is tri-stated when the 82385 is not the current bus master.

3.4.3 82385 Bus Address Status (BADS#)

BADS# is the 82385 local bus equivalent of ADS#, and indicates that a valid address (BA2–BA31, BBE0#–BBE3#) and cycle definition (BM/IO#, BW/R#, BD/C#) is available. It is asserted in BT1 and BT2P states, and is tri-stated when the 82385 does not own the bus.

3.4.4 82385 Bus Ready Input (BREADY#)

82385 local bus cycles are terminated by BREADY#, just as 80386 cycles are terminated by the 80386 READY# input. In 82385 local bus read cycles, BREADY# is gated by BRDYEN# onto the 80386 local bus, such that it terminates both the 80386 and 82385 local bus cycles.

3.4.5 82385 Bus Next Address Request (BNA#)

BNA# is the 82385 local bus equivalent to the 80386 NA# input, and indicates that the system is prepared to accept a pipelined address and cycle definition. If BNA# is asserted and the new cycle information is available, the 82385 begins a pipelined cycle on the 82385 local bus.

3.5 82385 BUS DATA TRANSCEIVER AND ADDRESS LATCH CONTROL SIGNALS

The 82385 data bus is the system side of a latching transceiver (typically an F or AS series 74646), and the 82385 address bus and cycle definition signals are the outputs of an edge-triggered latch (F or AS series 74374). The following is a discussion of the 82385 outputs that control these devices. An important characteristic of these signals and the devices they control is that they ensure that BD0–BD31, BA2–BA31, BM/IO#, BD/C#, and BW/R# reproduce the functionality and timing behavior of their 80386 equivalents.

3.5.1 Local Data Strobe (LDSTB), Data Output Enable (DOE#), and Bus Transmit/Receive (BT/R#)

These signals control the latching data transceiver. BT/R# defines the transceiver direction. When high, the transceiver drives the 82385 data bus in write cycles. When low, the transceiver drives the 80386 data bus in 82385 local bus read cycles. DOE# enables the transceiver outputs.

The rising edge of LDSTB latches the 80386 data bus in all write cycles. The interaction of this signal and the latching transceiver is used to perform the 82385's posted write capability.

3.5.2 Bus Address Clock Pulse (BACP) and Bus Address Output Enable (BAOE#)

These signals control the latch that drives BA2–BA31, BM/IO#, BW/R#, and BD/C#. In any 80386 cycle that is forwarded to the 82385 local bus, the rising edge of BACP latches the 80386 address and cycle definition signals. BAOE# enables the latch outputs when the 82385 is the current bus master and disables them otherwise.

3.6 STATUS AND CONTROL SIGNALS

3.6.1 Cache Miss Indication (MISS#)

This output accompanies cacheable read and write miss cycles. This signal transitions to its active low state when the 82385 determines that a cacheable 80386 access is a miss. Its timing behavior follows that of the 82385 local bus cycle definition signals (BM/IO#, BD/C#, BW/R#) so that it becomes available with BADS# in BT1 or the first BT2P. MISS# is floated when the 82385 does not own the bus, such that multiple 82385's can share the same node in multi-cache systems. (As discussed in Chapter 7, this signal also serves a reserved function in testing the 82385.)

3.6.2 Write Buffer Status (WBS)

The latching data transceiver is also known as the "posted write buffer." WBS indicates that this buffer contains data that has not yet been written to the system even though the 80386 may have begun its next cycle. It is activated when 80386 data is

latched, and deactivated when the corresponding 82385 local bus write cycle is completed (BREADY#). (As discussed in Chapter 7, this signal also serves a reserved function in testing the 82385.)

WBS can serve several functions. In multi-processor applications, it can act as a coherency mechanism by informing a bus arbiter that it should let a write cycle run on the system bus so that main memory has the latest data. If any other 82385 cache subsystems are on the bus, they will monitor the cycle via their bus watching mechanisms. Any 82385 that detects a snoop hit will invalidate the corresponding entry in its local cache.

3.6.3 Cache Flush (FLUSH)

When activated, this signal causes the 82385 to clear all of its directory tag valid bits, effectively flushing the cache. (As discussed in Chapter 7, this signal also serves a reserved function in testing the 82385.) The primary use of the FLUSH input is for diagnostics and multi-processor support. The use of this pin as a coherency mechanism may impact software transparency.

The FLUSH input must be held active for at least 4 CLK (8 CLK2) cycles to complete the flush sequence. If FLUSH is still active after 4 CLK cycles, any accesses to the cache will be misses and the cache will not be updated (since FLUSH is active).

3.7 BUS ARBITRATION SIGNALS (BHOLD AND BHLDA)

In master mode, BHOLD is an input that indicates a request by a slave device for bus ownership. The 82385 acknowledges this request via its BHLDA output. (These signals function identically to the 80386 HOLD and HLDA signals.)

The roles of BHOLD and BHLDA are reversed for an 82385 in slave mode. BHOLD is now an output indicating a request for bus ownership, and BHLDA an input indicating that the request has been granted.

3.8 COHERENCY (BUS WATCHING) SUPPORT SIGNALS (SA2–SA31, SSTB#, SEN)

These signals form the 82385's bus watching interface. The Snoop Address Bus (SA2–SA31) connects to the system address lines if masters reside at both the system and 82385 local bus levels, or the 82385 local bus address lines if masters reside only at the 82385 local bus level. Snoop Strobe (SSTB#) indicates that a valid address is on the snoop address inputs. Snoop Enable (SEN) indicates that the cycle is a write. In a system with masters only at the 82385 local bus level, SA2–SA31, SSTB#, and SEN can be driven respectively by BA2–BA31, BADS#, and BW/R# without any support circuitry.

3.9 CONFIGURATION INPUTS (2W/D#, M/S#)

These signals select the configurations supported by the 82385. They are hardware strap options and must not be changed dynamically. 2W/D# (2-Way/Direct Mapped Select) selects a two-way set associative cache when tied high, or a direct mapped cache when tied low. M/S# (Master/Slave Select) chooses between master mode (M/S# high) and slave mode (M/S# low).

4.0 80386 LOCAL BUS INTERFACE

The following is a detailed description of how the 82385 interfaces to the 80386 and to 80386 local bus resources. Items specifically addressed are the interfaces to the 80386, the cache SRAMs, and the 80387 Numerics Coprocessor.

The many timing diagrams in this and the next chapter provide insight into the dual pipelined bus structure of an 80386/82385 system. It's important to realize, however, that one need not know every possible cycle combination to use the 82385. The interface is simple, and the dual bus operation invisible to the 80386 and system. To facilitate discussion of the timing diagrams, several conventions have been adopted. Refer to Figure 4-2A, and note that 80386 bus cycles, 80386 bus states, and 82385 bus states are identified along the top. All states can be identified by the "frame numbers" along the bottom. The cycles in Figure 4-2A include a cache read hit (CRDH), a cache read miss (CRDM), and a write (WT). WT represents any write, cacheable or not. When necessary to distinguish cacheable writes, a write hit goes by CWTH and a write miss by CWTM. Non-cacheable system reads go by SBRD. Also, it is assumed that system bus pipelining occurs even though the BNA# signal is not shown. When the system pipeline begins is a function of the system bus controller.

80386 bus cycles can be tracked by ADS# and READYI#, and 82385 cycles by BADS# and BREADY#. These four signals are thus a natural

choice to help track parallel bus activity. Note in the timing diagrams that 80386 cycles are numbered using ADS# and READYI#, and 82385 cycles using BADS# and BREADY#. For example, when the address of the first 80386 cycle becomes available, the corresponding assertion of ADS# is marked "1", and the READYI# pulse that terminates the cycle is marked "1" as well. Whenever an 80386 cycle is forwarded to the system, its number is forwarded as well so that the corresponding 82385 bus cycle can be tracked by BADS# and BREADY#.

The "N" value in the timing diagrams is the assumed number of main memory wait states inserted in a non-pipelined 80386 bus cycle. For example, a non-pipelined access to N=2 memory requires a total of four bus states, while a pipelined access requires three. (The pipeline advantage effectively hides one main memory wait state.)

4.1 PROCESSOR INTERFACE

This section presents the 80386/82385 hardware interface and discusses the interaction and timing of this interface. Also addressed is how to decode the 80386 address bus to generate the 82385 inputs LBA#, NCA#, and X16#. (Recall that LBA# allows memory and/or I/O space to be set aside for 80386 local bus resources; NCA# allows system memory to be set aside as non-cacheable; and X16# allows system memory and/or I/O space to be reserved for 16-bit resources.) Finally, the 82385's handling of 16-bit space is discussed.

4.1.1 Hardware Interface

Figure 4-1 is a diagram of an 80386/82385 system, which can be thought of as three distinct interfaces. The first is the 80386/82385 interface (including the Ready Logic). The second is the cache interface, as depicted by the cache control bus in the upper left corner of Figure 4-1. The third is the 82385 bus interface, which includes both direct connects and signals that control the 74374 address/cycle definition latch and 74646 latching data transceiver. (The 82385 bus interface is the subject of the next chapter).

As seen in Figure 4-1, the 80386/82385 interface is a straightforward connection. The only necessary support logic is that required to sum all ready sources.

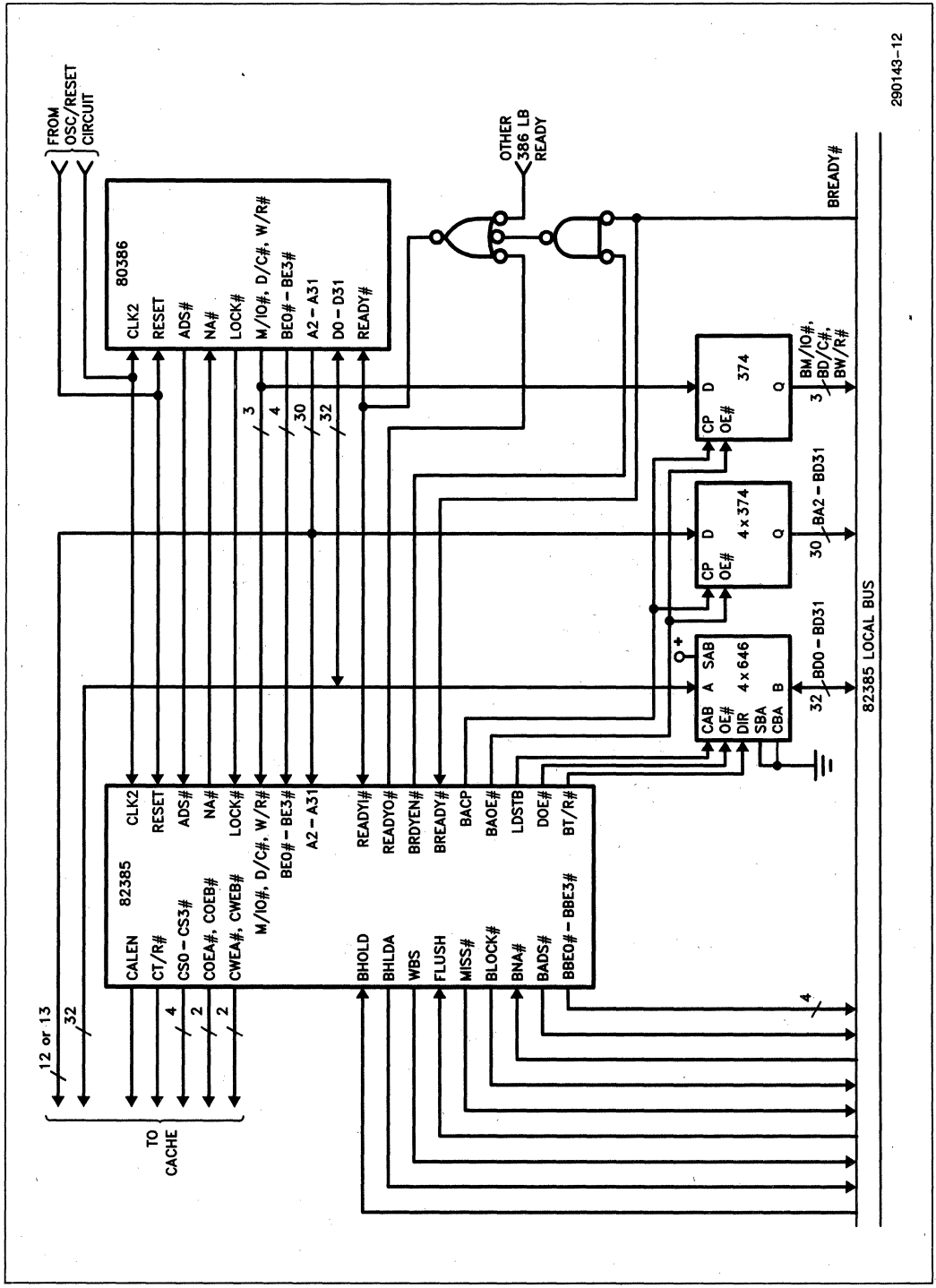


Figure 4-1. 80386/82385 Interface

4.1.2 Ready Generation

Note in Figure 4-1 that the ready logic consists of two gates. The upper three-input AND gate (shown as a negative logic OR) sums all 80386 local bus ready sources. One such source is the 82385 READYO# output, which terminates read hits and posted writes. The output of this gate drives the 80386 READY# input and is monitored by the 82385 (via READYI#) to track the 80386 bus state.

When the 82385 forwards an 80386 read cycle to the 82385 bus (cache read miss or non-cacheable read), it does not directly terminate the cycle via READYO#. Instead, the 80386 and 82385 bus cycles are concurrently terminated by a system ready source. This is the purpose of the additional two-input OR gate (negative logic AND) in Figure 4-1. When the 82385 forwards a read to the 82385 bus, it asserts BRDYEN# which enables the system ready signal (BREADY#) to directly terminate the 80386 bus cycle.

Figures 4-2A and 4-2B illustrate the behavior of the signals involved in ready generation. Note in cycle 1 of Figure 4-2A that the 82385 READYO# directly terminates the hit cycle. In cycle 2, READYO# is not activated. Instead the 82385 BRDYEN# is activated in BT2, BT2P, or BT2I states such that BREADY# can concurrently terminate the 80386 and 82385 bus cycles (frame 6). Cycle 3 is a posted write. The write data becomes available in T1P (frame 7), and

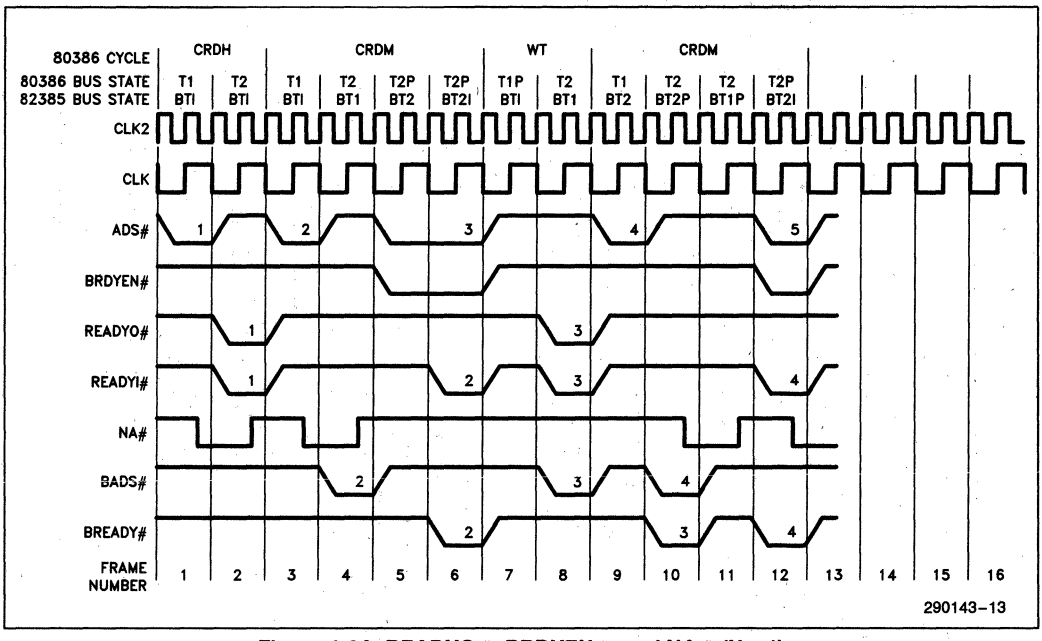


Figure 4-2A. READYO#, BRDYEN#, and NA# (N = 1)

the address, data, and cycle definition of the write are latched in T2 (frame 8). The 80386 cycle is terminated by READYO# in frame 8 with no wait states. The 82385, however, sees the write cycle through to completion on the 82385 bus where it is terminated in frame 10 by BREADY#. In this case, the BREADY# signal is not gated through to the 80386. Refer to Figures 4-2A and 4-2B for clarification.

4.1.3. NA# and 80386 Local Bus Pipelining

Cycle 1 of Figure 4-2A is a typical cache read hit. The 80386 address becomes available in T1, and the 82385 uses this address to determine if the referenced data resides in the cache. The cache look-up is completed and the cycle qualified as a hit or miss in T1. If the data resides in the cache, the cache is directed to drive the 80386 data bus, and the 82385 drives its READYO# output so the cycle can be terminated at the end of the first T2 with no wait states.

Although cycle 2 starts out like cycle 1, at the end of T1 (frame 3), it is qualified as a miss and forwarded to the 82385 bus. The 82385 bus cycle begins one state after the 80386 bus cycle, implying a one wait state overhead associated with cycle 2 due to the look-up. When the 82385 encounters the miss, it immediately asserts NA#, which puts the 80386 into pipelined mode. Once in pipelined mode, the 82385 is able to qualify an 80386 cycle using the 80386 pipelined address and control signals. The result is that the cache look-up state is hidden in all but the first of a contiguous sequence of read misses. This is shown in the first two cycles, both read misses, of Figure 4-2B. The CPU sees the look-up state in the first cycle, but not in the second. In fact, the second miss requires a total of only two states, as not only does 80386 pipelining hide the look-up state, but system pipelining hides one of the main memory wait states. (System level pipelining via BNA# is discussed in the next chapter.) Several characteristics of the 82385's pipelining of the 80386 are as follows:

— The above discussion applies to all system reads, not just cache read misses.

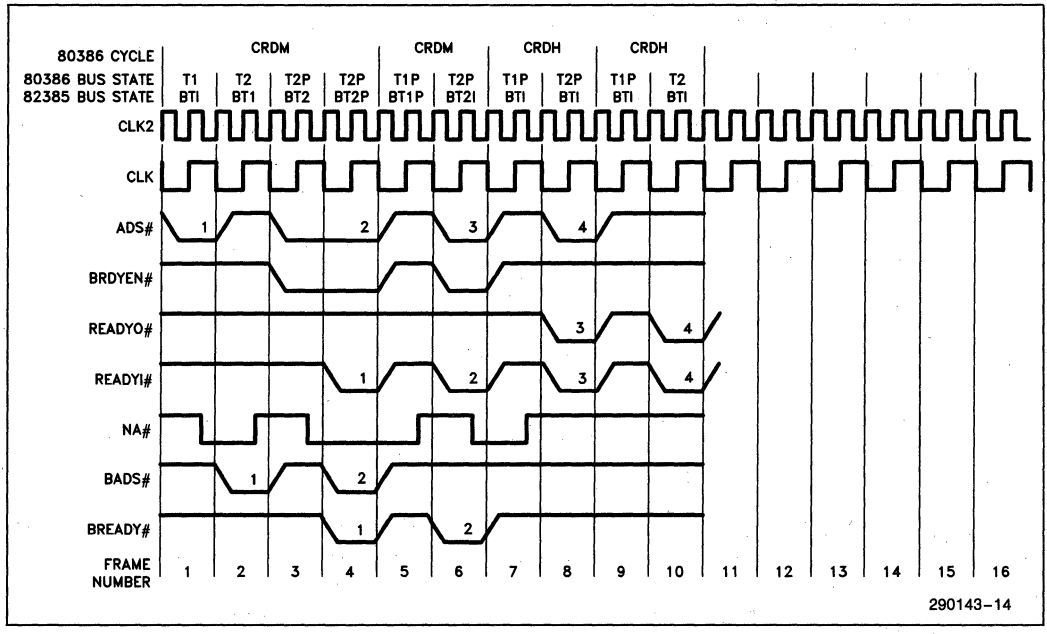


Figure 4-2B. READYO#, BRDYEN#, and NA# (N = 1)

— The 82385 provides the fastest possible switch to pipelining, T1-T2-T2P. The exception to this is when a system read follows a posted write. In this case, the sequence is T1-T2-T2-T2P. (Refer to cycle 4 of Figure 4-2A.) The number of T2 states is dependent on the number of main memory wait states.

— Refer to the read hit in Figure 4-2A (cycle 1), and note that NA# is actually asserted before the end of T1, before the hit/miss decision is made. This is of no consequence since even though NA# is sampled active in T2, the activation of READYO# in the same T2 renders NA# a "don't care." NA# is asserted in this manner to meet 80386 timing requirements and to ensure the fastest possible switch to pipelined mode.

— All read hits and the majority of writes can be serviced by the 82385 with zero wait states in non-pipelined mode, and the 82385 accordingly attempts to run all such cycles in non-pipelined mode. An exception is seen in the hit cycles (cycles 3 and 4) of Figure 4-2B. The 82385 does not know soon enough that cycle 3 is a hit, and thus sustains the pipeline. The result is that three sequential hits are required before the 80386 is totally out of pipelined mode. (The three hits look like T1P-T2P, T1P-T2, T1-T2.) Note that this does not occur if the number of main memory wait states is equal to or greater than two.

As far as the design is concerned, NA# is generally tied directly to the 80386 NA# input. However, other local NA# sources may be logically "AND"ed with the 82385 NA# output if desired. It is essential, however, that no device other than the 82385 drive the 80386 NA# input unless that device resides on the 80386 local bus in space decoded via LBA#. If desired, the 82385 NA# output can be ignored and the 80386 NA# input tied high. The 80386 NA# input should never be tied low, which would always keep it active.

4.1.4 LBA#, NCA#, and X16# Generation

The 82385 input signals LBA# and X16# are generated by decoding the 80386 address (A2–A31) and cycle definition (W/R#, D/C#, M/IO#) lines. The 82385 samples them at the end of the first state in which they become available, which is either T1 or the first T2P cycle. The decode configuration and timings are illustrated respectively in Figures 4-3A and 4-3B.

The 82385 samples NCA# in T1P, or the first T2, or the second T2P.

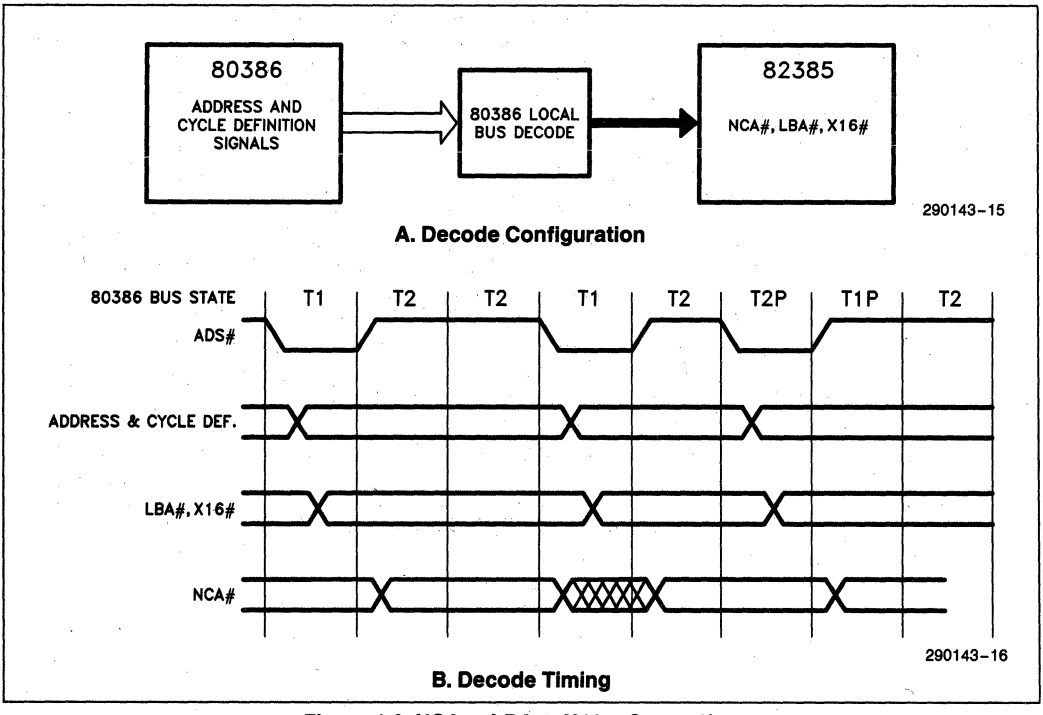


Figure 4-3. NCA#, LBA#, X16# Generation

4.1.5 82385 Handling of 16-Bit Space

As discussed previously, the 82385 does not cache devices decoded as 16-bit. Instead it makes provision to accommodate 16-bit space as non-cacheable via the X16# input. X16# is generated when the user decodes the 80386 address and cycle definition lines for the BS16# input of the 80386 (Figure 4-3). The decode output now drives both the 80386 BS16# input and the 82385 X16# input. Cycles decoded this way are treated as non-cacheable. They are forwarded to and executed on the 82385 bus, but have no impact on the cache or cache directory. The 82385 also monitors the 80386 byte enables in a 16-bit cycle to see if an additional cycle is required to complete the transfer. Specifically, a second cycle is required if (BE0# OR BE1#) AND (BE2# OR BE3#) is asserted in the current cycle. The 82385, like the 80386, will not allow the two halves of a 16-bit transfer to be interrupted by another master.

There is an important distinction between the handling of 16-bit space in an 80386 system with an 82385 as compared to a system without an 82385. The 80386 BS16# input need not be asserted until the last state of a 16-bit cycle for the 80386 to recognize it as such. The 82385, however, needs the information earlier, specifically at the end of the first 80386 bus state (T1 or first T2P) in which the address of the 16-bit cycle becomes available. The result is that in a system without an 82385, 16-bit devices can define themselves as 16-bit devices "on the fly," while in a system with an 82385, 16-bit devices should be located in space set aside for 16-bit devices via the X16# decode.

4.2 CACHE INTERFACE

The following is a description of the external data cache and 82385 cache interface.

4.2.1 Cache Configurations

The 82385 controls the cache memory via the control signals shown in Figure 4-1. These signals drive one of four possible cache configurations, as depicted in Figures 4-4A through 4-4D. Figure 4-4A shows a direct mapped cache organized as 8K doublewords. The likely design choice is four 8K x 8 SRAMs. Figure 4-4B depicts the same cache memory but with a data transceiver between the cache and 80386 data bus. In this configuration, CT/R# controls the transceiver direction, and COEA# drives the transceiver output enable. (COEB# could also be used.) A data buffer is required if the chosen SRAM does not have a separate output enable. Additionally, buffers may be used to ease SRAM timing requirements or in a system with a heavily loaded data bus. (Guidelines for SRAM selection are included in Chapter 6.)

Figure 4-4C depicts a two-way set associative cache organized as two banks (A and B) of 4K doublewords each. The likely design choice is sixteen 4K x 4 SRAMs. Finally, Figure 4-4D depicts the two-way organization with data buffers between the cache memory and data bus.

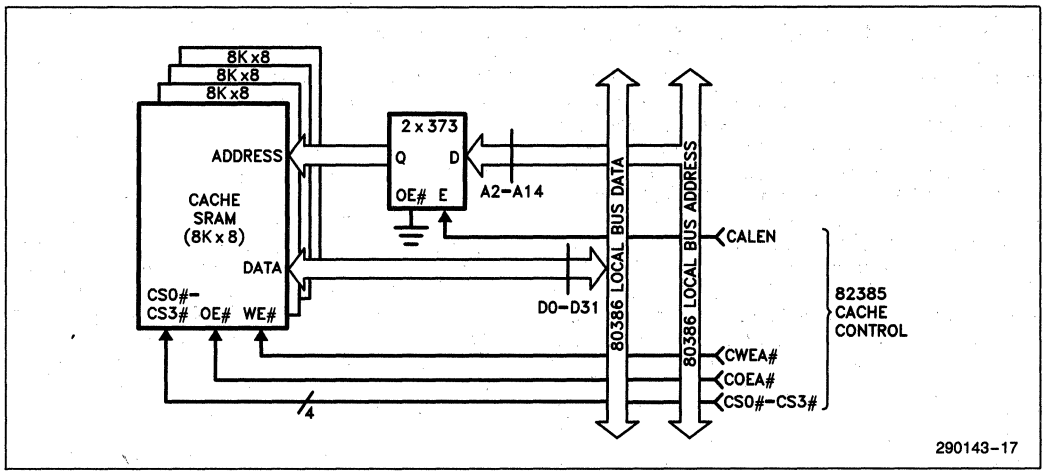


Figure 4-4A. Direct Mapped Cache without Data Buffers

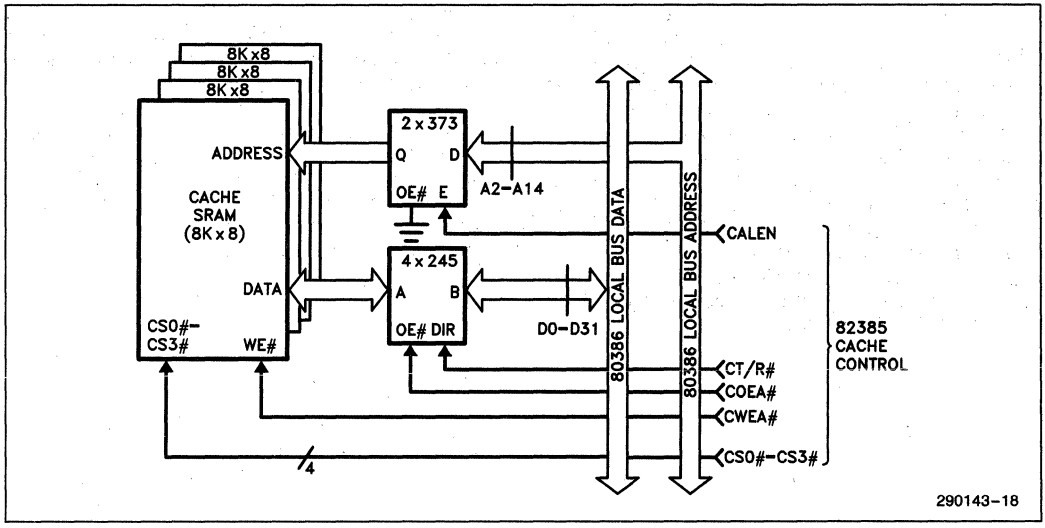


Figure 4-4B. Direct Mapped Cache with Data Buffers

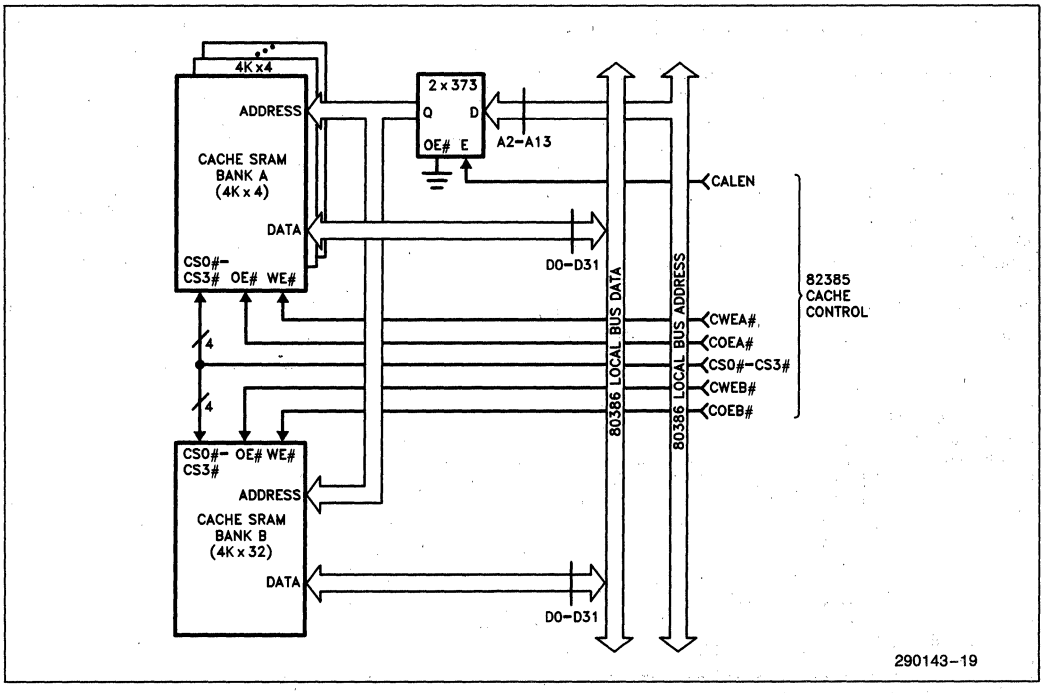


Figure 4-4C. Two-Way Set Associative Cache without Data Buffers

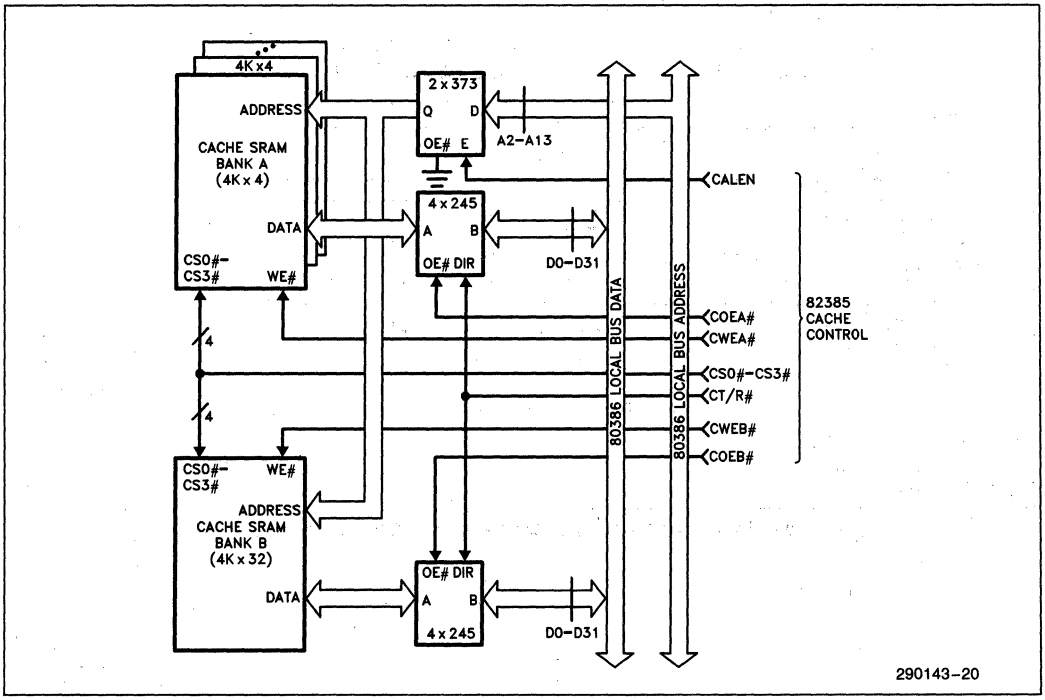


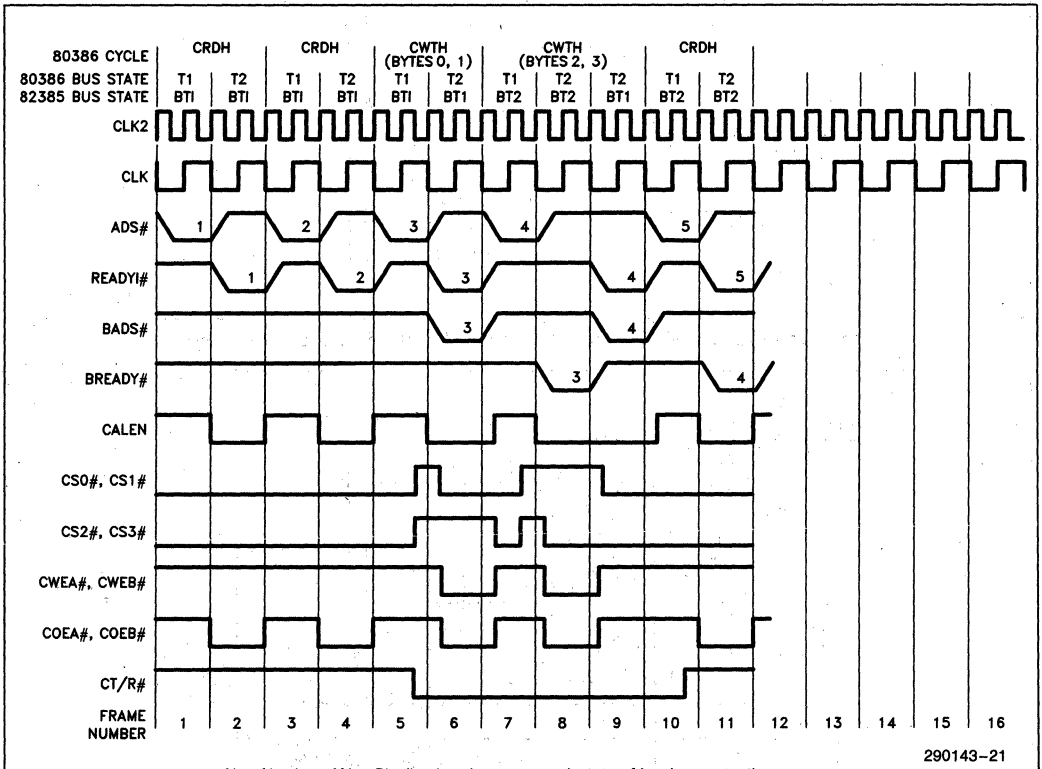
Figure 4-4D. Two-Way Set Associative Cache with Data Buffers

4.2.2 Cache Control—Direct Mapped

Figure 4-5A illustrates the timing of cache read and write hits, while Figure 4-5B illustrates cache updates. In a read hit, the cache output enables are driven from the beginning of T2 (cycle 1 of Figure 4-5A). If at the end of T1 the cycle is qualified as a cacheable read, the output enables are asserted on the assumption that the cycle will be a hit. (Driving the output enables before the actual hit/miss decision is made eases SRAM timing requirements.)

Cycle 1 of Figure 4-5B illustrates what happens when the assumption of a hit turns out to be wrong. Note that the output enables are asserted at the beginning of T2, but then disabled at the end of T2. Once the output enables are inactive, the 82385 turns the transceiver around (via CT/R#) and drives the write enables to begin the cache update cycle. Note in Figure 4-5B that once the 80386 is in pipelined mode, the output enables need not be driven prior to a hit/miss decision, since the decision is made earlier via the pipelined address information.

One consequence of driving the output enables low in a miss before the hit/miss decision is made is that since the cache starts driving the 80386 data bus,



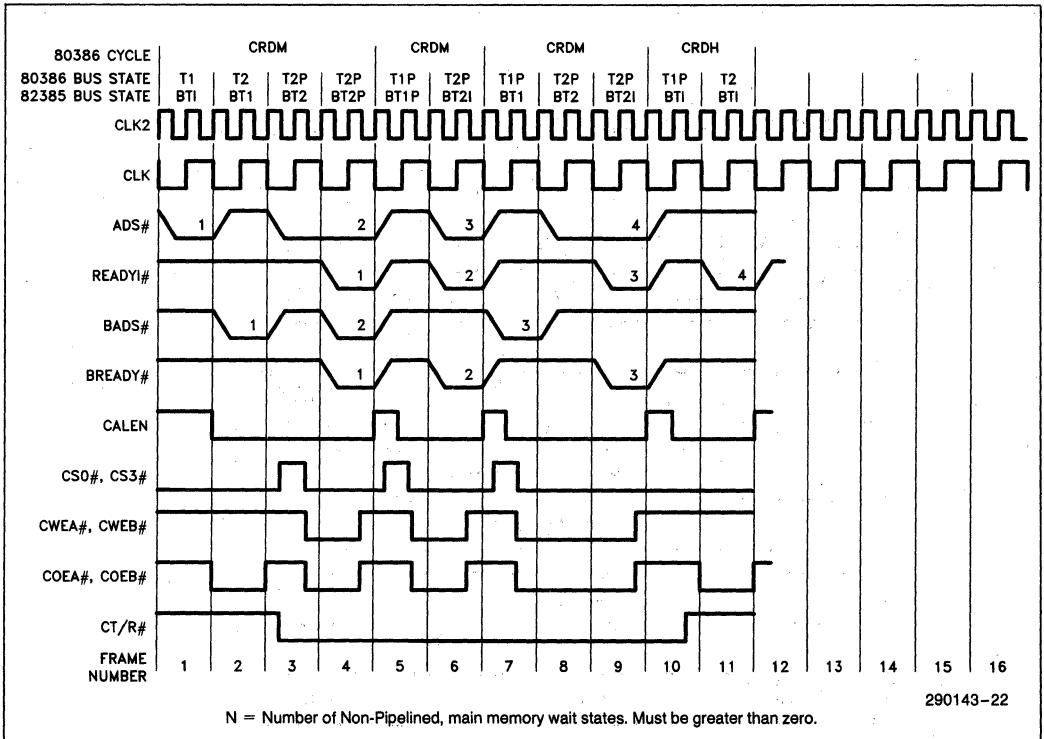
N = Number of Non-Pipelined, main memory wait states. Must be greater than zero.

NOTES:
CRDH = Cache Read Hit
CWTH = Cache Write Hit

Figure 4-5A. Cache Read and Write Cycles—Direct Mapped (N = 1)

the 82385 cannot enable the 74646 transceiver (Figure 4-1) until after the cache outputs are disabled. (The timing of the 74646 control signals is described in the next chapter.) The result is that the 74646 cannot be enabled soon enough to support N=0 main memory ("N" was defined in section 4.0 as the number of non-pipelined main memory wait states). This means that memory which can run with zero wait states in a non-pipelined cycle should not be mapped into cacheable memory. This should not present a problem, however, as a main memory system built with N=0 memory has no need of a cache. (The main memory is as fast as the cache.) Zero wait state memory can be supported if it is decoded as non-cacheable. The 82385 knows that a cycle is non-cacheable in time not to drive the cache output enables, and can thus enable the 74646 sooner.

In a write hit, the 82385 only updates the cache bytes that are meant to be updated as directed by the 80386 byte enables. This prevents corrupting cache data in partial doubleword writes. Note in Figure 4-5A that the appropriate bytes are selected via the cache byte select lines CS0#–CS3#. In a read hit, all four select lines are driven as the 80386 will simply ignore data it does not need. Also, in a cache update (read miss), all four selects are active in order to update the cache with a complete line (doubleword).



N = Number of Non-Pipelined, main memory wait states. Must be greater than zero.

NOTE:
CRDH = Cache Read Miss

Figure 4-5B. Cache Update Cycles—Direct Mapped (N=1)

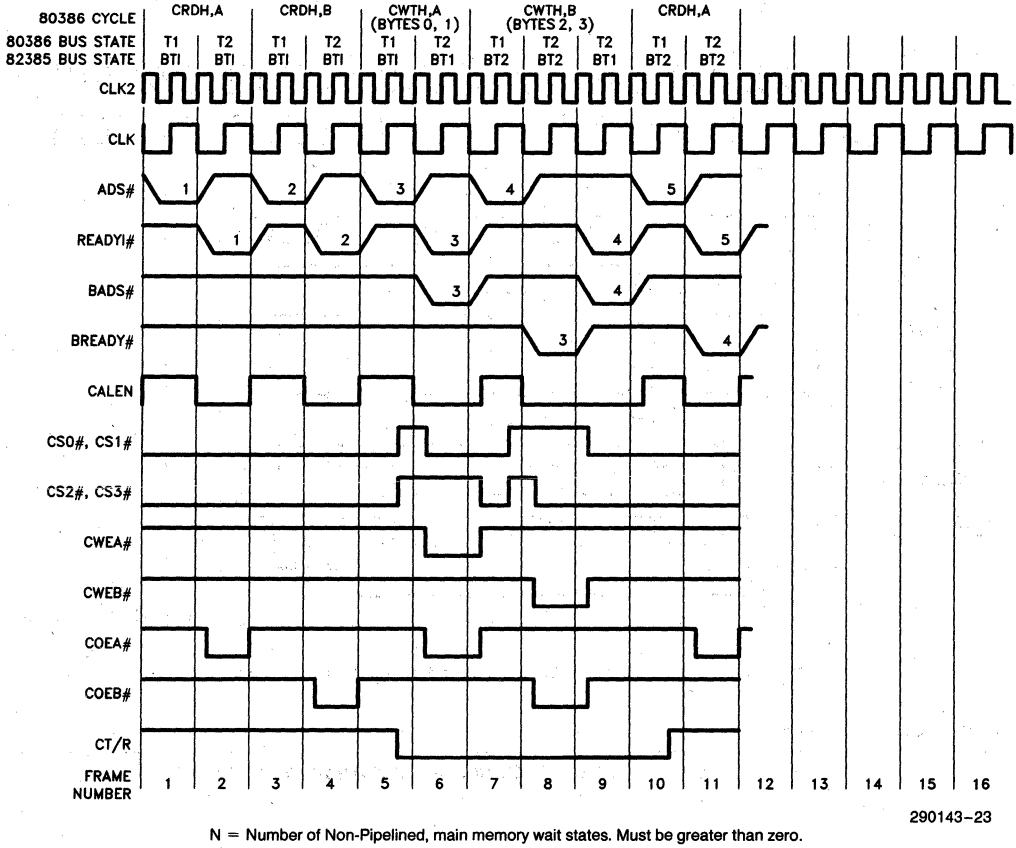
4.2.3 Cache Control—Two-Way Set Associative

Figures 4-6A and 4-6B illustrate the timing of cache read hits, write hits, and updates for a two-way set associative cache. (Note that the cycle sequences are the same as those in Figures 4-5A and 4-5B.) In a cache read hit, only one bank or the other is enabled to drive the 80386 data bus, so unlike the control of a direct mapped cache, the appropriate cache output enable cannot be driven until the outcome of the hit/miss decision is known. (This implies stricter SRAM timing requirements for a two-way set associative cache.) In write hits and read misses, only one bank or the other is updated.

4.3 80387 INTERFACE

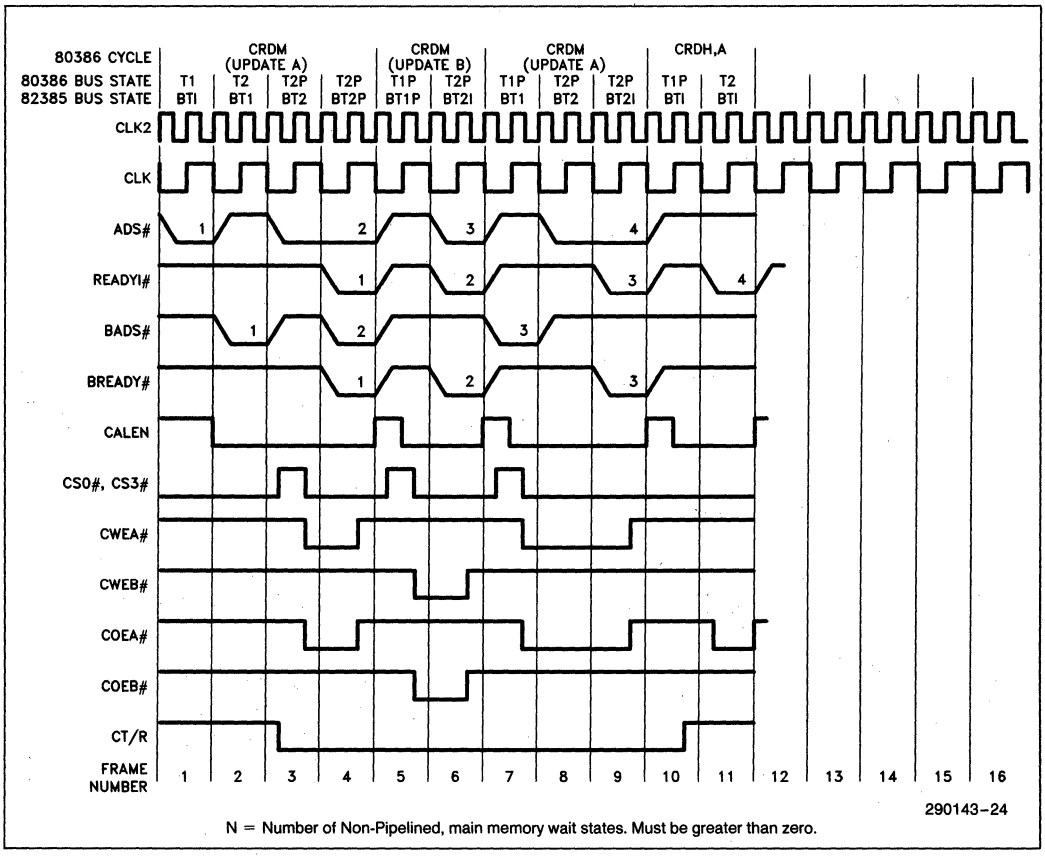
The 80387 Numerics Coprocessor interfaces to the 80386 just as it would in a system without an 82385. The 80387 READYO# output is logically "AND"ed along with all other 80386 local bus ready sources (Figure 4-1), and the output is fed to the 80387 READY#, 82385 READYI#, and 80386 READY# inputs.

The 80386 uniquely addresses the 80387 by driving M/IO# low and A31 high. The 82385 decodes this internally and treats 80387 accesses in the same way it treats 80386 cycles in which LBA# is asserted, it ignores them.



N = Number of Non-Pipelined, main memory wait states. Must be greater than zero.

Figure 4-6A. Cache Read and Write Cycles—Two Way Set Associative (N = 1)



N = Number of Non-Pipelined, main memory wait states. Must be greater than zero.

Figure 4-6B. Cache Update Cycles—Two Way Set Associative (N = 1)

5.0 82385 LOCAL BUS AND SYSTEM INTERFACE

The 82385 system interface is the 82385 Local Bus, which presents an "80386-like" front end to the system. The system ties to it just as it would to an 80386. Although this 80386-like front end is functionally equivalent to an 80386, there are timing differences which can easily be accounted for in a system design.

The following is a description of the interface the 82385 presents to a system. After presenting the 82385 bus state machine, the 82385 bus signals are described, as are techniques for accommodating any differences between the 82385 bus and 80386 bus. Following this is a discussion of the 82385's condition upon reset.

5.1 THE 82385 BUS STATE MACHINE

5.1.1 Master Mode

Figure 5-1A illustrates the 82385 bus state machine when the 82385 is programmed in master mode. Note that it is almost identical to the 80386 bus state machine, only now the bus states are 82385 bus states (BT1P, BTH, etc.) and the state transitions are conditioned by 82385 bus inputs (BNA#, BHOLD, etc.). Whereas a "pending request" to the 80386 state machine indicates that the 80386 execution or prefetch unit needs bus access, a pending request to the 82385 state machine indicates that an 80386 bus cycle needs to be forwarded to the system (read miss, non-cacheable read, write, etc.). The only difference between the state machines is

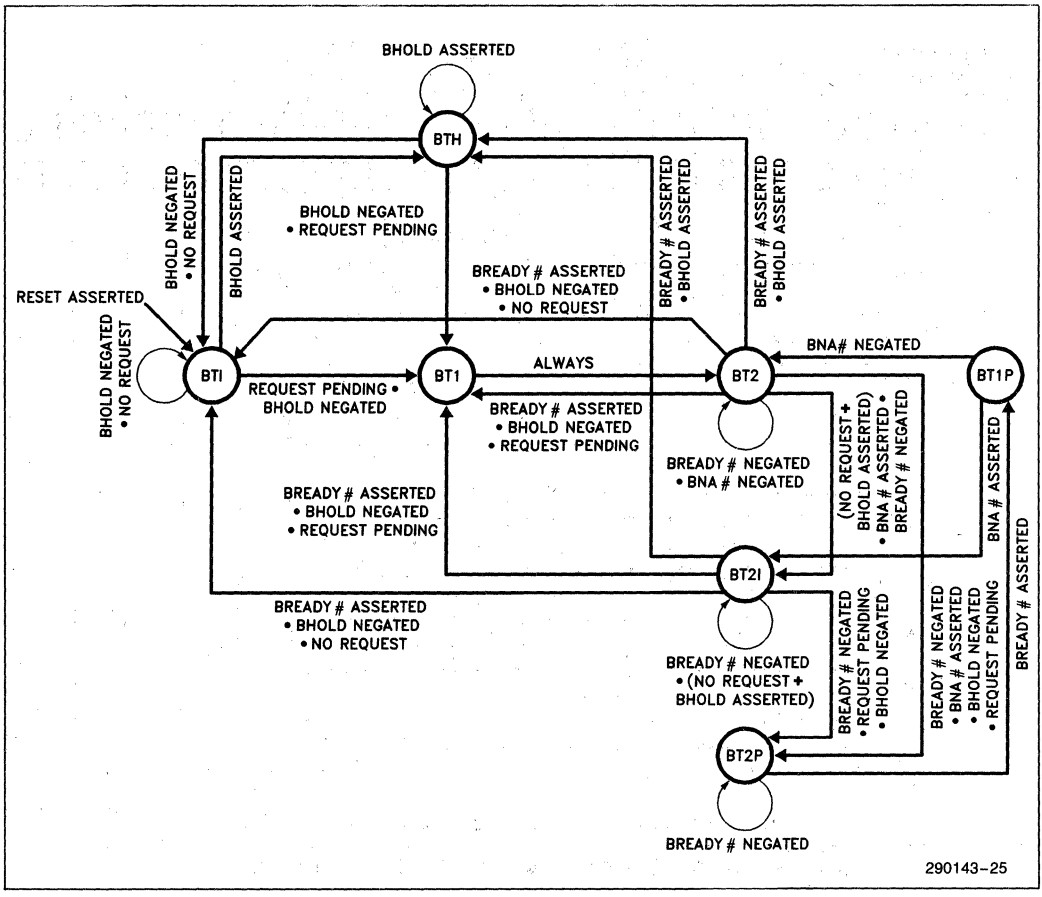


Figure 5-1A. 82385 Local Bus State Machine—Master Mode

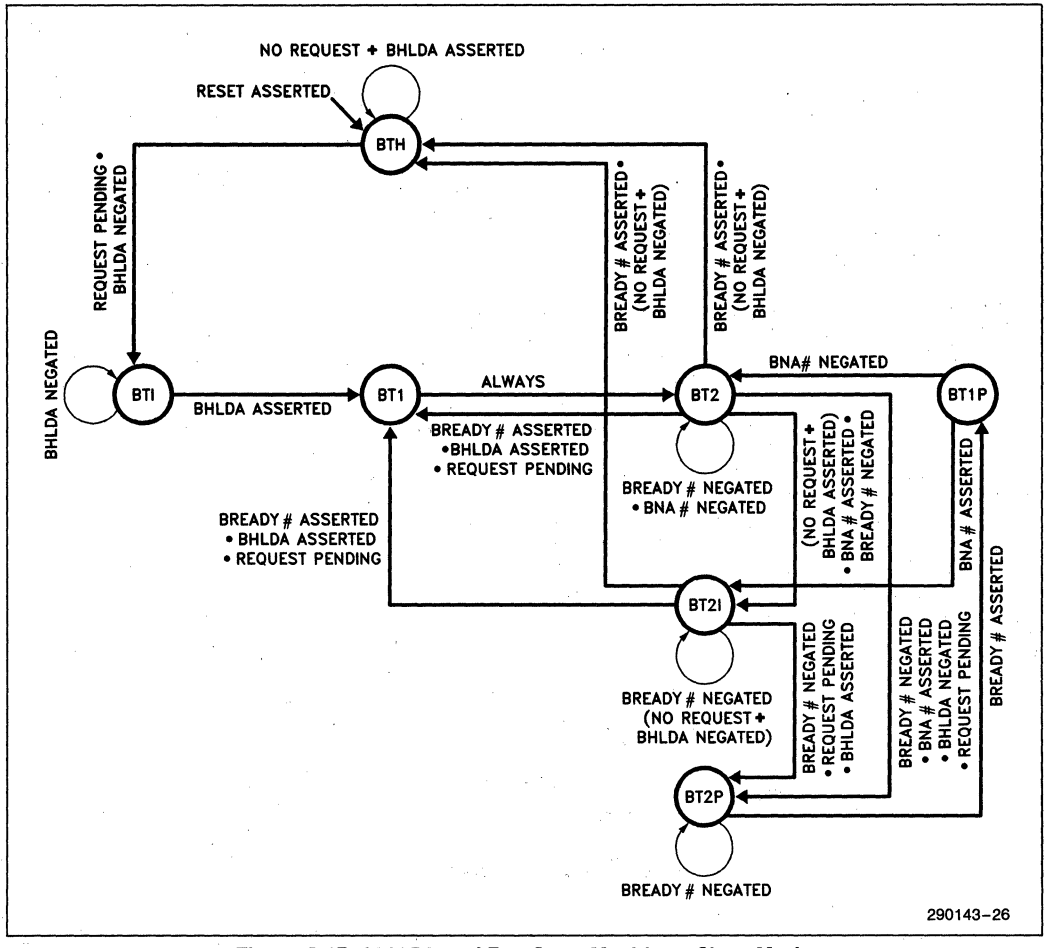


Figure 5-1B. 82385 Local Bus State Machine—Slave Mode

that the 82385 does not implement a direct BT1P–BT2P transition. If BNA# is asserted in BT1P, the resulting state sequence is BT1P-BT2I-BT2P. The 82385's ability to sustain a pipeline is not affected by the lack of this state transition.

5.1.2 Slave Mode

The 82385's slave mode state machine (Figure 5-1B) is similar to the master mode machine except that now transitions are conditioned by BHLDA rather than BHOLD. (Recall that in slave mode, the roles of BHOLD and BHLDA are reversed from their master mode roles.) Figure 5-2 clarifies slave mode state machine operation. Upon reset, a slave mode 82385 enters the BTH state. When the 80386 of the slave 82385 subsystem has a cycle that needs to be forwarded to the system, the 82385 moves to BTI and issues a hold request via BHOLD. It is important to note that a slave mode 82385 does not drive the bus in a BTI state. When the master or bus arbiter returns BHLDA, the slave 82385 enters BT1 and runs the cycle. When the cycle is completed, and if no additional requests are pending, the 82385 moves back to BTH and disables BHOLD.

If, while a slave 82385 is running a cycle, the master or arbiter drops BHLDA (Figure 5-2B), the 82385 will complete the current cycle, move to BTH and remove the BHOLD request. If the 82385 still had cycles to run when it was kicked off the bus, it will immediately assert a new BHOLD and move to BTI to await bus acknowledgement. Note, however, that it will only move to BTI if BHLDA is negated, ensuring that the handshake sequence is completed.

There are several cases in which a slave 82385 will not immediately release the bus if BHLDA is dropped. For example, if BHLDA is dropped during a BT2P state, the 82385 has already committed to the next system bus pipelined cycle and will execute it before releasing the bus. Also, the 82385 will complete the second half of a two-cycle 16-bit transfer, or will complete a sequence of locked cycles before releasing the bus. This should not present any problems, as a properly designed arbiter will not assume that the 82385 has released the bus until it sees BHOLD become inactive.

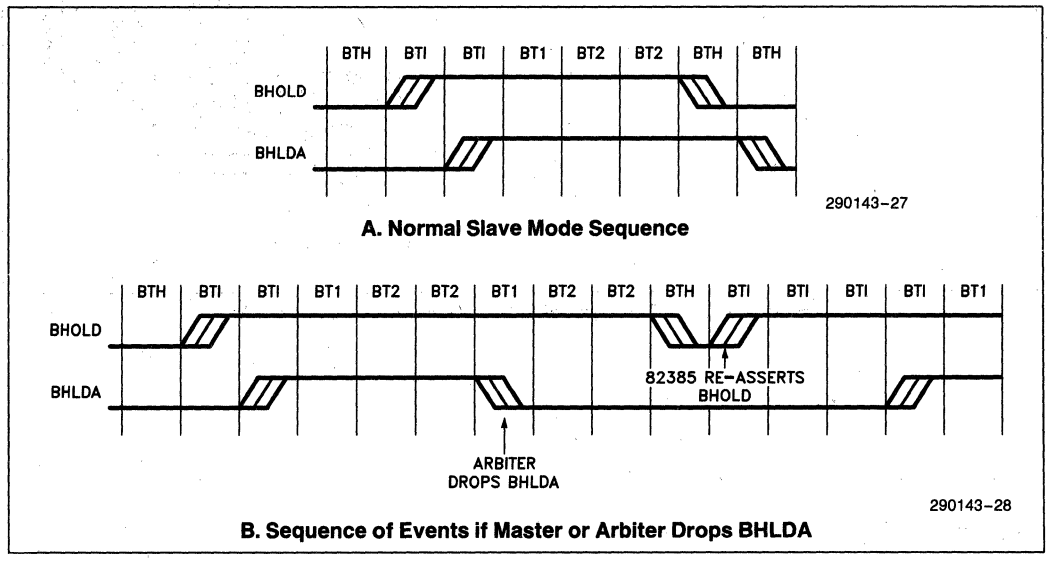


Figure 5-2. BHOLD/BHLDA—Slave Mode

5.2 The 82385 Local Bus

The 82385 bus can be broken up into two groups of signals: those which have direct 80386 counterparts, and additional status and control signals provided by the 82385. The operation and interaction of all 82385 bus signals are depicted in Figures 5-3A through 5-3L for a wide variety of cycle sequences. These diagrams serve as a reference for the 82385 bus discussion and provide insight into the dual bus operation of the 82385.

5.2.1 82385 Bus Counterparts to 80386 Signals

The following sections discuss the signals presented on the 82385 local bus which are functional equivalents to the signals present at the 80386 local bus.

5.2.1.1 ADDRESS BUS (BA2–BA31) AND CYCLE DEFINITION SIGNALS (BM/IO#, BD/C#, BW/R#)

These signals are not driven directly by the 82385, but rather are the outputs of the 74374 address/cycle definition latch. (Refer to Figure 4-1 for the hardware interface.) This latch is controlled by the 82385 BACP and BAOE# outputs. The behavior and timing of these outputs and the latch they control (typically F or AS series TTL) ensure that BA2–BA31, BM/IO#, BW/R#, and BD/C# are completely compatible in timing and function to their 80386 counterparts.

The behavior of BACP can be seen in Figure 5-3B, where the rising edge of BACP latches and forwards the 80386 address and cycle definition signals in a BT1 or first BT2P state. However, the 82385 need not be the current bus master to latch the 80386 address, as evidenced by cycle 4 of Figure 5-3A. In this case, the address is latched in frame 8, but not forwarded to the system (via BAOE#) until frame 10. (The latch and output enable functions of the 74374 are independent and invisible to one another.)

Note that in frames 2 and 6 of Figure 5-3E, the BACP pulses are marked "False." The reason is that BACP is issued and the address latched before the hit/miss determination is made. This ensures that should the cycle be a miss, the 82385 bus can move directly into BT1 without delay. In the case of a hit, the latched address is simply never qualified by the assertion of BADS#. The 82385 bus stays in BTI if there is no access pending (new cycle is a hit) and no bus activity. It will move to and stay in BT2I if the system has requested a pipelined cycle and the 82385 does not have a pending bus access (new cycle is a hit).

5.2.1.2 DATA BUS (BD0–BD31)

The 82385 data bus is the system side of the 74646 latching transceiver. (See Figure 4-1.) This device is controlled by the 82385 outputs LDSTB, DOE#, and BT/R#. LDSTB latches data in write cycles, DOE# enables the transceiver outputs, and BT/R# controls the transceiver direction. The interaction of these signals and the transceiver is such that BD0–BD31 behave just like their 80386 counterparts. The transceiver is configured such that data flow in write cycles (A to B) is latched, and data flow in read cycles (B to A) is flow-through.

Although BD0–BD31 function just like their 80386 counterparts, there is a timing difference that must be accommodated for in a system design. As mentioned above, the transceiver is transparent during read cycles, so the transceiver propagation delay must be added to the 80386 data setup. In addition, the cache SRAM setup must be accommodated for in cache read miss cycles.

For non-cacheable reads the data setup is given by:

Min BD0–BD31 Read Data Setup = 80386 Min Data Setup + 74646 B-to-A Max Propagation Delay

The required BD0–BD31 setup in a cache read miss is given by:

Min BD0–BD31 Read Data Setup = 74646 B-to-A Max Propagation Delay + Cache SRAM Min Write Setup + One CLK2 Period − 82385 CWEA# or CWEB# Min Delay

If a data buffer is located between the 80386 data bus and the cache SRAMs, then its maximum propagation delay must be added to the above formula as well. A design analysis should be completed for every new design to determine actual margins.

A design can accommodate the increased data setup by choosing appropriately fast main memory DRAMs and data buffers. Alternatively, a designer may deal with the longer setup by inserting an extra wait state into cache read miss cycles. If an additional state is to be inserted, the system bus controller should sample the 82385 MISS# output to distinguish read misses from cycles that do not require the longer setup. Tips on using the 82385 MISS# signal are presented later in this chapter.

The behavior of LDSTB, DOE#, and BT/R# can be understood via Figures 5-3A through 5-3L. Note that in cycle 1 of Figure 5-3A (a non-cacheable system read), DOE# is activated midway through BT1, but

in cycle 1 of Figure 5-3B (a cache read miss), DOE# is not activated until midway through BT2. As described in the last chapter, the reason is that in a cacheable read cycle, the cache SRAMs are enabled to drive the 80386 data bus before the outcome of the hit/miss decision (in anticipation of a hit). In cycle 1 of Figure 5-3B, the assertion of DOE# must be delayed until after the 82385 has disabled the cache output buffers. The result is that N = 0 main memory should not be mapped into the cache.

5.2.1.3 BYTE ENABLES (BBE0# - BBE3#)

These outputs are driven directly by the 82385, and are completely compatible in timing and function with their 80386 counterparts. When an 80386 cycle is forwarded to the 82385 bus, the 80386 byte enables are duplicated on BBE0# - BBE3#. The one exception is a cache read miss, during which BBE0# - BBE3# are all active regardless of the status of the 80386 byte enables. This ensures that the cache is updated with a valid 32-bit entry.

5.2.1.4 ADDRESS STATUS (BADS#)

BADS# is identical in function and timing to its 80386 counterpart. It is asserted in BT1 and BT2P states, and indicates that valid address and cycle definition (BA2 - BA31, BBE0# - BBE3#, BM/IO#, BW/R#, BD/C#) information is available on the 82385 bus.

5.2.1.5 READY (BREADY#)

The 82385 BREADY# input terminates 82385 bus cycles just as the 80386 READY# input terminates 80386 bus cycles. The behavior of BREADY# is the same as that of READY#, but note in the A.C. timing specifications that a cache read miss requires a longer BREADY# setup than do other cycles. This must be accommodated for in ready logic design.

5.2.1.6 NEXT ADDRESS (BNA#)

BNA# is identical in function and timing to its 80386 counterpart. Note that in Figures 5-3A through 5-3L, BNA# is assumed asserted in every BT1P or first BT2 state. Along with the 82385's pipelining of the 80386, this ensures that the timing diagrams accurately reflect the full pipelined nature of the dual bus structure.

5.2.1.7 BUS LOCK (BLOCK#)

The 80386 flags a locked sequence of cycles by asserting LOCK#. During a locked sequence, the 80386 does not acknowledge hold requests, so the sequence executes without interruption by another master. The 82385 forces all locked 80386 cycles to run on the 82385 bus (unless LBA# is active), regardless of whether or not the referenced location resides in the cache. In addition, a locked sequence of 80386 cycles is run as a locked sequence on the 82385 bus; BLOCK# is asserted and the 82385 does not allow the sequence to be interrupted. Locked writes (hit or miss) and locked read misses affect the cache and cache directory just as their

unlocked counterparts do. A locked read hit, however, is handled differently. The read is necessarily forced to run on the 82385 local bus, and as the data returns from main memory, it is "re-copied" into the cache. (See Figure 5-3L.) The directory is not changed as it already indicates that this location exists in the cache. This activity is invisible to the system and ensures that semaphores are properly handled.

BLOCK# is asserted during locked 82385 bus cycles just as LOCK# is asserted during locked 80386 cycles. The BLOCK# maximum valid delay, however, differs from that of LOCK#, and this must be accounted for in any circuitry that makes use of BLOCK#. The difference is due to the fact that LOCK#, unlike the other 80386 cycle definition signals, is not pipelined. The situation is clarified in Figure 5-3K. In cycle 2 the state of LOCK# is not known before the corresponding system read starts (Frames 4 and 5). In this case, LOCK# is asserted at the beginning of T1P, and the delay for BLOCK# to become active is the delay of LOCK# from the 80386 plus the propagation delay through the 82385. This occurs because T1P and the corresponding BT1P are concurrent (Frame 5). The result is that BLOCK# should not be sampled at the end of BT1P. The first appropriate sampling point is midway through the next state, as shown in Frame 6. In Figure 5-3L, the maximum delay for BLOCK# to become valid in Frame 4 is the same as the maximum delay for LOCK# to become valid from the 80386. This is true since the pipelining issue discussed above does not occur.

5.2.2 Additional 82385 Bus Signals

The 82385 bus provides two status outputs and one control input that are unique to cache operation and thus have no 80386 counterparts. The outputs are MISS#, and WBS, and the input is FLUSH.

5.2.2.1 CACHE READ/WRITE MISS INDICATION (MISS#)

MISS# can be thought of as an extra 82385 bus cycle definition signal similar to BM/IO#, BW/R#, and BD/C#, that distinguishes cacheable read and write misses from other cycles. MISS#, like the other definition signals, becomes valid with BADS# (BT1 or first BT2P). The behavior of MISS# is illustrated in Figures 5-3B, 5-3C, and 5-3J. The 82385 floats MISS# when another master owns the bus, allowing multiple 82385s to share the same node in multi-cache systems. MISS# should thus be lightly pulled up (~20 KΩ) to keep it negated during hold (BTH) states.

MISS# can serve several purposes. As discussed previously, the BD0–BD31 and BREADY# setup times in a cache read miss are longer than in other cycles. A bus controller can distinguish these cycles by gating MISS# with BW/R#. MISS# may also prove useful in gathering 82385 system performance data.

5.2.2.2 WRITE BUFFER STATUS (WBS)

WBS is activated when 80386 write cycle data is latched into the 74646 latching transceiver (via LDSTB). It is deactivated upon completion of the write cycle on the 82385 bus when the 82385 sees the BREADY# signal. WBS behavior is illustrated in Figures 5-3F through 5-3J, and potential applications are discussed in chapter 3.

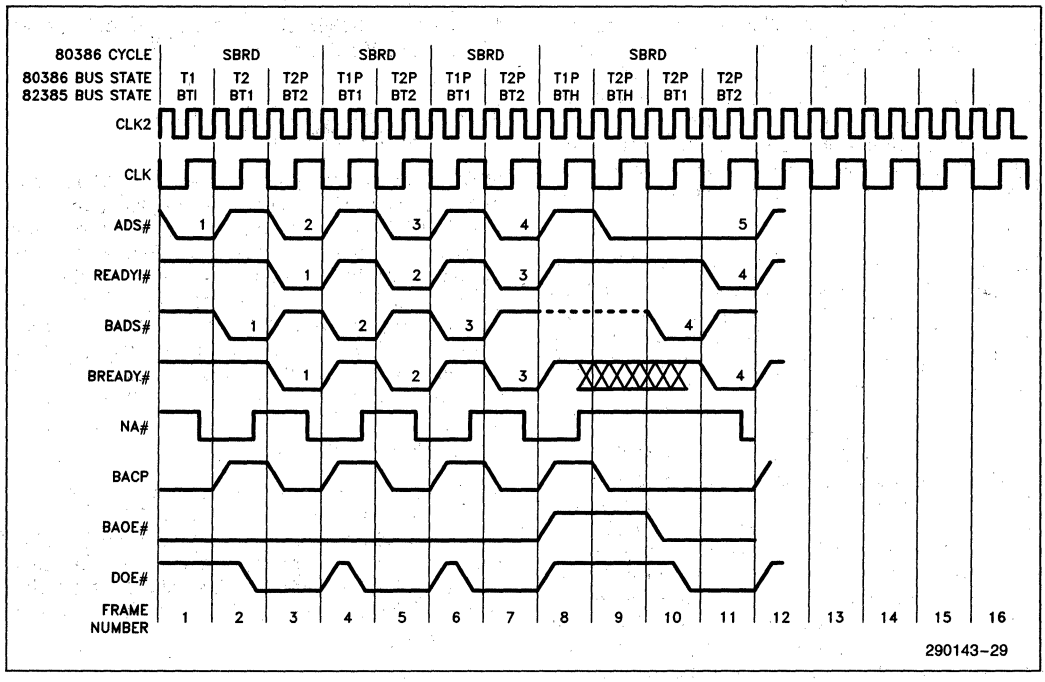


Figure 5-3A. Consecutive SBRD Cycles—(N = 0)

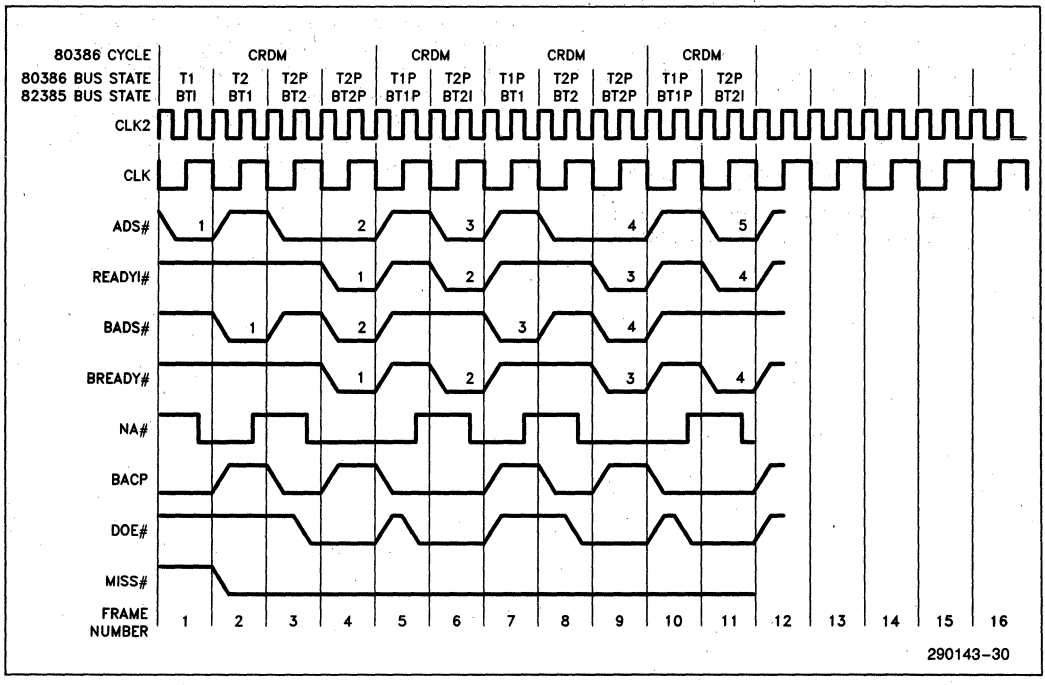


Figure 5-3B. Consecutive CRDM Cycles—(N = 1)

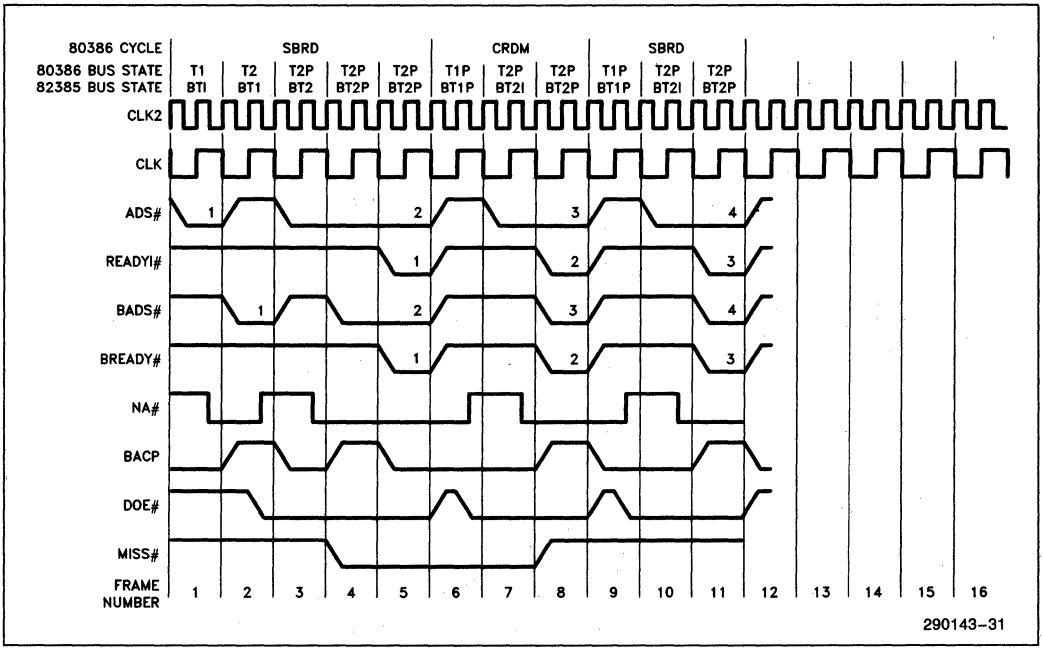


Figure 5-3C. SBRD, CRDM, SBRD—(N = 2)

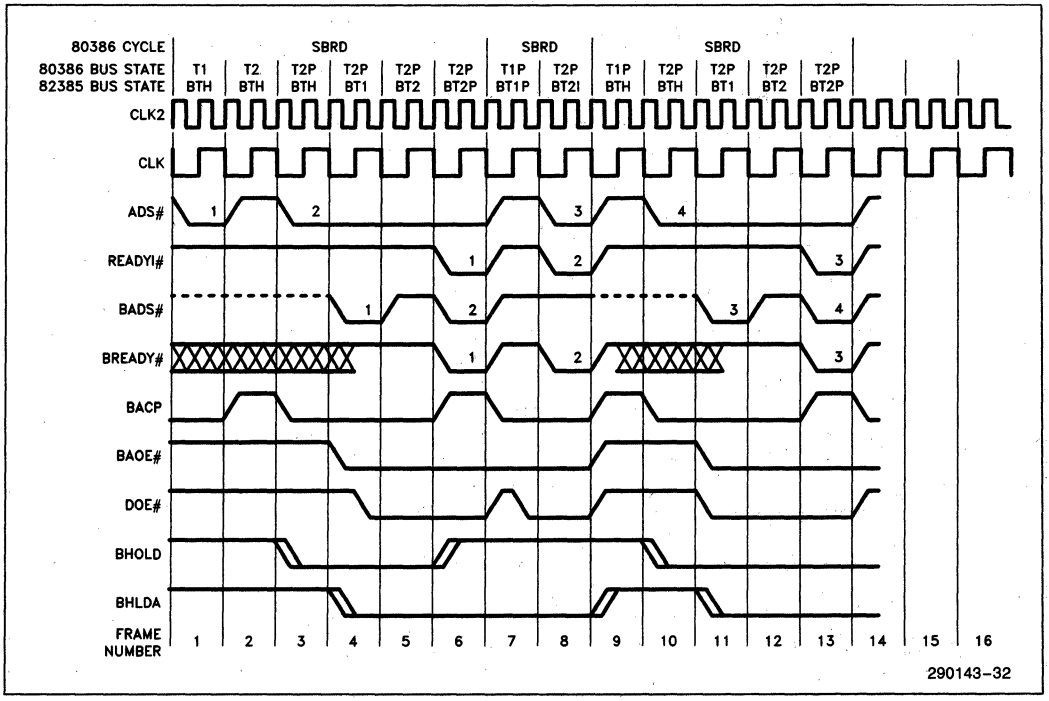


Figure 5-3D. SBRD Cycles Interleaved with BTH States—(N = 1)

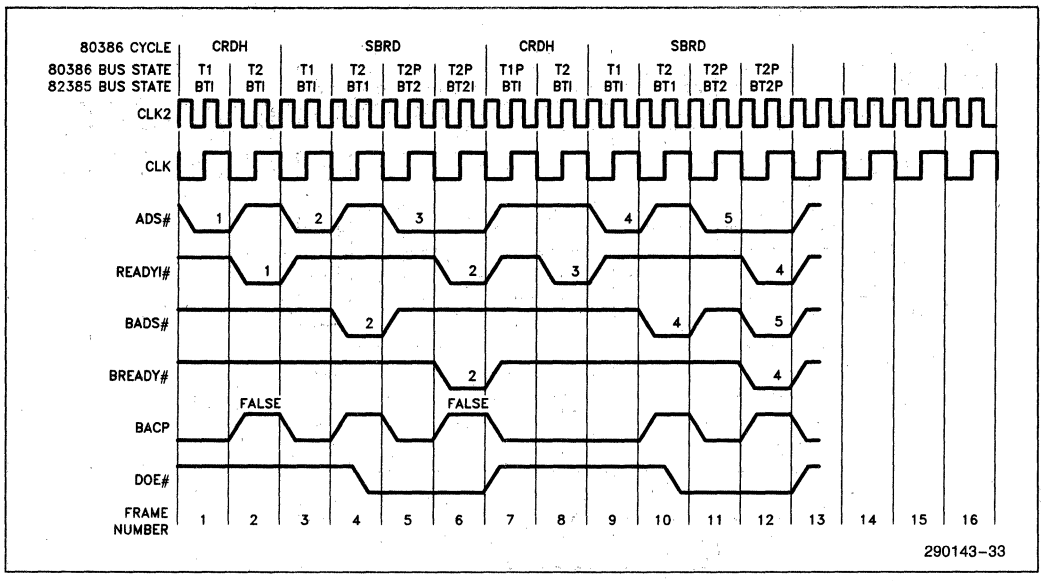


Figure 5-3E. Interleaved SBRD/CRDH Cycles—(N = 1)

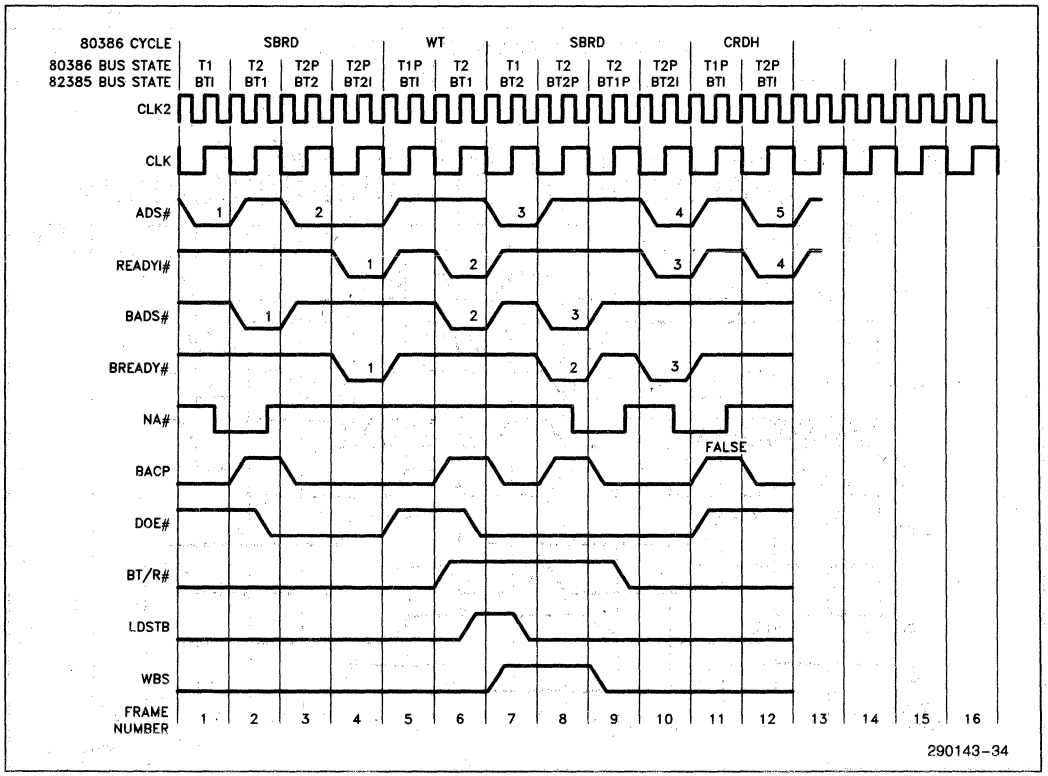


Figure 5-3F. SBRD, WT, SBRD, CRDH—(N = 1)

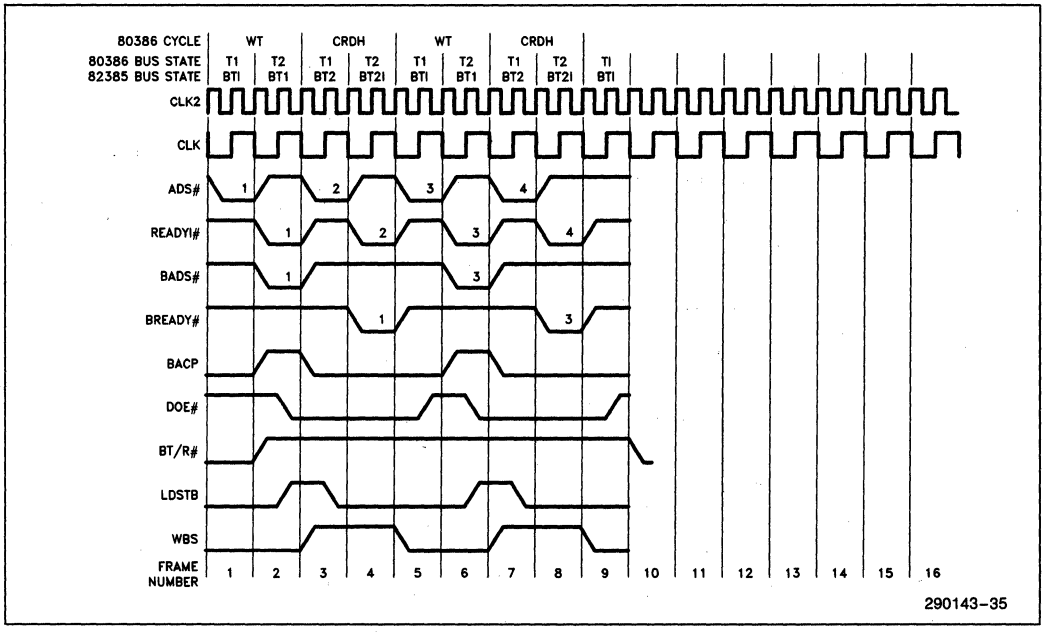


Figure 5-3G. Interleaved WT/CRDH Cycles—(N = 1)

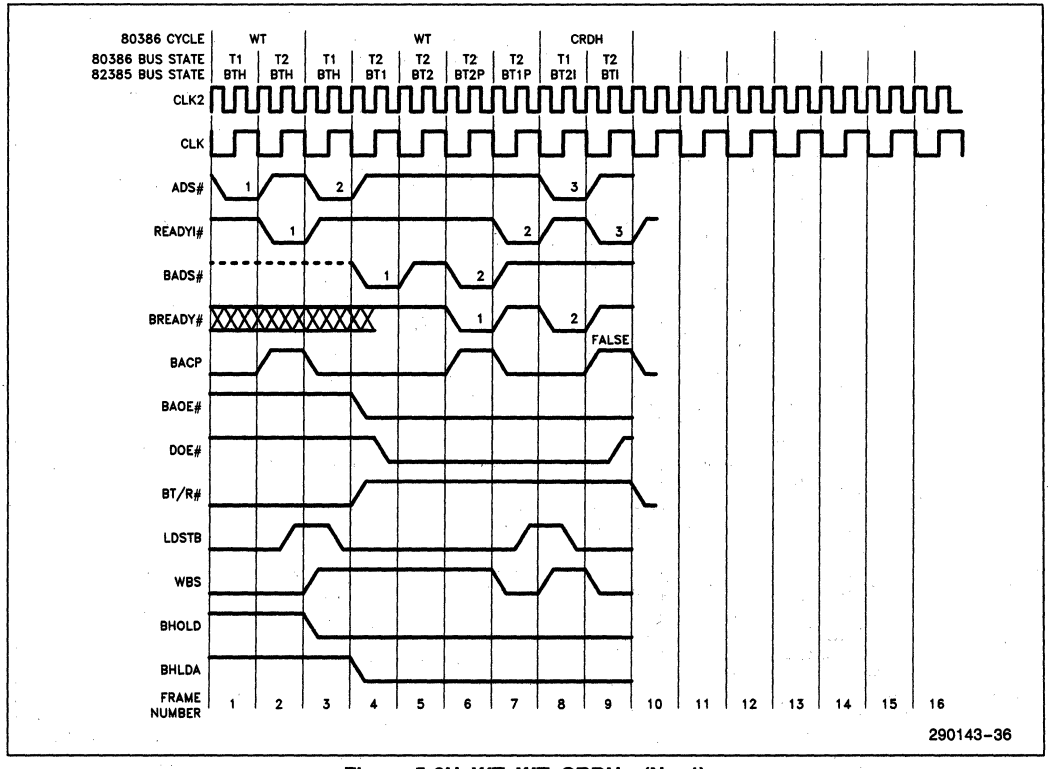


Figure 5-3H. WT, WT, CRDH—(N = 1)

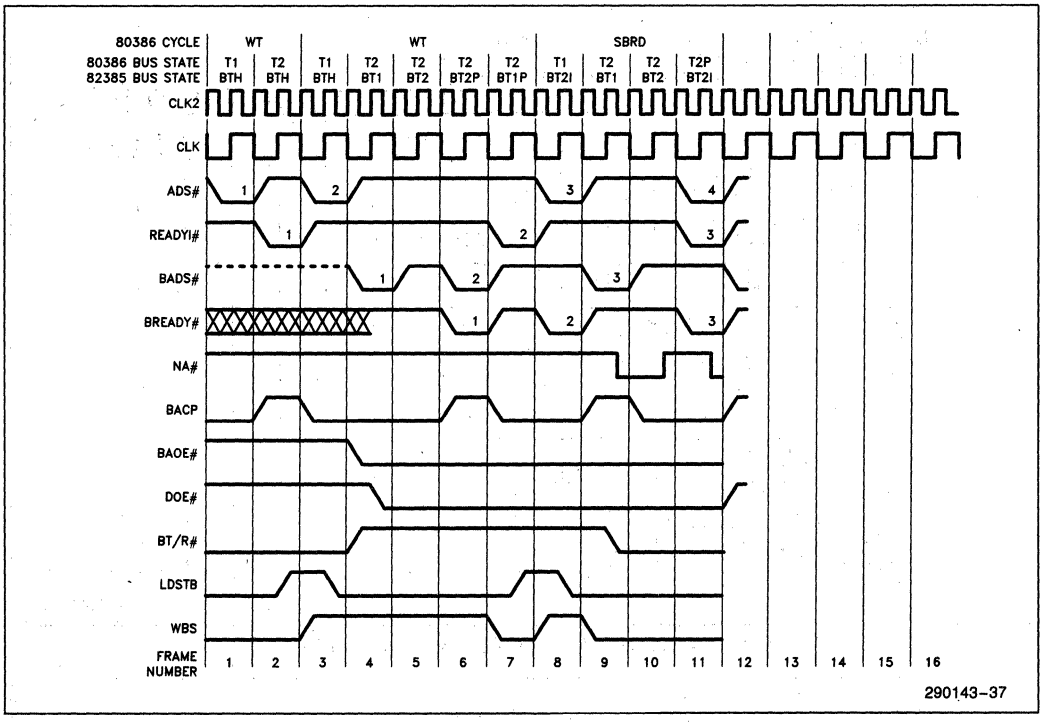


Figure 5-3I. WT, WT, SBRD—(N = 1)

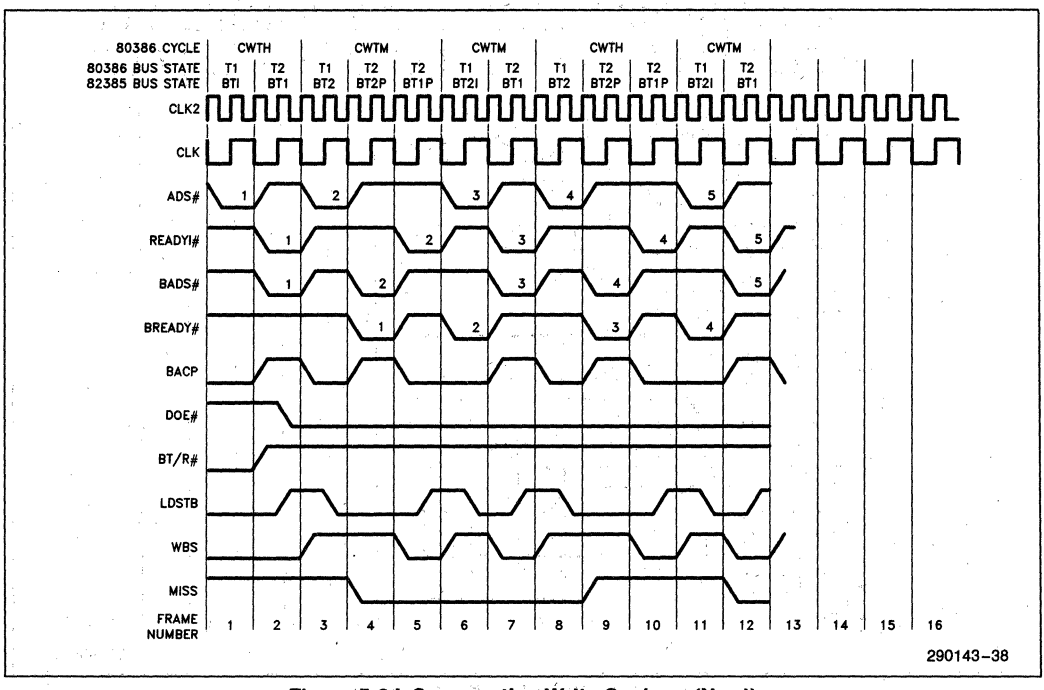


Figure 5-3J. Consecutive Write Cycles—(N = 1)

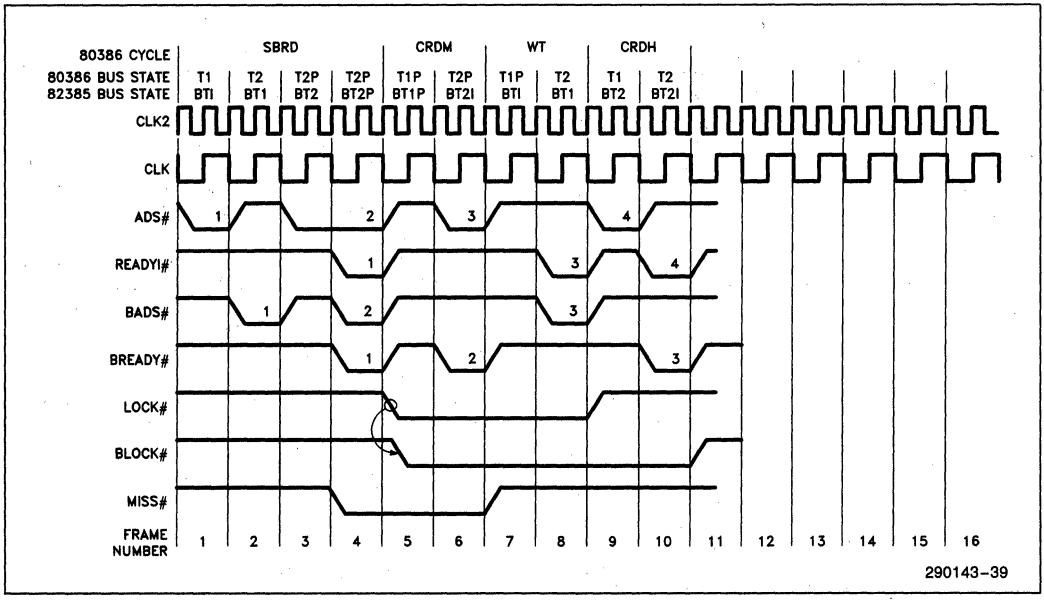


Figure 5-3K. LOCK# /BLOCK# in Non-Cacheable or Miss Cycles—(N = 1)

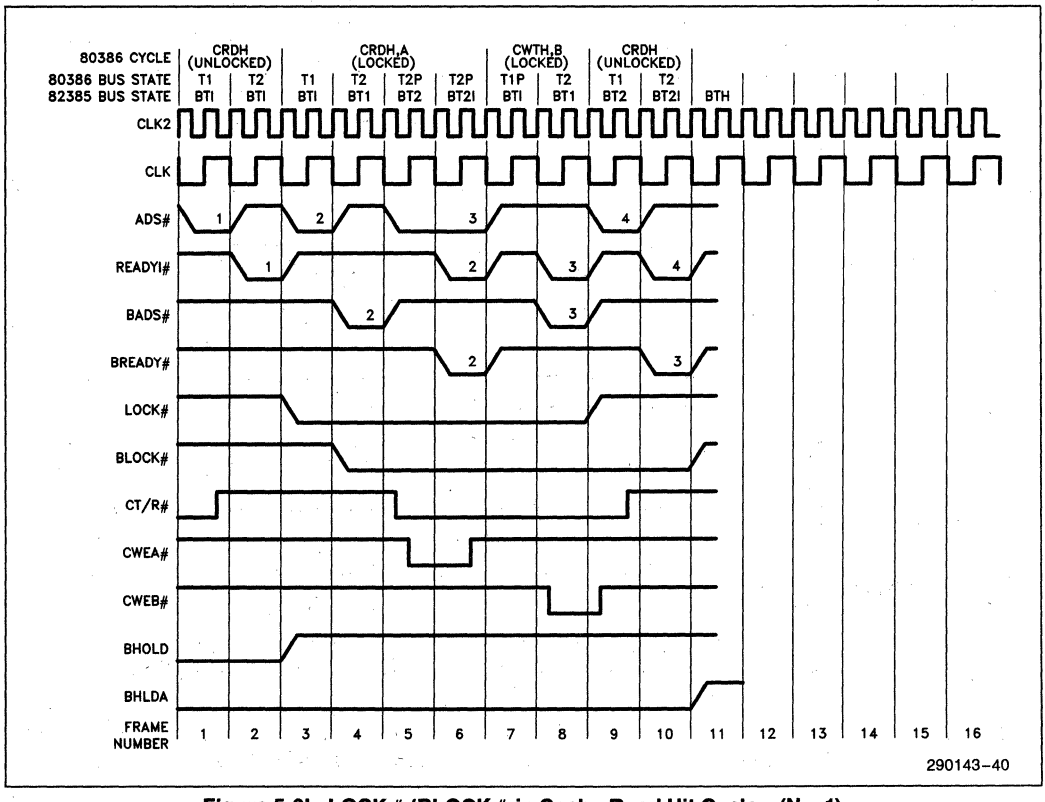


Figure 5-3L. LOCK# /BLOCK# in Cache Read Hit Cycle—(N = 1)

5.2.2.3 CACHE FLUSH (FLUSH)

FLUSH is an 82385 input which is used to reset all tag valid bits within the cache directory. The FLUSH input must be kept active for at least 4 CLK (8 CLK2) periods to complete the directory flush. Flush is generally used in diagnostics but can also be used in applications where snooping cannot guarantee coherency.

5.3 BUS WATCHING (SNOOP) INTERFACE

The 82385's bus watching interface consists of the snoop address (SA2–SA31), snoop strobe (SSTB#), and snoop enable (SEN) inputs. If masters reside at the system bus level, then the SA2–SA31 inputs are connected to the system address lines and SEN the system bus memory write command. SSTB# indicates that a valid address is present on the system bus. Note that the snoop bus inputs are synchronous, so care must be taken to ensure that they are stable during their sample windows. If no master resides beyond the 82385 bus level, then SA2–SA31, SEN, and SSTB# can respectively tie directly to BA2–BA31, BW/R#, and BADS#. However, it is recommended that SEN be driven by the logical "AND" of BW/R# and BM/IO# so as to prevent I/O writes from unnecessarily invalidating cache data.

When the 82385 detects a system write by another master, it internally latches SA2–SA31 and runs a cache look-up to see if the altered main memory location is duplicated in the cache. If yes (a snoop hit), the line valid bit associated with that cache entry is cleared. An important feature of the 82385 is that even if the 83086 is running zero wait state hits out of the cache, all snoops are serviced. This is accomplished by time multiplexing the cache directory between the 80386 address and latched system address. If the SSTB# signal occurs during an 82385 comparison cycle (for the 80386), the 80386 cycle has the highest priority in accessing the cache directory. This takes the first of the two 80386 states. The other state is then used for the snoop comparison. This worst case example, depicted in Figure 5-4, shows the 80386 running zero wait state hits on the 80386 local bus, and another master running zero wait state writes on the 82385 bus. No snoops are missed, and no performance penalty incurred.

5.4 RESET DEFINITION

Table 5-1 summarizes the states of all 82385 outputs during reset and initialization. A slave mode 82385 tri-states its "80386-like" front end. A master mode 82385 emits a pulse stream on its BACP output. As the 80386 address and cycle definition lines reach their reset values, this stream will latch the reset values through to the 82385 bus.

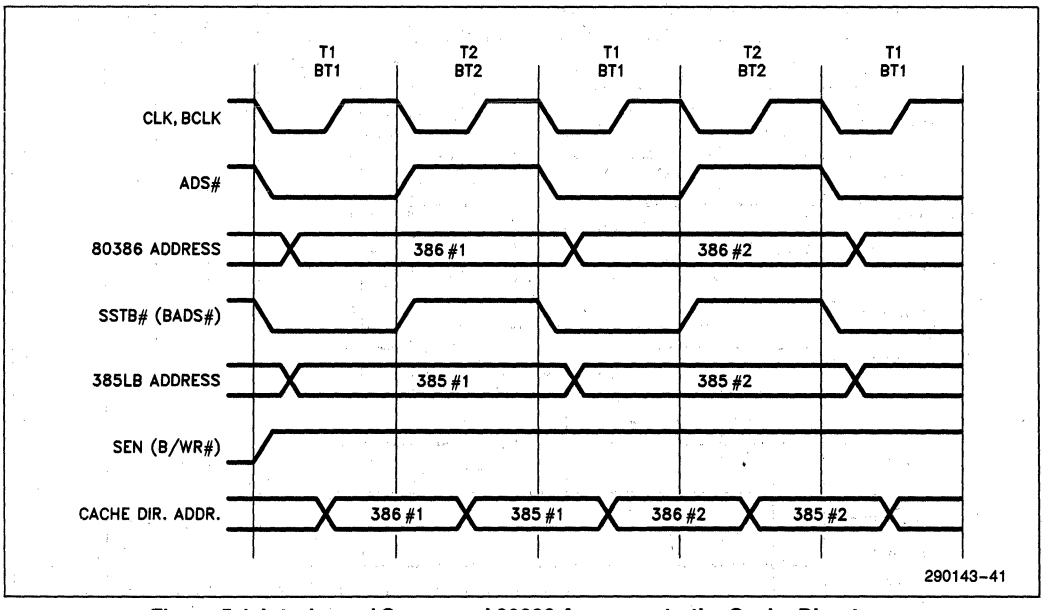


Figure 5.4. Interleaved Snoop and 80386 Accesses to the Cache Directory

Table 5-1. Pin State During RESET and Initialization

Output Name	Signal Level During RESET and Initialization	
	Master Mode	Slave Mode
NA#	High	High
READY0#	High	High
BRDYEN#	High	High
CALEN	High	High
CWEA# – CWEB#	High	High
CS0# – CS3#	Low	Low
CT/R#	High	High
COEA# – COEB#	High	High
BADS#	High	High Z
BBE0# – BBE3#	386 BE#	High Z
BLOCK#	High	High Z
MISS#	High	High Z
BACP	Pulse(1)	Pulse
BAOE#	Low	High
BT/R#	Low	Low
DOE#	High	High
LDSTB	Low	Low
BHOLD	—	Low
BHLDA	Low	—
WBS	Low	Low

NOTE:
1. In Master Mode, BAOE# is low and BACP emits a pulse stream during reset. As the 80386 address and cycle definition signals reach their reset values, the pulse stream on BACP will latch these values through to the 82385 local bus.

6.0 82385 SYSTEM DESIGN CONSIDERATIONS

6.1 INTRODUCTION

This chapter discusses techniques which should be implemented in an 82385 system. Because of the high frequencies and high performance nature of the 80386/82385 system, good design and layout techniques are necessary. It is always recommended to perform a complete design analysis on new system designs.

6.2 POWER AND GROUNDING

6.2.1 Power Connections

The 82385 utilizes 8 power (V_{CC}) and 10 ground (V_{SS}) pins. All V_{CC} and V_{SS} pins must be connected to their appropriate plane. On a printed circuit board, all V_{CC} pins must be connected to the power plane and all V_{SS} pins must be connected to the ground plane.

6.2.2 Power Decoupling

Although the 82385 itself is generally a "passive" device in that it has few output signals, the cache

subsystem as a whole is quite active. Therefore, liberal decoupling capacitance should be placed around the 82385 cache subsystem.

Low inductance capacitors and interconnects are recommended for best high frequency electrical performance. Inductance can be reduced by shortening circuit board traces between the decoupling capacitors and their respective devices as much as possible. Capacitors specifically for PGA packages are also commercially available, for the lowest possible inductance.

6.2.3 Resistor Recommendations

Because of the dual bus structure of the 82385 subsystem (80386 Local Bus and 82385 Local Bus), any signals which are recommended to be pulled up will be respective to one of the busses. The following sections will discuss signals for both busses.

6.2.3.1 80386 LOCAL BUS

For typical designs, the pullup resistors shown in Table 6-1 are recommended. This table correlates to chapter 7 of the 80386 Data Sheet. However, particular designs may have a need to differ from the listed values. Design analysis is recommended to determine specific requirements.

6.2.3.2 82385 LOCAL BUS

Pullup resistor recommendations for the 82385 Local Bus signals are shown in Table 6-2. Design analysis is necessary to determine if deviations to the typical values given is needed.

Table 6-1. Recommended Resistor Pullups to V_{CC} (80386 Local Bus)

Pin and Signal	Pullup Value	Purpose
ADS# E13	20 KΩ ±10%	Lightly Pull ADS# Negated for 80386 Hold States
LOCK# F13	20 KΩ ±10%	Lightly Pull LOCK# Negated for 80386 Hold States

Table 6-2. Recommended Resistor Pullups to V_{CC} (82385 Local Bus)

Signal and Pin	Pullup Value	Purpose
BADS# N9	20 KΩ ±10%	Lightly Pull BADS# Negated for 82385 Hold States
BLOCK# P9	20 KΩ ±10%	Lightly Pull BLOCK# Negated for 82385 Hold States
MISS# N8	20 KΩ ±10%	Lightly Pull MISS# Negated for 82385 Hold States

6.3 82385 SIGNAL CONNECTIONS

6.3.1 Configuration Inputs

The 82385 configuration signals (M/S#, 2W/D#) must be connected (pulled up) to the appropriate logic level for the system design. There are also two reserved 82385 inputs which must be tied to the appropriate level. Refer to Table 6-3 for the signals and their required logic level.

Table 6-3. 82385 Configuration Inputs Logic Levels

Pin and Signal	Logic Level	Purpose
M/S# B13	High	Master Mode Operation
	Low	Slave Mode Operation
2W/D# D12	High	2-Way Set Associative
	Low	Direct Mapped
Reserved L14	High	Must be tied to V_{CC} via a pull-up for proper functionality
Reserved A14	High	Must be tied to V_{CC} via a pull-up for proper functionality

NOTE:
The listed 82385 pins which need to be tied high should use a pull-up resistor in the range of 5 KΩ to 20 KΩ.

6.3.2 CLK2 and RESET

The 82385 has two inputs to which the 80386 CLK2 signal must be connected. One is labeled CLK2 (82385 pin C13) and the other is labeled BCLK2 (82385 pin L13). These two inputs must be tied together on the printed circuit board.

The 82385 also has two reset inputs. RESET (82385 pin D13) and BRESET (82385 pin K12) must be connected on the printed circuit board.

6.4 UNUSED PIN REQUIREMENTS

For reliable operation, ALWAYS connect unused inputs to a valid logic level. As is the case with most other CMOS processes, a floating input will increase the current consumption of the component and give an indeterminate state to the component.

6.5 CACHE SRAM REQUIREMENTS

The 82385 offers the option of using SRAMs with or without an output enable pin. This is possible by inserting a transceiver between the SRAMs and the 80386 local data bus. This transceiver may also be desirable in a system which has a very heavily loaded 80386 local data bus. The following sections discuss the SRAM requirements for all cache configurations.

6.5.1 Cache Memory without Transceivers

As discussed in section 3.2, the 82385 presents all of the control signals necessary to access the cache memory. The SRAM chip selects, write enables, and output enables are driven directly by the 82385. Table 6-4 lists the required SRAM specifications. These specifications allow for zero margin. They should be used as guides for the actual system design.

6.5.2 Cache Memory With Transceivers

To implement an 82385 subsystem using cache memory transceivers, it is necessary to create an output enable signal for the transceiver. In a 2-way set associative organization this signal is the logical "AND" of COEA# and CWEA# for bank A and the "AND" of COEB# and CWEB# for bank B. A direct mapped cache needs to only use the equation of one bank (A or B). All other cache control signals are driven directly by the 82385. Table 6-5 lists the required SRAM specifications. These specifications allow for zero margin. They should be used as guides for the actual system design.

Table 6-4. SRAM Specs for Non-Buffered Cache Memory

SRAM Spec Requirements				
	Direct Mapped		2-Way Set Associative	
	16 MHz	20 MHz	16 MHz	20 MHz
Read Cycle Requirements				
Address Access (MAX)	64 ns	44 ns	62 ns	42 ns
Chip Select Access (MAX)	76	56	76	56
OE# to Data Valid (MAX)	25	19	19	14
OE# to Data Float (MAX)	20	20	20	20
Write Cycle Requirements				
Chip Select to End of Write (MIN)	40	30	40	30
Address Valid to End of Write (MIN)	58	42	56	40
Write Pulse Width (MIN)	40	30	40	30
Data Setup (MAX)	—	—	—	—
Data Hold (MIN)	4	4	4	4

Table 6-5. SRAM Specs for Buffered Cache Memory

SRAM Spec Requirements				
	Direct Mapped		2-Way Set Associative	
	16 MHz	20 MHz	16 MHz	20 MHz
Read Cycle Requirements				
Address Access (MAX)	57 ns	37 ns	55 ns	35 ns
Chip Select Access (MAX)	68	48	68	48
OE# to Data Valid (MAX)	N/A	N/A	N/A	N/A
OE# to Data Float (MAX)	N/A	N/A	N/A	N/A
Write Cycle Requirements				
Chip Select to End of Write (MIN)	40	30	40	30
Address Valid to End of Write (MIN)	58	42	56	40
Write Pulse Width (MIN)	40	30	40	30
Data Setup (MAX)	25	15	25	15
Data Hold (MIN)	3	3	3	3

7.0 SYSTEM TEST CONSIDERATIONS

7.1 INTRODUCTION

Power On Self Testing (POST) is performed by most systems after a reset. This chapter discusses the requirements for properly testing an 82385 based system after power up.

7.2 MAIN MEMORY (DRAM) TESTING

Most systems perform a memory test by writing a data pattern and then reading and comparing the data. This test may also be used to determine the total available memory within the system. Without properly taking into account the 82385 cache memory, the memory test can give erroneous results. This will occur if the cache responds with read hits during the memory test routine.

7.2.1 Memory Testing Routine

In order to properly test main memory, the test routine must not read from the same block consecutively. For instance, if the test routine writes a data pattern to the first 32 kbytes of memory (0000–7FFFH), read from the same block, writes a new pattern to the same locations (0000–7FFFH), and read the new pattern, the second pattern tested would have had data returned from the 82385 cache memory. Therefore, it is recommended that the test routine work with a memory block of at least 64 kbytes. This will guarantee that no 32 kbyte block will be read twice consecutively.

7.3 82385 CACHE MEMORY TESTING

With the addition of SRAMs for the cache memory, it may be desirable for the system to be able to test the cache SRAMs during system diagnostics. This requires the test routine to access only the cache memory. The requirements for this routine are based on where it resides within the memory map. This can be broken into two areas: the routine residing in cacheable memory space or the routine residing in either non-cacheable memory or on the 80386 local bus (using the LBA# input).

7.3.1 Test Routine in the NCA# or LBA# Memory Map

In this configuration, the test routine will never be cached. The recommended method is code which will access a single 32 kbyte block during the test. Initially, a 32 kbyte read (assume 0000-7FFFH) must be executed. This will fill the cache directory with the address information which will be used in the diagnostic procedure. Then, a 32 kbyte write to the same address locations (0000–7FFFH) will load the cache with the desired test pattern (due to write hits). The comparison can be made by completing another 32 kbyte read (same locations, 0000–7FFFH), which will be cache read hits. Subsequent writes and reads to the same addresses will enable various patterns to be tested.

7.3.2 Test Routine in Cacheable Memory

In this case, it must be understood that the diagnostic routine must reside in the cache memory before the actual data testing can begin. Otherwise, when the 80386 performs a code fetch, a location within the cache memory which is to be tested will be altered due to the read miss (code fetch) update.

The first task is to load the diagnostic routine into the top of the cache memory. It must be known how much memory is required for the code as the rest of the cache memory will be tested as in the earlier method. Once the diagnostics have been cached (via read updates), the code will perform the same type of read/write/read/compare as in the routine explained in the previous section. The difference is that now the amount of cache memory to be tested is 32 kbytes minus the length of the test routine.

7.4 82385 CACHE DIRECTORY TESTING

Since the 82385 does not directly access the data bus, it is not possible to easily complete a comparison of the cache directory. However, the cache memory tests described in section 7.3 will indicate if the directory is working properly. Otherwise, the data comparison within the diagnostics will show locations which fail.

There is a slight possibility that the cache memory comparison could pass even if locations within the directory gave false hit/miss results. This could cause the comparison to always be performed to main memory instead of the cache and give a proper comparison to the 80386. The solution here is to use the MISS# output of the 82385 as an indicator to a diagnostic port which can be read by the 80386. It could also be used to flag an interrupt if a failure occurs.

The implementation of these techniques in the diagnostics will guarantee the proper functionality of the 82385 subsystem.

7.5 SPECIAL FUNCTION PINS

As mentioned in chapter 3, there are three 82385 pins which have reserved functions in addition to their normal operational functions. These pins are MISS#, WBS, and FLUSH.

As discussed previously, the 82385 performs a directory flush when the FLUSH input is held active for at least 4 CLK (8 CLK2) cycles. However, the FLUSH pin also serves as a diagnostic input to the 82385. The 82385 will enter a reserved mode if the FLUSH pin is high at the falling edge of RESET.

If, during normal operation, the FLUSH input is active for only one CLK (2 CLK2) cycle/s, the 82385 will enter another reserved mode. Therefore it must be guaranteed that FLUSH is active for at least the 4 CLK (8 CLK2) cycle specification.

WBS and MISS# serve as outputs in the 82385 reserved modes.

8.0 MECHANICAL DATA

8.1 INTRODUCTION

This chapter discusses the physical package and its connections in detail.

8.2 PIN ASSIGNMENT

The 82385 pinout as viewed from the top side of the component is shown by Figure 8-1. Its pinout as viewed from the Pin side of the component is shown in Figure 8-2.

V_{CC} and V_{SS} connections must be made to multiple V_{CC} and V_{SS} (GND) pins. Each V_{CC} and V_{SS} must be connected to the appropriate voltage level. The circuit board should include V_{CC} and GND planes for power distribution and all V_{CC} and V_{SS} pins must be connected to the appropriate plane.

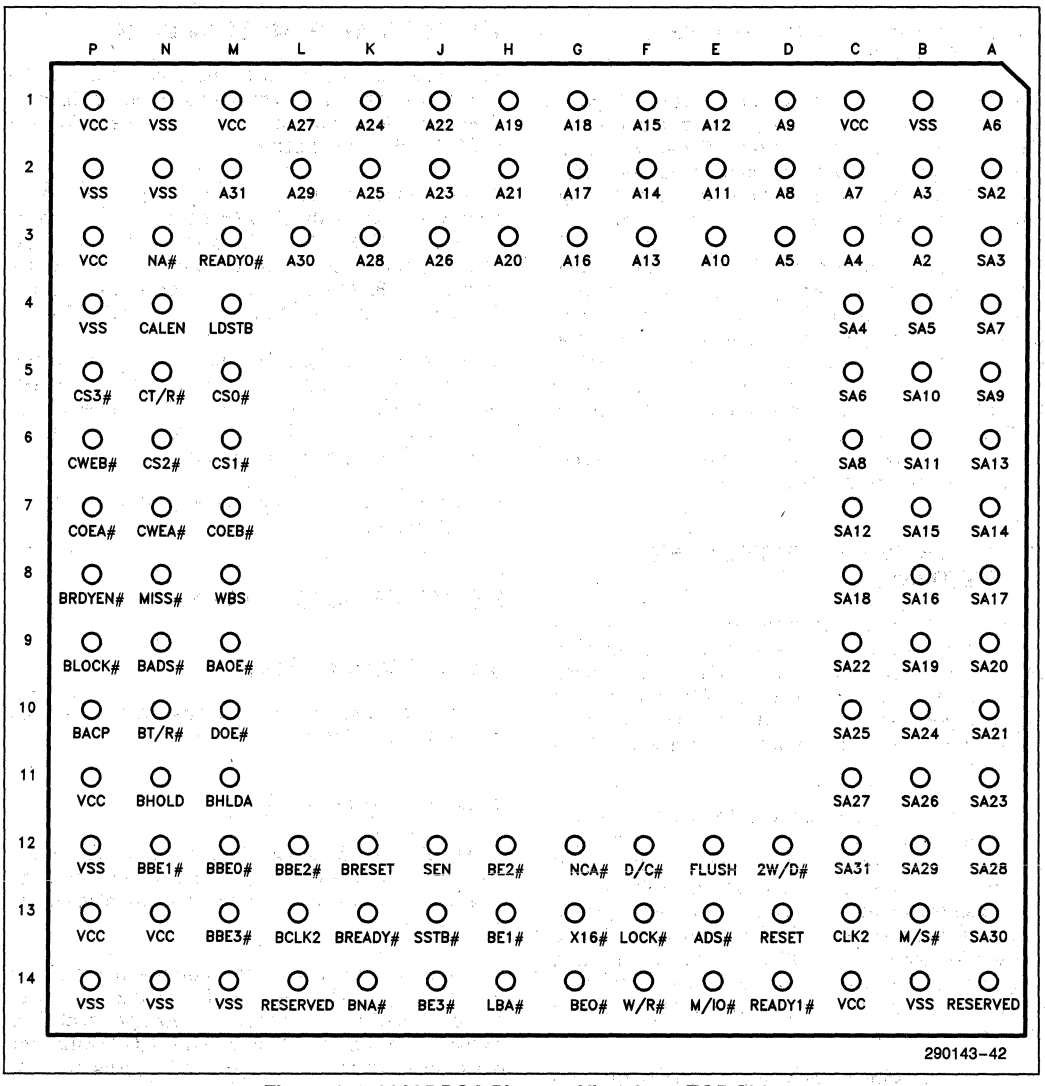


Figure 8-1. 82385 PGA Pinout—View from TOP Side

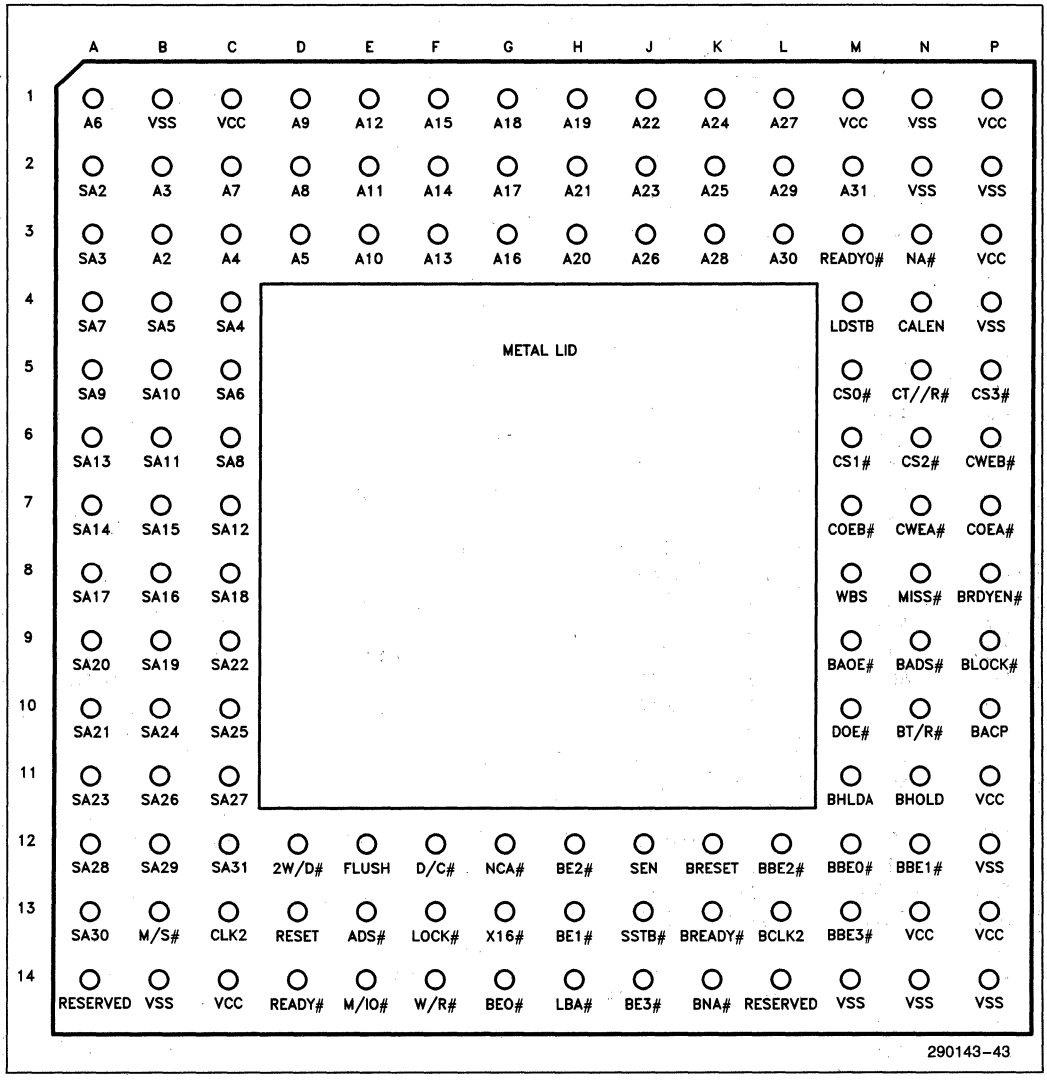


Figure 8-2. 82385 PGA Pinout—View from PIN Side

Table 8-1. 82385 PGA Pinout—Functional Grouping

Pin/Signal		Pin/Signal		Pin/Signal		Pin/Signal	
M2	A31	C12	SA31	C1	V_{CC}	B1	V_{SS}
L3	A30	A13	SA30	C14	V_{CC}	B14	V_{SS}
L2	A29	B12	SA29	M1	V_{CC}	M14	V_{SS}
K3	A28	A12	SA28	N13	V_{CC}	N1	V_{SS}
L1	A27	C11	SA27	P1	V_{CC}	N2	V_{SS}
J3	A26	B11	SA26	P3	V_{CC}	N14	V_{SS}
K2	A25	C10	SA25	P11	V_{CC}	P2	V_{SS}
K1	A24	B10	SA24	P13	V_{CC}	P4	V_{SS}
J2	A23	A11	SA23	E13	ADS#	P12	V_{SS}
J1	A22	C9	SA22			P14	V_{SS}
H2	A21	A10	SA21	F14	W/R#		
H3	A20	A9	SA20	F12	D/C#	N9	BADS#
H1	A19	B9	SA19	E14	M/IO#	M12	BBE0#
G1	A18	C8	SA18	F13	LOCK#	N12	BBE1#
G2	A17	A8	SA17			L12	BBE2#
G3	A16	B8	SA16	N3	NA#	M13	BBE3#
F1	A15	B7	SA15			P9	BLOCK#
F2	A14	A7	SA14	G13	X16#		
F3	A13	A6	SA13	G12	NCA#	K14	BNA#
E1	A12	C7	SA12	H14	LBA#		
E2	A11	B6	SA11	D14	READYI#	N4	CALEN
E3	A10	B5	SA10	M3	READYO#	P7	COEA#
D1	A9	A5	SA9			M7	COEB#
D2	A8	C6	SA8	E12	FLUSH	N7	CWEA#
C2	A7	A4	SA7	M8	WBS	P6	CWEB#
A1	A6	C5	SA6	N8	MISS#	M5	CS0#
D3	A5	B4	SA5			M6	CS1#
C3	A4	C4	SA4	D12	2W/D#	N6	CS2#
B2	A3	A3	SA3	B13	M/S#	P5	CS3#
B3	A2	A2	SA2	M10	DOE#		
G14	BE0#	J12	SEN	M4	LDSTB	N5	CT/R#
H13	BE1#	J13	SSTB#				
H12	BE2#			N11	BHOLD	P8	BRDYEN#
J14	BE3#	A14	RESERVED	M11	BHLDA	K13	BREADY#
		L14	RESERVED			P10	BACP
C13	CLK2					M9	BAOE#
D13	RESET					N10	BT/R#
K12	BRESET						
L13	BCLK2						

8.3 PACKAGE DIMENSIONS AND MOUNTING

The 82385 package is a 132-pin ceramic Pin Grid Array (PGA). The pins are arranged 0.100 inch (2.54 mm) center-to-center, in a 14 x 14 matrix, three rows around (Figure 8-3).

A wide variety of available sockets allow low insertion force or zero insertion force mounting. These come in a choice of terminals such as soldertail, surface mount, or wire wrap.

8.4 PACKAGE THERMAL SPECIFICATION

The PGA case temperature should be measured at the center of the top surface opposite the pins, as in Figure 8-4. The case temperature may be measured in any environment to determine whether or not the 82385 is within the specified operating range.

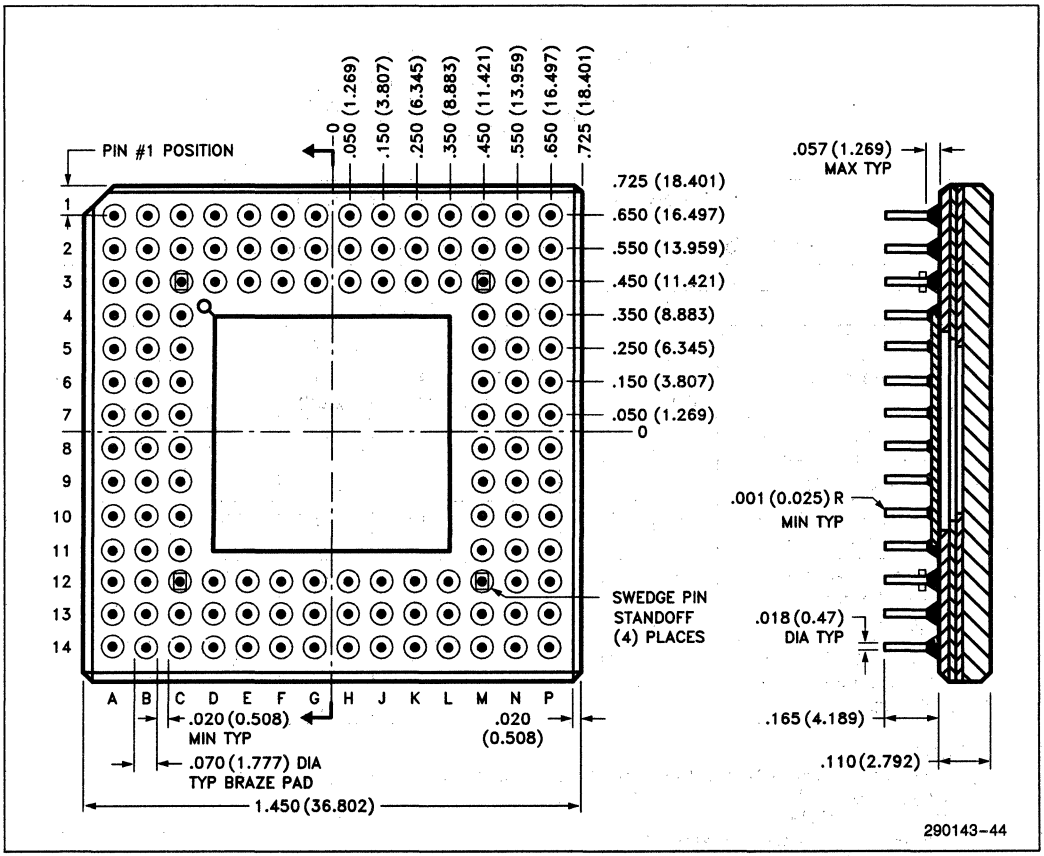


Figure 8-3. 132-Pin PGA Package Dimensions

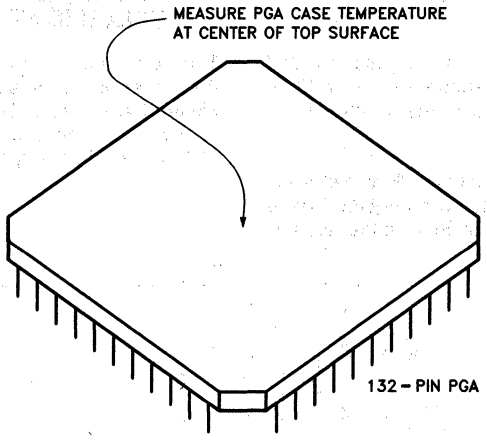
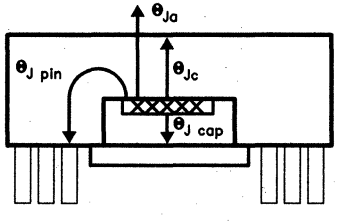


Figure 8-4. Measuring 82385 PGA Case Temperature

Table 8-2. 82385 PGA Package Typical Thermal Characteristics.

Parameter	Thermal Resistance—°C/Watt, Airflow—f³/min (m³/sec)						
	0 (0)	50 (0.25)	100 (0.50)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)
θ Junction-to-Case (Case Measured as Figure 8.4)	2	2	2	2	2	2	2
θ Case-to-Ambient (No Heatsink)	19	18	17	15	12	10	9
θ Case-to-Ambient (with Omnidirectional Heatsink)	16	15	14	12	9	7	6
θ Case-to-Ambient (with Unidirectional Heatsink)	15	14	13	11	8	6	5

NOTES:
1. Table 8-2 applies to 82385 PGA plugged into socket or soldered directly onto board.
2. $\theta_{JA} = \theta_{JC} + \theta_{CA}$.
3. $\theta_{J\text{-}CAP} = 4°C/W$ (approx.)
   $\theta_{J\text{-}PIN} = 4°C/W$ (inner pins) (approx.)
   $\theta_{J\text{-}PIN} = 8°C/W$ (outer pins) (approx.)



9.0 ELECTRICAL DATA

9.1 INTRODUCTION

This chapter presents the A.C. and D.C. specifications for the 82385.

9.2 MAXIMUM RATINGS

Storage Temperature ..... −65°C to +150°C

Case Temperature Under Bias... −65°C to +110°C

Supply Voltage with Respect to V_{SS} ..... −0.5V to +6.5V

Voltage on any other Pin ..... −0.5V to V_{CC} + 0.5V

NOTE:
Stress above those listed may cause permanent damage to the device. This is a stress rating only and functional operation at these or any other conditions above those listed in the operational sections of this specification is not implied.

Exposure to absolute maximum rating conditions for extended periods may affect device reliability. Although the 82385 contains protective circuitry to resist damage from static electrical discharges, always take precautions against high static voltages or electric fields.

9.3 D.C. SPECIFICATIONS T_{CASE} = 0°C to +85°C; V_{CC} = 5V ±5%; V_{SS} = 0V

Table 9-1. D.C. Specifications (16 MHz and 20 MHz)

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{IL}	Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{CL}	CLK2, BCLK2 Input Low	−0.3	0.8	V	(Note 1)
V_{CH}	CLK2, BCLK2 Input High	V_{CC} − 0.8	V_{CC} + 0.3	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 4 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −1 mA
I_{CC}	Power Supply Current		275	mA	(Note 2)
I_{LI}	Input Leakage Current		±15	μA	$0V < V_{IN} \leq V_{CC}$
I_{LO}	Output Leakage Current		±15	μA	$0.45 < V_{OUT} < V_{CC}$
C_{IN}	Input Capacitance		10	pF	(Note 3)
C_{CLK}	CLK2 Input Capacitance		20	pF	(Note 3)

NOTES:
1. Minimum value is not 100% tested.
2. I_{CC} is specified with inputs driven to CMOS levels. I_{CC} may be higher if driven to TTL levels.
3. Sampled only.

9.3 D.C. SPECIFICATIONS T_{CASE} = 0°C to +85°C; V_{CC} = 5V ±5%; V_{SS} = 0V (Continued)

Table 9-2. D.C. Specifications (25 MHz)

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{IL}	Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{ILC}	CLK2, BCLK2 Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IHC}	CLK2, BCLK2 Input High Voltage	3.7	V_{CC} + 0.3	V	
V_{OL}	Output Low Voltage		0.45	V	
V_{OH}	Output High Voltage	2.4		V	
I_{LI}	Input Leakage Current		±15	μA	$0V < V_{IN} < V_{CC}$
I_{LO}	Output Leakage Current		±15	μA	$0.45 < V_{OUT} < V_{CC}$
I_{CC}	Supply Current		300	mA	(Note 2)
C_I	Input Capacitance		10	pF	(Note 3)
C_{CLK}	CLK2 Input Capacitance		20	pF	(Note 3)

NOTES:
1. Minimum value is not 100% tested.
2. I_{CC} is specified with inputs driven to CMOS levels. I_{CC} may be higher if driven to TTL levels.
3. Not 100% tested. Test conditions f_C = 1 MHz, Inputs = 0V, T_{CASE} = room.

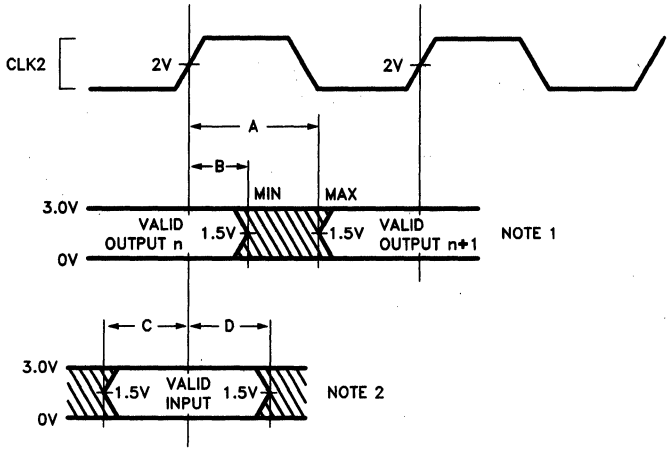
9.4 A.C. SPECIFICATIONS

The A.C. specifications given in the following tables consist of output delays and input setup requirements. The A.C. diagram's purpose is to illustrate the clock edges from which the timing parameters are measured. The reader should not infer any other timing relationships from them. For specific information on timing relationships between signals, refer to the appropriate functional section.

A.C. spec measurement is defined in Figure 9-1. Inputs must be driven to the levels shown when A.C. specifications are measured. 82385 output delays are specified with minimum and maximum limits, which are measured as shown. 82385 input setup and hold times are specified as minimums and define the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct 82385 operation.

9.4.1 Frequency Dependent Signals

The 82385 has signals whose output valid delays are dependent on the clock frequency. These signals are marked in the A.C. Specification Tables with a Note 1.



LEGEND:
A—Maximum output delay specification
B—Minimum output delay specification
C—Minimum input setup specification
B—Minimum input hold specification

NOTES:
1. Under rated loading 82385 output (t_r and t_f) is typically ≤ 4.0 ns from 0.8V to 2.0V.
2. Input waveforms have t_r ≤ 2.0 ns from 0.8V to 2.0V.

Figure 9-1. Drive Levels and Measurement Points for A.C. Specification

A.C. SPECIFICATION TABLES

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C

Table 9-3. A.C. Specifications

Symbol	Parameter	82385-16		82385-20		Units	Notes
		Min	Max	Min	Max		
t1	Operating Frequency	12	16	12	20	MHz	
t2	CLK2 Period	31.25	41.67	25	41.67	ns	
t3	CLK2 High Time	9		8		ns	
t4	CLK2 Low Time	9		8		ns	
t5	CLK2 Fall Time		8		8	ns	
t6	CLK2 Rise Time		8		8	ns	
t7	A(2-31), BE(0-3)#, Lock Setup Time	25		19		ns	(Note 1)
t8	A(2-31), BE(0-3)#, Lock Hold Time	3		3		ns	
t9	W/R#, M/IO#, D/C#, ADS# Setup Time	28		21		ns	(Note 1)
t10	W/R#, M/IO#, D/C#, ADS# Hold Time	5		5		ns	

A.C. SPECIFICATION TABLES (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C

Table 9-3. A.C. Specifications (Continued)

Symbol	Parameter	82385-16		82385-20		Units	Notes
		Min	Max	Min	Max		
t11	READYI# Setup	21		12		ns	(Note 1)
t12	READYI# Hold	4		4		ns	
t13	LBA#, NCA#, X16# Setup Time	16		10		ns	
t14	LBA#, NCA#, X16# Hold Time	4		4		ns	
t15	RESET, BRESET Setup	13		12		ns	
t16	RESET, BRESET Hold	4		4		ns	
t17	NA# Delay	15	42	15	34	ns	(Note 1) C_L = 25 pF
t18	READYO# Delay	4	31	4	28	ns	(Note 1) C_L = 25 pF
t19	BRDYEN# Delay	4	31	4	28	ns	C_L = 40 pF
t21a	CALEN Delay	3	25	3	19	ns	(Note 2) C_L = 40 pF
t21b	CALEN Rising Delay	3	38	3	33	ns	(Notes 1, 3)
t22a	CWEA#, CWEB# Delay	14	31	12	25	ns	(Notes 1, 4) C_L = 75 pF
t22b	CWEA#, CWEB# Pulse Width	40		30		ns	(Notes 1, 5)
t23	CS(0–3)# Delay	14	38	12	33	ns	(Notes 1, 6) C_L = 50 pF
t24	CT/R# Delay	14	38	12	33	ns	(Notes 1, 7) C_L = 75 pF
t25a	COEA#, COEB#, Falling Delay	1	24	1	18	ns	(Note 8) C_L = 75 pF
t25b	COEA#, COEB#, Falling Delay	1	30	1	23	ns	(Notes 1, 9)
t25c	COEA#, COEB#, Rising Delay	5	19	5	15	ns	(Note 10)
t26	CS(0–3)# Active to CWEA#, CWEB# Rising	40		30		ns	(Notes 1, 5)
t27	CWEA#, CWEB# Falling to CS(0–3)# Falling Delay	0		0		ns	
t28	CWEA#, CWEB# Rising to CALEN Rising and CS(0–3)# Falling Delay	0		0		ns	
t31	SA(2–31) Setup	25		19		ns	
t32	SA(2–31) Hold	3		3		ns	
t33	BADS# Valid Delay	6	33	6	28	ns	(Note 1) C_L = 75 pF
t34	BADS# Float Delay	6	35	6	30	ns	
t55	BLOCK#, BBE(0–3)# Valid Delay	4	36	4	30	ns	(Note 1) C_L = 75 pF
t56	MISS# Valid Delay	4	43	4	35	ns	(Note 1) C_L = 75 pF
t57	MISS#, BBE(0–3)#, BLOCK# Float Delay	4	40	4	32	ns	
t58	WBS Delay	4	36	4	30	ns	(Note 1) C_L = 75 pF

A.C. SPECIFICATION TABLES (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C

Table 9-3. A.C. Specifications (Continued)

Symbol	Parameter	82385-16		82385-20		Units	Notes
		Min	Max	Min	Max		
t35	BNA# Setup	11		9		ns	
t36	BNA# Hold	15		15		ns	
t37a	BREADY# Setup	31		26		ns	(Notes 1, 11)
t37b	BREADY# Setup	21		12		ns	(Note 12)
t38	BREADY# Hold	4		4		ns	
t40	BACP Delay	4	23	4	18	ns	C_L = 60 pF
t41	BAOE# Delay	4	23	4	18	ns	
t43a	BT/R#, DOE# Delay	2	25	2	17	ns	C_L = 50 pF
t43b	DOE# Rising Delay	4	21	4	17	ns	
t43c	LDSTB Delay	2	33	2	26	ns	C_L = 50 pF
t44	SEN, SSTB# Setup	15		11		ns	
t45	SEN, SSTB# Hold	5		5		ns	
t46	BHOLD Setup	26		17		ns	(Note 13)
t47	BHOLD Hold	5		5		ns	(Note 13)
t48	BHLDA Delay	6	33	5	28	ns	(Note 13) C_L = 75 pF
t49	BHLDA Setup	20		17		ns	(Note 14)
t50	BHLDA Hold	5		5		ns	(Note 14)
t51	BHOLD Delay	6	33	6	28	ns	(Note 14) C_L = 75 pF
t59	FLUSH Setup	21		16		ns	
t60	FLUSH Hold	5		5		ns	
t61	FLUSH Setup to RESET Low	31		26		ns	
t62	FLUSH Hold from RESET Low	31		26		ns	

NOTES:
1. Frequency dependent specifications.
2. All cycles except cache write hit. CALEN triggers by PHI2 in TIP state.
3. The end of cache write hit cycles. Triggered by PHI1.
4. CWE# transitions by PHI1 in cache write hit cycles. CWE# transitions by PHI2 in cache read miss cycles.
5. Used for cache data memory (SRAM) specifications.
6. In cache write hit cycles, CS(0–3)# transition high by PHI2 and low by PHI1. In cache read miss cycles, CS(0–3)# transition high by PHI1 and low by PHI2.
7. In cache write hit cycles, CT/R# transitions low by PHI2. In cache read hit cycles, CT/R# goes high by PHI2. In cache read miss cycles, CT/R# goes low by PHI1.
8. Direct mapped configuration.
9. Two way set associative configuration.
10. COE# switches high by PHI1 at the end of a cache read hit cycle.
11. Cache read miss cycles.
12. Non-cacheable read cycles and system write cycles.
13. Master mode configuration. BHOLD is an input and BHLDA is an output.
14. Slave mode configuration. BHOLD is an output and BHLDA is an input.

A.C. SPECIFICATION TABLES (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C

All outputs tested specified at 50 pF load unless otherwise noted.

Table 9-4. A.C. Specifications

Symbol	Parameter	82385-25		Units	Notes
		Min	Max		
t1	Operating Frequency	15.4	25	MHz	
t2	CLK2 Period	20	32.5	ns	
t3a	CLK2 High Time	7		ns	(2.0V Threshold)
t3b	CLK2 High Time	4		ns	(3.7V Measurement)
t4a	CLK2 Low Time	7		ns	(2.0V Threshold)
t4b	CLK2 Low Time	5		ns	(0.8V Measurement)
t5	CLK2 Fall Time		7	ns	(3.7V to 0.8V)
t6	CLK2 Rise Time		7	ns	(0.8V to 3.7V)
t7a	A2–A19, A21–A31 Setup Time	14		ns	(Notes 1, 15)
t7b	LOCK# Setup Time	17		ns	(Notes 1, 15)
t7c	BE0–BE3# Setup Time	14		ns	(Notes 1, 15)
t7d	A20 Setup Time	13		ns	(Notes 1, 15)
t8	A2–A31, BE0#–BE3#, LOCK# Hold	3		ns	
t9a	M/IO#, D/C#, Setup Time	17		ns	(Notes 1, 15)
t9b	ADS#, W/R# Setup Time	17		ns	(Notes 1, 15)
t10	W/R#, M/IO#, D/C#, ADS# Hold	3		ns	
t11	READYI# Setup Time	9		ns	(Note 1)
t12	READYI# Hold Time	4		ns	
t13a	NCA# Setup Time	15		ns	(Notes 15, 19)
t13b	LBA#, X16# Setup Time	7		ns	
t14	LBA#, NCA#, X16# Hold Time	3		ns	
t15	RESET, BRESET Setup Time	10		ns	
t16	RESET, BRESET Hold Time	3		ns	
t17	NA# Valid Delay	4	27	ns	(25 pF Load) (Note 1)
t18	READYO# Valid Delay	4	21	ns	(25 pF Load) (Note 1)
t19	BRDYEN# Valid Delay	4	21	ns	
t21a	CALEN Valid Delay	4	21	ns	(Note 2)
t21b	CALEN Valid Delay	4	26	ns	(Notes 1, 3)
t22a	CWEA#, CWEB# Valid Delay	4	23	ns	(Notes 1, 4)
t22b	CWEA#, CWEB# Pulse Width	25		ns	(Notes 1, 5)
t22c	CWEA#, CWEB# Rising Delay	8	21	ns	(Notes 1, 4)
t23	CS0#–CS3# Valid Delay	9	29	ns	(Notes 1, 6)
t24	CT/R# Valid Delay	9	30	ns	(Notes 1, 7)
t25a	COEA#, COEB# Valid Delay	4	18	ns	(25 pF Load) (Note 8)
t25b	COEA#, COEB# Valid Delay	4	18	ns	(25 pF Load) (Notes 1, 9)
t25c	COEA#, COEB# Rising Delay	4	18	ns	(25 pF Load) (Notes 10, 16)
t26	CS0#–CS3# to CWE# Delay	25		ns	(Notes 1, 5)
t27	CWE# to CS0#–CS3# Delay	0		ns	
t28a	CWE# to CALEN	0		ns	
t28a	CWE# to CS#	0		ns	
t31	SA2–SA31 Setup Time	10		ns	
t32	SA2–SA31 Hold Time	3		ns	
t33	BADS# Valid Delay	4	21	ns	(Note 1)
t34	BADS# Float Delay	4	30	ns	
t35	BNA# Setup Time	7		ns	
t36	BNA# Hold Time	3		ns	

A.C. SPECIFICATION TABLES (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C

All outputs tested specified at a 50 pF load unless otherwise noted.

Table 9-4. A.C. Specifications (Continued)

Symbol	Parameter	82385-25		Units	Notes
		Min	Max		
t37a	BREADY# Setup Time	20		ns	(Notes 1, 11)
t37b	BREADY# Setup Time	20		ns	(Note 12)
t38	BREADY# Hold	3		ns	
t40a	BACP Rising Delay	4	16	ns	
t40b	BACP Falling Delay	4	20	ns	
t41	BAOE# Valid Delay	4	15	ns	
t43a	BT/R# Valid Delay	4	16	ns	
t43b	DOE# Falling Delay	4	20	ns	
t43ab	DOE# Rising Delay	4	17	ns	(Note 17)
t43c	LDSTB Valid Delay	4	21	ns	
t44a	SEN Setup Time	9		ns	
t44b	SSTB# Setup Time	5		ns	
t45	SEN, SSTB# Hold Time	5		ns	
t46	BHOLD Setup Time	15		ns	(Note 13)
t47	BHOLD Hold Time	3		ns	(Note 13)
t48	BHOLDA Valid Delay	4	23	ns	(Note 13)
t55a	BLOCK# Valid Delay	3	26	ns	(Notes 1, 18)
t55b	BBE# Valid Delay	4	28	ns	(Notes 1, 18)
t56	MISS# Valid Delay	4	30	ns	(Note 1)
t57	MISS#, BBE#, BLOCK# Float Delay	4	30	ns	
t58	WBS Valid Delay	4	25	ns	(Note 1)
t59	FLUSH Setup Time	12		ns	
t60	FLUSH Hold Time	5		ns	
t61	FLUSH Setup to RESET Low	21		ns	
t62	FLUSH Hold from RESET Low	21		ns	

NOTES:

1. Frequency dependent specifications.
2. All cycles except cache hit. CALEN triggers PHI2 in T1P state.
3. The end of cache write hit cycles. Triggered by PHI1.
4. CWE# transitions by PHI1 in cache write cycles. CWE# transitions by PHI2 in cache read miss cycles.
5. Used for cache memory (SRAM) specifications.
6. In cache write hit cycles, CS0#–CS3# transition high by PHI2 and low by PHI1. In cache write miss cycles, CS0#–CS3# transition high by PHI1 and low by PHI2.
7. In cache write hit cycles, CT/R# transition low by PHI2. In cache read hit cycles, CT/R# goes high by PHI2. In cache read miss cycles, CT/R# goes low by PHI1.
8. Direct mapped configuration.
9. Two way set associative configuration.
10. COE# switches high by PHI1 at the end of a cache read hit cycle.
11. Cache read miss cycles.
12. Non-cacheable read cycles and system write cycles.
13. Master mode configuration. BHOLD is an input and BHLDA is an output.
14. Not supported as listed in 82385 Data Sheet.
15. These inputs are directly affected by capacitive loading on the corresponding 80386 outputs. Use of the 80386 derating curve shows a maximum load of 80 pF–90 pF in order to meet the required 82385 setup times.
16.

Symbol	Temperature	Parameter Max
t25C	t_{case} = 0	16
	t_{case} = 85	18

17.

Symbol	Temperature	Parameter Max
t43B	t_{case} = 0	15
	t_{case} = 85	17

18. BLOCK# delay is either from BPHI1 or from 80386 LOCK#. Refer to Figure 5-3K and 5-3L in the 82385 data sheet.
19. NCA# setup time is now specified to the rising edge of PHI2 in the state after 80386 addresses become valid (either the first T2 or the state after the first T2P).

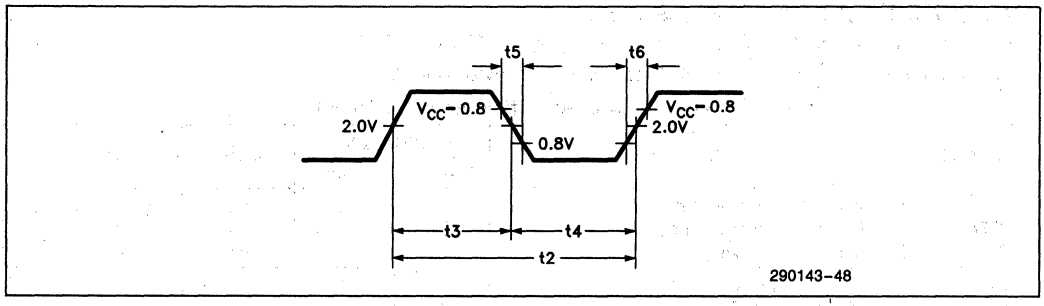


Figure 9-2. CLK2, BCLK2 Timing

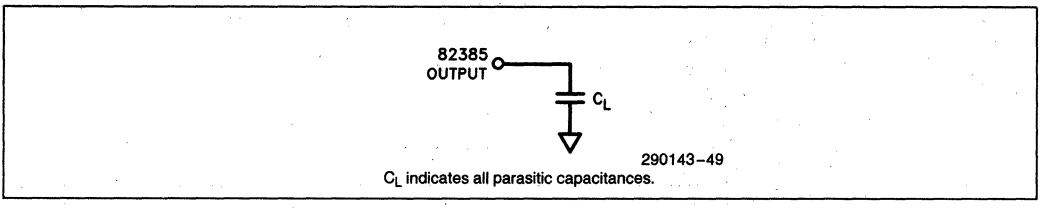
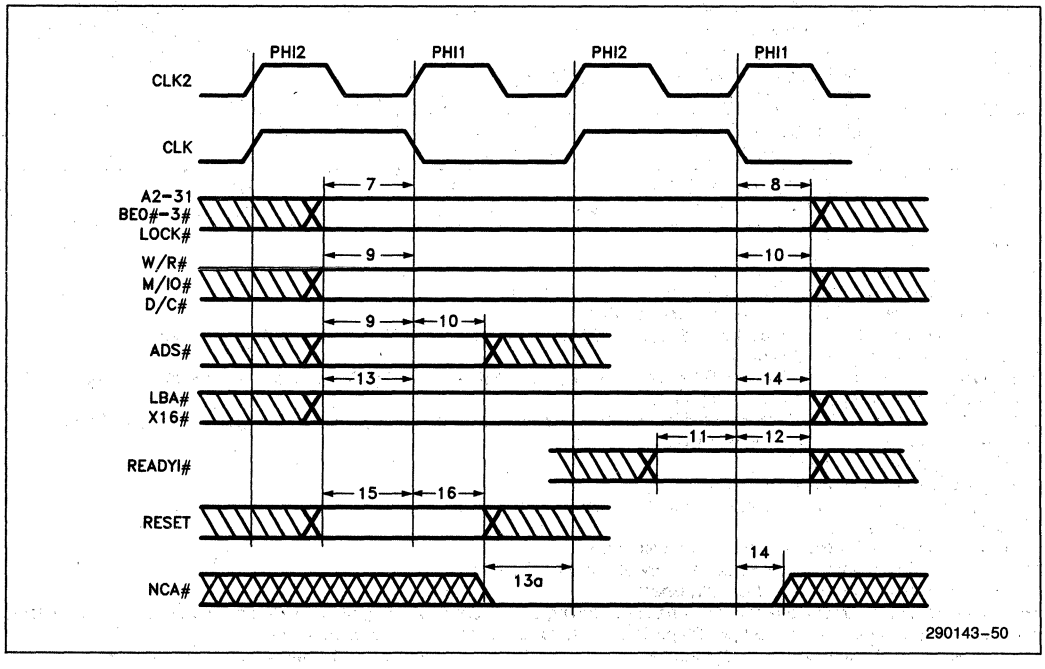
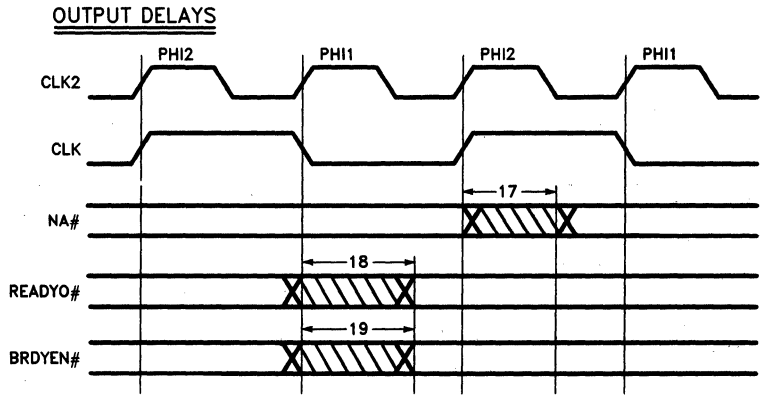


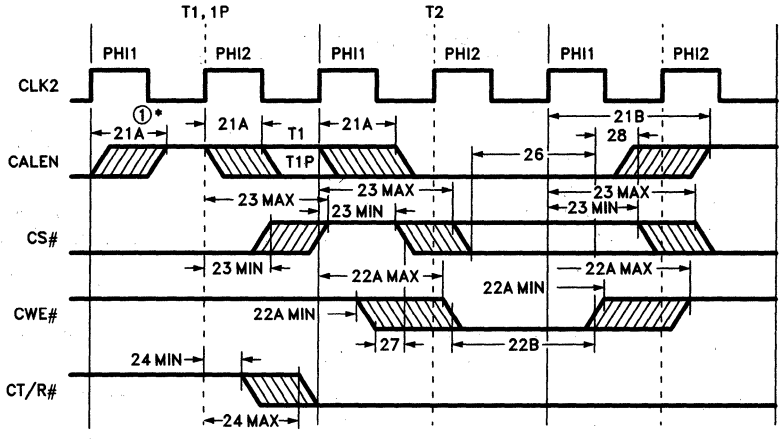
Figure 9-3. A.C. Test Load

386 Interface Parameters



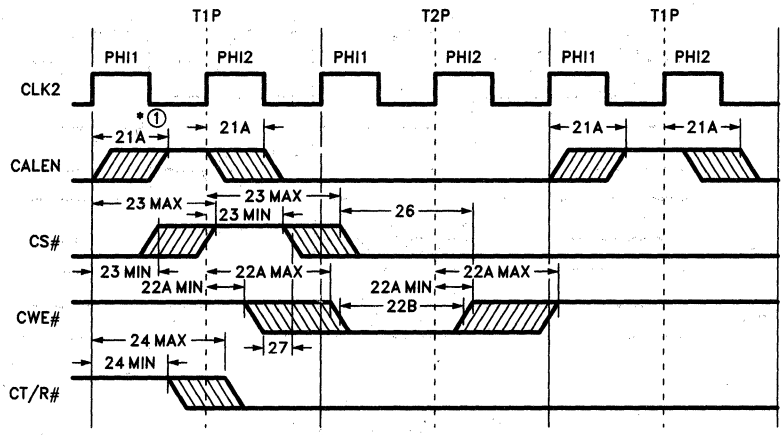


Cache Write Hit Cycle



①*. This would be 21B if previous bus cycle was Cache Write Hit cycle.

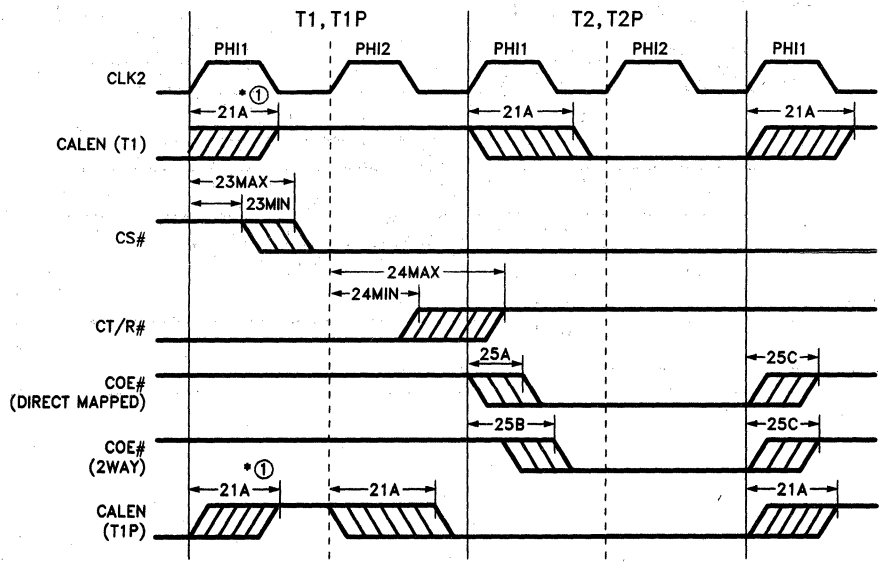
Cache Read Miss (Cache Update Cycle)



290143-53

*①. This would be 21B if previous bus cycle was Cache Write Hit cycle.

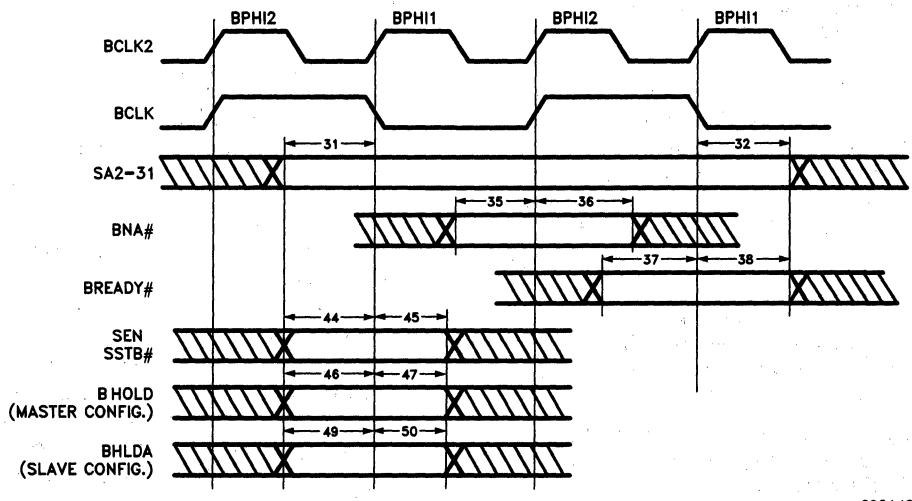
Cache Read Cycle



290143-54

*①. This would be 21B if previous bus cycle was Cache Write Hit cycle.

System Bus Interface Parameters



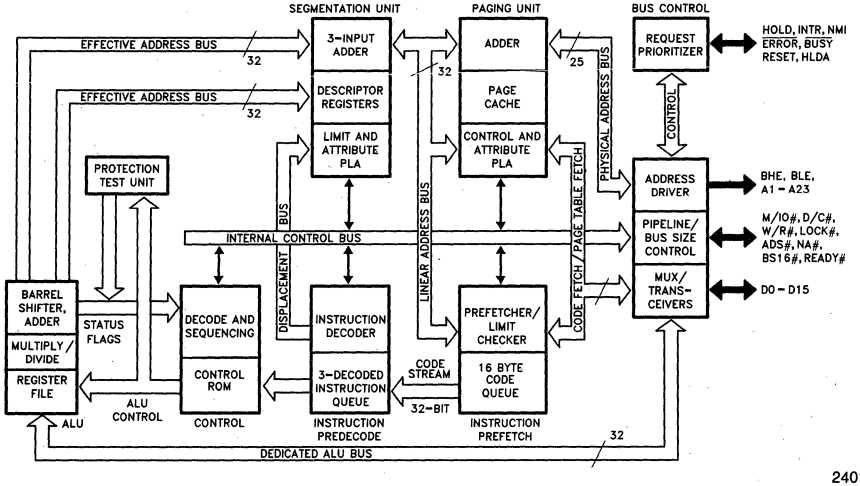
290143-55

*This would be 21B if previous cycle was Cache Write Hit.

386SX™ MICROPROCESSOR

- **Full 32-Bit Internal Architecture**
  - **— 8-, 16-, 32-Bit Data Types**
  - **— 8 General Purpose 32-Bit Registers**
- **Runs Intel386™ Software in a Cost Effective 16-Bit Hardware Environment**
  - **— Runs Same Applications and O.S.'s as the 386™ Processor**
  - **— Object Code Compatible with 8086, 80186, 80286, and 386™ Processors**
  - **— Runs MS-DOS*, OS/2* and UNIX****
- **Very High Performance 16-Bit Data Bus**
  - **— 16 MHz Clock**
  - **— Two-Clock Bus Cycles**
  - **— 16 Megabytes/Sec Bus Bandwidth**
  - **— Address Pipelining Allows Use of Slower/Cheaper Memories**
- **Integrated Memory Management Unit**
  - **— Virtual Memory Support**
  - **— Optional On-Chip Paging**
  - **— 4 Levels of Hardware Enforced Protection**
  - **— MMU Fully Compatible with Those of the 80286 and 386™ CPUs**
- **Virtual 8086 Mode Allows Execution of 8086 Software in a Protected and Paged System**
- **Large Uniform Address Space**
  - **— 16 Megabyte Physical**
  - **— 64 Terabyte Virtual**
  - **— 4 Gigabyte Maximum Segment Size**
- **High Speed Numerics Support with the 80387SX Coprocessor**
- **On-Chip Debugging Support Including Breakpoint Registers**
- **Complete System Development Support**
  - **— Software: C, PL/M, Assembler**
  - **— Debuggers: PMON-386, ICE™-386SX**
  - **— Extensive Third-Party Support: C, Pascal, FORTRAN, BASIC, Ada*** on VAX, UNIX**, MS-DOS*, and Other Hosts**
- **High Speed CHMOS III Technology**
- **100-Pin Plastic Quad Flatpack Package**
  (See Packaging Outlines and Dimensions #231369)

The 386SX™ Microprocessor is a 32-bit CPU with a 16-bit external data bus and a 24-bit external address bus. The 386SX CPU brings the high-performance software of the Intel386™ Architecture to midrange systems. It provides the performance benefits of a 32-bit programming architecture with the cost savings associated with 16-bit hardware systems.



240187-47

386SX™ Pipelined 32-Bit Microarchitecture

*MS-DOS and OS/2 are trademarks of Microsoft Corporation.
**UNIX is a trademark of AT&T.
***Ada is a trademark of the Department of Defense.

November 1988
Order Number: 240187-002

1.0 PIN DESCRIPTION

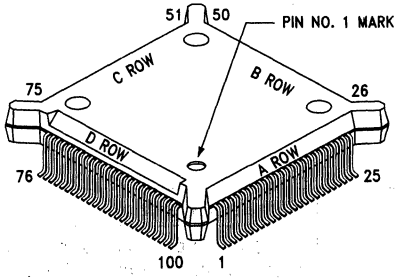


Figure 1.1. 386SX™ Microprocessor Pin out Top View

Table 1.1. Pin Assignments

A Row		B Row		C Row		D Row	
Pin	**Label**	**Pin**	**Label**	**Pin**	**Label**	**Pin**	**Label**
1	D_0	26	LOCK#	51	A_2	76	A_{21}
2	V_{SS}	27	N/C	52	A_3	77	V_{SS}
3	HLDA	28	N/C	53	A_4	78	V_{SS}
4	HOLD	29	N/C	54	A_5	79	A_{22}
5	V_{SS}	30	N/C	55	A_6	80	A_{23}
6	NA#	31	N/C	56	A_7	81	D_{15}
7	READY#	32	V_{CC}	57	V_{CC}	82	D_{14}
8	V_{CC}	33	RESET	58	A_8	83	D_{13}
9	V_{CC}	34	BUSY#	59	A_9	84	V_{CC}
10	V_{CC}	35	V_{SS}	60	A_{10}	85	V_{SS}
11	V_{SS}	36	ERROR#	61	A_{11}	86	D_{12}
12	V_{SS}	37	PEREQ	62	A_{12}	87	D_{11}
13	V_{SS}	38	NMI	63	V_{SS}	88	D_{10}
14	V_{SS}	39	V_{CC}	64	A_{13}	89	D_9
15	CLK2	40	INTR	65	A_{14}	90	D_8
16	ADS#	41	V_{SS}	66	A_{15}	91	V_{CC}
17	BLE#	42	V_{CC}	67	V_{SS}	92	D_7
18	A_1	43	N/C	68	V_{SS}	93	D_6
19	BHE#	44	N/C	69	V_{CC}	94	D_5
20	N/C	45	N/C	70	A_{16}	95	D_4
21	V_{CC}	46	N/C	71	V_{CC}	96	D_3
22	V_{SS}	47	N/C	72	A_{17}	97	V_{CC}
23	M/IO#	48	V_{CC}	73	A_{18}	98	V_{SS}
24	D/C#	49	V_{SS}	74	A_{19}	99	D_2
25	W/R#	50	V_{SS}	75	A_{20}	100	D_1

1.0 PIN DESCRIPTION (Continued)

The following are the 386SX™ Microprocessor pin descriptions. The following definitions are used in the pin descriptions:

The named signal is active LOW.
I Input signal.
O Output signal.
I/O Input and Output signal.
- No electrical connection.

Symbol	Type	Pin	Name and Function
CLK2	I	15	**CLK2** provides the fundamental timing for the 386SX™ Microprocessor. For additional information see **Clock** (page 39).
RESET	I	33	**RESET** suspends any operation in progress and places the 386SX™ Microprocessor in a known reset state. See **Interrupt Signals** (page 43) for additional information.
D_{15}–D_0	I/O	81-83,86-90, 92-96,99-100,1	**Data Bus** inputs data during memory, I/O and interrupt acknowledge read cycles and outputs data during memory and I/O write cycles. See **Data Bus** (page 39) for additional information.
A_{23}–A_1	O	80-79,76-72,70, 66-64,62-58, 56-51,18	**Address Bus** outputs physical memory or port I/O addresses. See **Address Bus** (page 40) for additional information.
W/R#	O	25	**Write/Read** is a bus cycle definition pin that distinguishes write cycles from read cycles. See **Bus Cycle Definition Signals** (page 40) for additional information.
D/C#	O	24	**Data/Control** is a bus cycle definition pin that distinguishes data cycles, either memory or I/O, from control cycles which are: interrupt acknowledge, halt, and code fetch. See **Bus Cycle Definition Signals** (page 40) for additional information.
M/IO#	O	23	**Memory/IO** is a bus cycle definition pin that distinguishes memory cycles from input/output cycles. See **Bus Cycle Definition Signals** (page 40) for additional information.
LOCK#	O	26	**Bus Lock** is a bus cycle definition pin that indicates that other system bus masters are not to gain control of the system bus while it is active. See **Bus Cycle Definition Signals** (page 40) for additional information.
ADS#	O	16	**Address Status** indicates that a valid bus cycle definition and address (W/R#, D/C#, M/IO#, BHE#, BLE# and A_{23}–A_1 are being driven at the 386SX™ Microprocessor pins. See **Bus Control Signals** (page 41) for additional information.
NA#	I	6	**Next Address** is used to request address pipelining. See **Bus Control Signals** (page 41) for additional information.
READY#	I	7	**Bus Ready** terminates the bus cycle. See **Bus Control Signals** (page 41) for additional information.
BHE#, BLE#	O	19,17	**Byte Enables** indicate which data bytes of the data bus take part in a bus cycle. See **Address Bus** (page 40) for additional information.

1.0 PIN DESCRIPTION (Continued)

Symbol	Type	Pin	Name and Function
HOLD	I	4	**Bus Hold Request** input allows another bus master to request control of the local bus. See **Bus Arbitration Signals** (page 41) for additional information.
HLDA	O	3	**Bus Hold Acknowledge** output indicates that the 386SX™ Microprocessor has surrendered control of its local bus to another bus master. See **Bus Arbitration Signals** (page 41) for additional information.
INTR	I	40	**Interrupt Request** is a maskable input that signals the 386SX™ Microprocessor to suspend execution of the current program and execute an interrupt acknowledge function. See **Interrupt Signals** (page 43) for additional information.
NMI	I	38	**Non-Maskable Interrupt Request** is a non-maskable input that signals the 386SX™ Microprocessor to suspend execution of the current program and execute an interrupt acknowledge function. See **Interrupt Signals** (page 43) for additional information.
BUSY#	I	34	**Busy** signals a busy condition from a processor extension. See **Coprocessor Interface Signals** (page 42) for additional information.
ERROR#	I	36	**Error** signals an error condition from a processor extension. See **Coprocessor Interface Signals** (page 42) for additional information.
PEREQ	I	37	**Processor Extension Request** indicates that the processor has data to be transferred by the 386SX™ Microprocessor. See **Coprocessor Interface Signals** (page 42) for additional information.
N/C	-	20, 27-31, 43-47	**No Connects** should always be left unconnected. Connection of a N/C pin may cause the processor to malfunction or be incompatible with future steppings of the 386SX™ Microprocessor.
V_{CC}	I	8-10,21,32,39 42,48,57,69, 71,84,91,97	**System Power** provides the +5V nominal DC supply input.
V_{SS}	I	2,5,11-14,22 35,41,49-50, 63,67-68, 77-78,85,98	**System Ground** provides the 0V connection from which all inputs and outputs are measured.

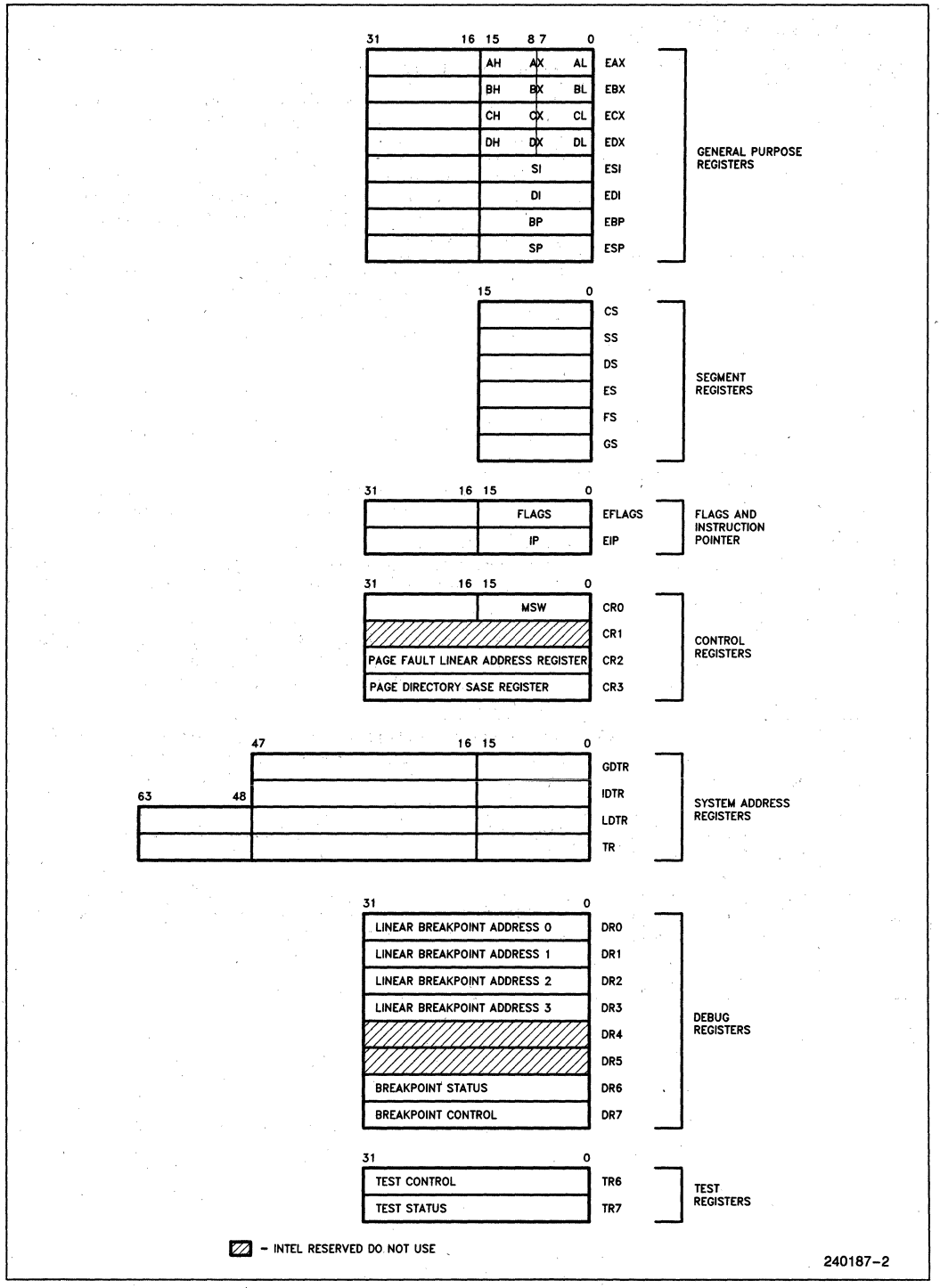


Figure 2.1. 386SX™ Microprocessor Registers

INTRODUCTION

The 386SX™ Microprocessor is 100% object code compatible with the 386, 286 and 8086 microprocessors. System manufacturers can provide 386™ CPU based systems optimized for performance and 386SX CPU based systems optimized for cost, both sharing the same operating systems and application software. Systems based on the 386SX CPU can access the world's largest existing microcomputer software base, including the growing 32-bit software base. Only the Intel386 architecture can run UNIX, OS/2 and MS-DOS.

Instruction pipelining, high bus bandwidth, and a very high performance ALU ensure short average instruction execution times and high system throughput. The 386SX processor is capable of execution at sustained rates of 2.5–3.0 million instructions per second.

The integrated memory management unit (MMU) includes an address translation cache, advanced multi-tasking hardware, and a four-level hardware-enforced protection mechanism to support operating systems. The virtual machine capability of the 386SX CPU allows simultaneous execution of applications from multiple operating systems such as MS-DOS and UNIX.

The 386SX CPU offers on-chip testability and debugging features. Four breakpoint registers allow conditional or unconditional breakpoint traps on code execution or data accesses for powerful debugging of even ROM-based systems. Other testability features include self-test, tri-state of output buffers, and direct access to the page translation cache.

2.0 BASE ARCHITECTURE

The 386SX Microprocessor consists of a central processing unit, a memory management unit and a bus interface.

The central processing unit consists of the execution unit and the instruction unit. The execution unit contains the eight 32-bit general purpose registers which are used for both address calculation and data operations and a 64-bit barrel shifter used to speed shift, rotate, multiply, and divide operations. The instruction unit decodes the instruction opcodes and stores them in the decoded instruction queue for immediate use by the execution unit.

The memory management unit (MMU) consists of a segmentation unit and a paging unit. Segmentation allows the managing of the logical address space by providing an extra addressing component, one that allows easy code and data relocatability, and efficient sharing. The paging mechanism operates beneath and is transparent to the segmentation process, to allow management of the physical address space.

The segmentation unit provides four levels of protection for isolating and protecting applications and the operating system from each other. The hardware enforced protection allows the design of systems with a high degree of integrity.

The 386SX Microprocessor has two modes of operation: Real Address Mode (Real Mode), and Protected Virtual Address Mode (Protected Mode). In Real Mode the 386SX Microprocessor operates as a very fast 8086, but with 32-bit extensions if desired. Real Mode is required primarily to set up the processor for Protected Mode operation.

Within Protected Mode, software can perform a task switch to enter into tasks designated as Virtual 8086 Mode tasks. Each such task behaves with 8086 semantics, thus allowing 8086 software (an application program or an entire operating system) to execute. The Virtual 8086 tasks can be isolated and protected from one another and the host 386SX Microprocessor operating system by use of paging.

Finally, to facilitate high performance system hardware designs, the 386SX Microprocessor bus interface offers address pipelining and direct Byte Enable signals for each byte of the data bus.

2.1 Register Set

The 386SX Microprocessor has thirty-four registers as shown in Figure 2-1. These registers are grouped into the following seven categories:

General Purpose Registers: The eight 32-bit general purpose registers are used to contain arithmetic and logical operands. Four of these (EAX, EBX, ECX, and EDX) can be used either in their entirety as 32-bit registers, as 16-bit registers, or split into pairs of separate 8-bit registers.

Segment Registers: Six 16-bit special purpose registers select, at any given time, the segments of memory that are immediately addressable for code, stack, and data.

Flags and Instruction Pointer Registers: The two 32-bit special purpose registers in figure 2.1 record or control certain aspects of the 386SX Microprocessor state. The EFLAGS register includes status and control bits that are used to reflect the outcome of many instructions and modify the semantics of

some instructions. The Instruction Pointer, called EIP, is 32 bits wide. The Instruction Pointer controls instruction fetching and the processor automatically increments it after executing an instruction.

Control Registers: The four 32-bit control register are used to control the global nature of the 386SX Microprocessor. The CR0 register contains bits that set the different processor modes (Protected, Real, Paging and Coprocessor Emulation). CR2 and CR3 registers are used in the paging operation.

System Address Registers: These four special registers reference the tables or segments supported by the 80286/386SX/386 CPU's protection model. These tables or segments are:

GDTR (Global Descriptor Table Register),
IDTR (Interrupt Descriptor Table Register),
LDTR (Local Descriptor Table Register),
TR (Task State Segment Register).

Debug Registers: The six programmer accessible debug registers provide on-chip support for debugging. The use of the debug registers is described in Section 2.10 **Debugging Support**.

Test Registers: Two registers are used to control the testing of the RAM/CAM (Content Addressable Memories) in the Translation Lookaside Buffer portion of the 386SX Microprocessor. Their use is discussed in **Testability**.

EFLAGS REGISTER

The flag register is a 32-bit register named EFLAGS. The defined bits and bit fields within EFLAGS, shown in Figure 2.2, control certain operations and indicate the status of the 386SX Microprocessor. The lower 16 bits (bits 0–15) of EFLAGS contain the 16- bit flag register named FLAGS. This is the default flag register used when executing 8086, 80286, or real mode code. The functions of the flag bits are given in Table 2.1.

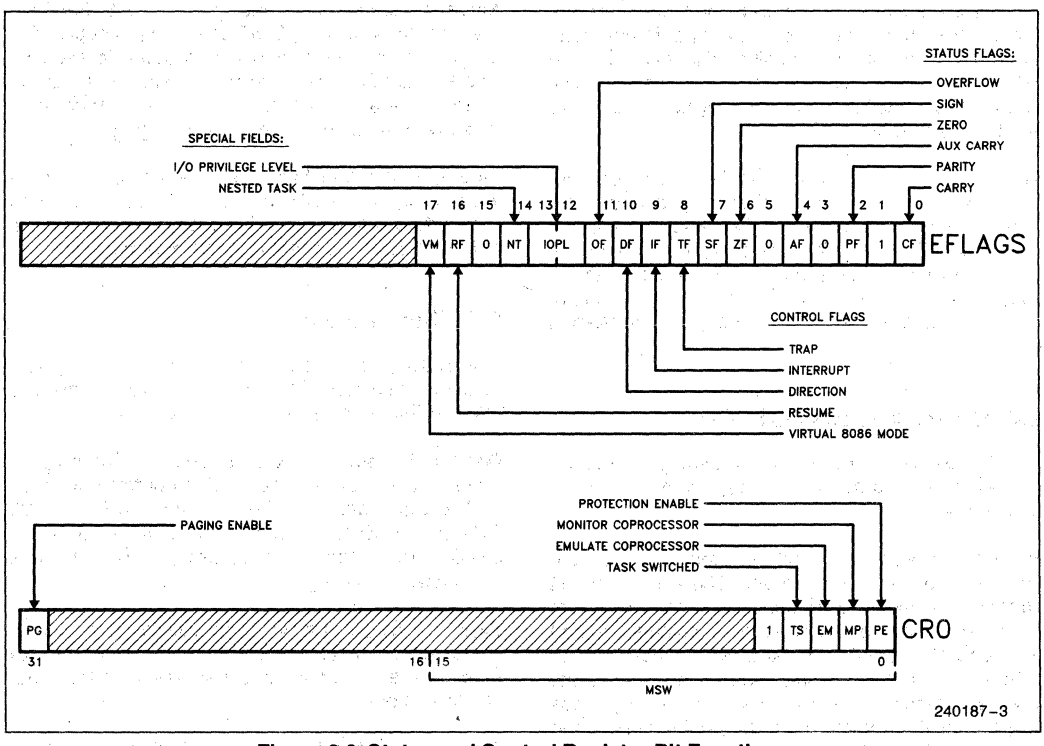


Figure 2.2. Status and Control Register Bit Functions

Table 2.1. Flag Definitions

Bit Position	Name	Function
0	CF	Carry Flag—Set on high-order bit carry or borrow; cleared otherwise.
2	PF	Parity Flag—Set if low-order 8 bits of result contain an even number of 1-bits; cleared otherwise.
4	AF	Auxiliary Carry Flag—Set on carry from or borrow to the low order four bits of AL; cleared otherwise.
6	ZF	Zero Flag—Set if result is zero; cleared otherwise.
7	SF	Sign Flag—Set equal to high-order bit of result (0 if positive, 1 if negative).
8	TF	Single Step Flag—Once set, a single step interrupt occurs after the next instruction executes. TF is cleared by the single step interrupt.
9	IF	Interrupt-Enable Flag—When set, maskable interrupts will cause the CPU to transfer control to an interrupt vector specified location.
10	DF	Direction Flag—Causes string instructions to auto-increment (default) the appropriate index registers when cleared. Setting DF causes auto-decrement.
11	OF	Overflow Flag—Set if the operation resulted in a carry/borrow into the sign bit (high-order bit) of the result but did not result in a carry/borrow out of the high-order bit or vice-versa.
12,13	IOPL	I/O Privilege Level—Indicates the maximum CPL permitted to execute I/O instructions without generating an exception 13 fault or consulting the I/O permission bit map while executing in protected mode. For virtual 86 mode it indicates the maximum CPL allowing alteration of the IF bit.
14	NT	Nested Task—Indicates that the execution of the current task is nested within another task.
16	RF	Resume Flag—Used in conjunction with debug register breakpoints. It is checked at instruction boundaries before breakpoint processing. If set, any debug fault is ignored on the next instruction.
17	VM	Virtual 8086 Mode—If set while in protected mode, the 386SX™ Microprocessor will switch to virtual 8086 operation, handling segment loads as the 8086 does, but generating exception 13 faults on privileged opcodes.

CONTROL REGISTERS

The 386SX Microprocessor has three control registers of 32 bits, CR0, CR2 and CR3, to hold the machine state of a global nature. These registers are shown in Figures 2.1 and 2.2. The defined CR0 bits are described in Table 2.2.

Table 2.2. CR0 Definitions

Bit Position	Name	Function
0	PE	Protection mode enable—places the 386SX™ Microprocessor into protected mode. If PE is reset, the processor operates again in Real Mode. PE may be set by loading MSW or CR0. PE can be reset only by loading CR0, it cannot be reset by the LMSW instruction.
1	MP	Monitor coprocessor extension—allows WAIT instructions to cause a processor extension not present exception (number 7).
2	EM	Emulate processor extension—causes a processor extension not present exception (number 7) on ESC instructions to allow emulating a processor extension.
3	TS	Task switched—indicates the next instruction using a processor extension will cause exception 7, allowing software to test whether the current processor extension context belongs to the current task.
31	PG	Paging enable bit—is set to enable the on-chip paging unit. It is reset to disable the on-chip paging unit.

2.2 Instruction Set

The instruction set is divided into nine categories of operations:

Data Transfer
Arithmetic
Shift/Rotate
String Manipulation
Bit Manipulation
Control Transfer
High Level Language Support
Operating System Support
Processor Control

These instructions are listed in Table 9.1 **Instruction Set Clock Count Summary**.

All 386SX Microprocessor instructions operate on either 0, 1, 2 or 3 operands; an operand resides in a register, in the instruction itself, or in memory. Most zero operand instructions (e.g CLI, STI) take only one byte. One operand instructions generally are two bytes long. The average instruction is 3.2 bytes long. Since the 386SX Microprocessor has a 16 byte prefetch instruction queue, an average of 5 instructions will be prefetched. The use of two operands permits the following types of common instructions:

Register to Register
Memory to Register
Immediate to Register
Memory to Memory
Register to Memory
Immediate to Memory.

The operands can be either 8, 16, or 32 bits long. As a general rule, when executing code written for the 386SX Microprocessor (32 bit code), operands are 8 or 32 bits; when executing existing 8086 or 80286 code (16-bit code), operands are 8 or 16 bits. Prefixes can be added to all instructions which override the default length of the operands (i.e. use 32-bit operands for 16-bit code, or 16-bit operands for 32-bit code).

2.3 Memory Organization

Memory on the 386SX Microprocessor is divided into 8-bit quantities (bytes), 16-bit quantities (words), and 32-bit quantities (dwords). Words are stored in two consecutive bytes in memory with the low-order byte at the lowest address. Dwords are stored in four consecutive bytes in memory with the low-order byte at the lowest address. The address of a word or dword is the byte address of the low-order byte.

In addition to these basic data types, the 386SX Microprocessor supports two larger units of memory: pages and segments. Memory can be divided up into one or more variable length segments, which can be swapped to disk or shared between programs. Memory can also be organized into one or more 4K byte pages. Finally, both segmentation and paging can be combined, gaining the advantages of both systems. The 386SX Microprocessor supports both pages and segmentation in order to provide maximum flexibility to the system designer. Segmentation and paging are complementary. Segmentation is useful for organizing memory in logical modules, and as such is a tool for the application programmer, while pages are useful to the system programmer for managing the physical memory of a system.

ADDRESS SPACES

The 386SX Microprocessor has three types of address spaces: **logical**, **linear**, and **physical**. A **logical** address (also known as a **virtual** address) consists of a selector and an offset. A selector is the contents of a segment register. An offset is formed by summing all of the addressing components (BASE, INDEX, DISPLACEMENT), discussed in section 2.4 **Addressing Modes**, into an effective address. This effective address along with the selector is known as the logical address. Since each task on the 386SX Microprocessor has a maximum of 16K ($2^{14} - 1$) selectors, and offsets can be 4 gigabytes (with paging enabled) this gives a total of 2^{46} bits, or 64 terabytes, of **logical** address space per task. The programmer sees the logical address space.

The segmentation unit translates the **logical** address space into a 32-bit **linear** address space. If the paging unit is not enabled then the 32-bit **linear** address is truncated into a 24-bit **physical** address. The **physical address** is what appears on the address pins.

The primary differences between Real Mode and Protected Mode are how the segmentation unit performs the translation of the **logical** address into the **linear** address, size of the address space, and paging capability. In Real Mode, the segmentation unit shifts the selector left four bits and adds the result to the effective address to form the **linear** address. This **linear** address is limited to 1 megabyte. In addition, real mode has no paging capability.

Protected Mode will see one of two different address spaces, depending on whether or not paging is enabled. Every selector has a **logical base** address associated with it that can be up to 32 bits in length. This 32-bit **logical base** address is added to the effective address to form a final 32-bit **linear**

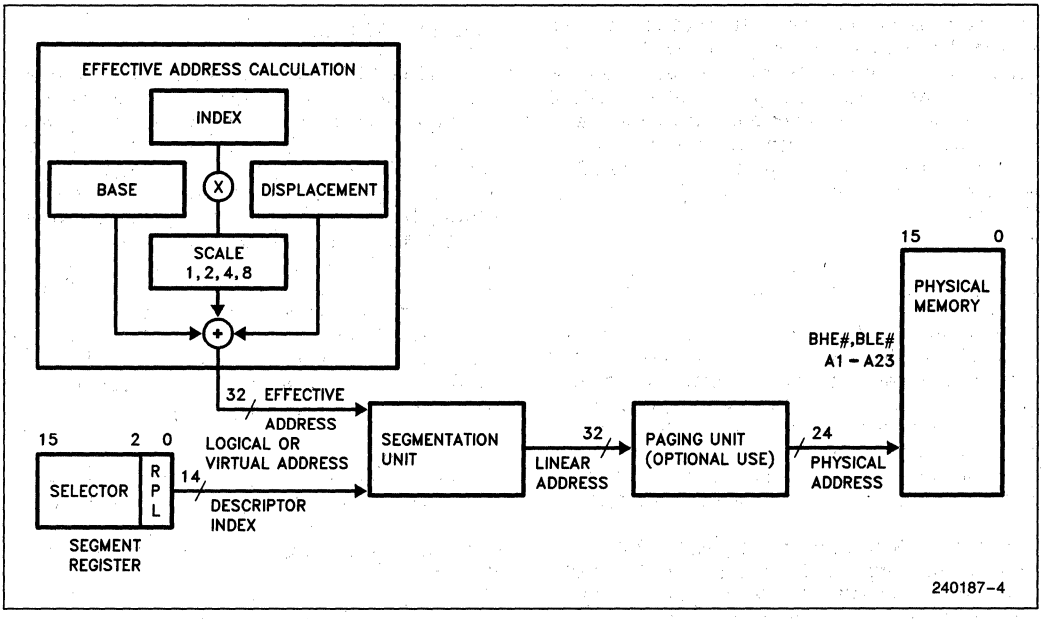


Figure 2.3. Address Translation

address. If paging is disabled this final **linear** address reflects physical memory and is truncated so that only the lower 24 bits of this address are used to address the 16 megabyte memory address space. If paging is enabled this final **linear** address reflects a 32-bit address that is translated through the paging unit to form a 16-megabyte physical address. The **logical base** address is stored in one of two operating system tables (i.e. the Local Descriptor Table or Global Descriptor Table).

Figure 2.3 shows the relationship between the various address spaces.

SEGMENT REGISTER USAGE

The main data structure used to organize memory is the segment. On the 386SX Microprocessor, segments are variable sized blocks of linear addresses which have certain attributes associated with them. There are two main types of segments, code and data. The segments are of variable size and can be as small as 1 byte or as large as 4 gigabytes (2^{32} bits).

In order to provide compact instruction encoding and increase processor performance, instructions do not need to explicitly specify which segment register is used. The segment register is automatically chosen according to the rules of Table 2.3 (Segment Register Selection Rules). In general, data references use the selector contained in the DS register, stack references use the SS register and instruction fetches use the CS register. The contents of the Instruction Pointer provide the offset. Special segment override prefixes allow the explicit use of a given segment register, and override the implicit rules listed in Table 2.3. The override prefixes also allow the use of the ES, FS and GS segment registers.

There are no restrictions regarding the overlapping of the base addresses of any segments. Thus, all 6 segments could have the base address set to zero and create a system with a four gigabyte linear address space. This creates a system where the virtual address space is the same as the linear address space. Further details of segmentation are discussed in chapter 4 **PROTECTED MODE ARCHITECTURE.**

2.4 Addressing Modes

The 386SX Microprocessor provides a total of 8 addressing modes for instructions to specify operands. The addressing modes are optimized to allow the efficient execution of high level languages such as C and FORTRAN, and they cover the vast majority of data references needed by high-level languages.

REGISTER AND IMMEDIATE MODES

Two of the addressing modes provide for instructions that operate on register or immediate operands:

Table 2.3. Segment Register Selection Rules

Type of Memory Reference	Implied (Default) Segment Use	Segment Override Prefixes Possible
Code Fetch	CS	None
Destination of PUSH, PUSHA instructons	SS	None
Source of POP, POPA instructions	SS	None
Destination of STOS, MOVE, REP STOS, and REP MOVS instructions	ES	None
Other data references, with effective address using base register of:		
[EAX]	DS	CS,SS,ES,FS,GS
[EBX]	DS	CS,SS,ES,FS,GS
[ECX]	DS	CS,SS,ES,FS,GS
[EDX]	DS	CS,SS,ES,FS,GS
[ESI]	DS	CS,SS,ES,FS,GS
[EDI]	DS	CS,SS,ES,FS,GS
[EBP]	SS	CS,DS,ES,FS,GS
[ESP]	SS	CS,DS,ES,FS,GS

Register Operand Mode: The operand is located in one of the 8, 16 or 32-bit general registers.

Immediate Operand Mode: The operand is included in the instruction as part of the opcode.

32-BIT MEMORY ADDRESSING MODES

The remaining 6 modes provide a mechanism for specifying the effective address of an operand. The linear address consists of two components: the segment base address and an effective address. The effective address is calculated by summing any combination of the following three address elements (see figure 2.3):

DISPLACEMENT: an 8, 16 or 32-bit immediate value, following the instruction.

BASE: The contents of any general purpose register. The base registers are generally used by compilers to point to the start of the local variable area.

INDEX: The contents of any general purpose register except for ESP. The index registers are used to access the elements of an array, or a string of characters. The index register's value can be multiplied by a scale factor, either 1, 2, 4 or 8. The scaled index is especially useful for accessing arrays or structures.

Combinations of these 3 components make up the 6 additional addressing modes. There is no performance penalty for using any of these addressing combinations, since the effective address calculation is pipelined with the execution of other instructions. The one exception is the simultaneous use of Base and Index components which requires one additional clock.

As shown in Figure 2.4, the effective address (EA) of an operand is calculated according to the following formula:

$$EA = Base_{Register} + (Index_{Register} * scaling) + Displacement$$

1. **Direct Mode:** The operand's offset is contained as part of the instruction as an 8, 16 or 32-bit displacement.
2. **Register Indirect Mode:** A BASE register contains the address of the operand.
3. **Based Mode:** A BASE register's contents are added to a DISPLACEMENT to form the operand's offset.
4. **Scaled Index Mode:** An INDEX register's contents are multiplied by a SCALING factor, and the result is added to a DISPLACEMENT to form the operand's offset.

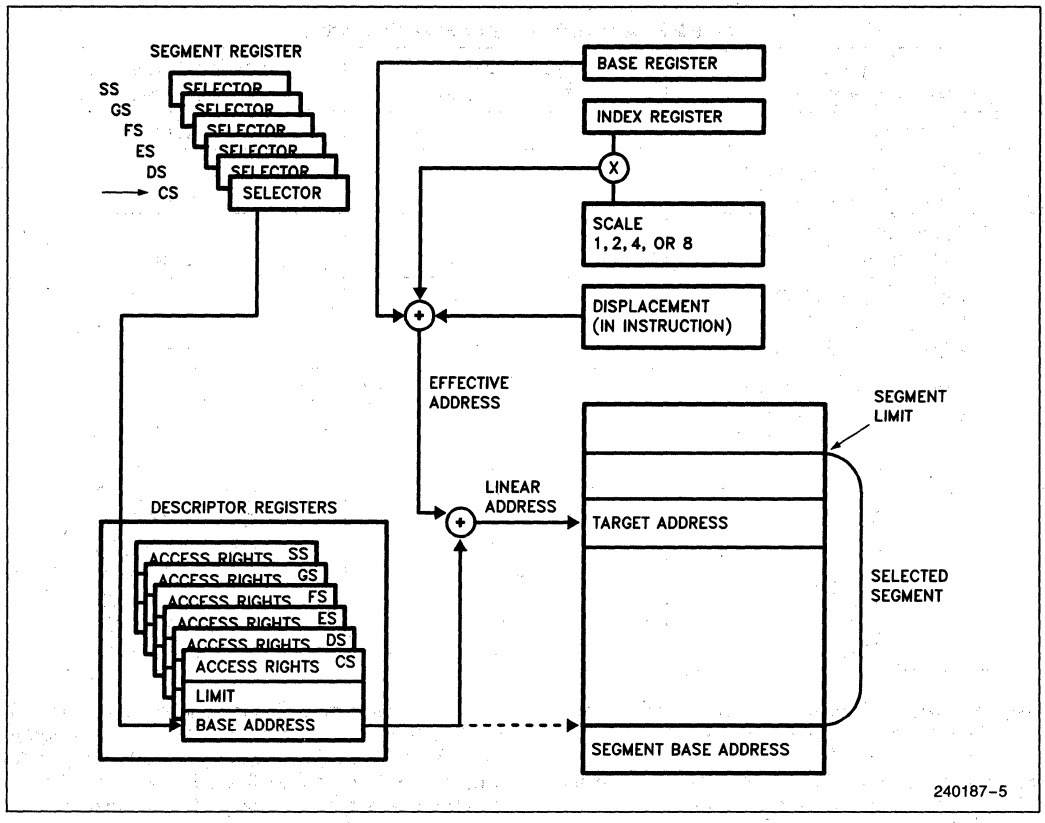


Figure 2.4. Addressing Mode Calculations

5. **Based Scaled Index Mode:** The contents of an INDEX register are multiplied by a SCALING factor, and the result is added to the contents of a BASE register to obtain the operand's offset.
6. **Based Scaled Index Mode with Displacement:** The contents of an INDEX register are multiplied by a SCALING factor, and the result is added to the contents of a BASE register and a DISPLACEMENT to form the operand's offset.

DIFFERENCES BETWEEN 16 AND 32 BIT ADDRESSES

In order to provide software compatibility with the 8086 and the 80286, the 386SX Microprocessor can execute 16-bit instructions in Real and Protected Modes. The processor determines the size of the instructions it is executing by examining the D bit in a Segment Descriptor. If the D bit is 0 then all operand lengths and effective addresses are assumed to be 16 bits long. If the D bit is 1 then the default length for operands and addresses is 32 bits. In Real Mode the default size for operands and addresses is 16 bits.

Regardless of the default precision of the operands or addresses, the 386SX Microprocessor is able to execute either 16 or 32-bit instructions. This is specified through the use of override prefixes. Two prefixes, the **Operand Length Prefix** and the **Address Length Prefix**, override the value of the D bit on an individual instruction basis. These prefixes are automatically added by assemblers.

The Operand Length and Address Length Prefixes can be applied separately or in combination to any instruction. The Address Length Prefix does not allow addresses over 64K bytes to be accessed in Real Mode. A memory address which exceeds 0FFFFH will result in a General Protection Fault. An Address Length Prefix only allows the use of the additional 386SX Microprocessor addressing modes.

When executing 32-bit code, the 386SX Microprocessor uses either 8 or 32-bit displacements, and any register can be used as base or index registers. When executing 16-bit code, the displacements are either 8 or 16-bits, and the base and index register conform to the 80286 model. Table 2.4 illustrates the differences.

Table 2.4. BASE and INDEX Registers for 16- and 32-Bit Addresses

	16-Bit Addressing	32-Bit Addressing
BASE REGISTER	BX,BP	Any 32-bit GP Register
INDEX REGISTER	SI,DI	Any 32-bit GP Register Except ESP
SCALE FACTOR	None	1, 2, 4, 8
DISPLACEMENT	0, 8, 16-bits	0, 8, 32-bits

2.5 Data Types

The 386SX Microprocessor supports all of the data types commonly used in high level languages:

Bit: A single bit quantity.

Bit Field: A group of up to 32 contiguous bits, which spans a maximum of four bytes.

Bit String: A set of contiguous bits; on the 386SX Microprocessor, bit strings can be up to 4 gigabits long.

Byte: A signed 8-bit quantity.

Unsigned Byte: An unsigned 8-bit quantity.

Integer (Word): A signed 16-bit quantity.

Long Integer (Double Word): A signed 32-bit quantity. All operations assume a 2's complement representation.

Unsigned Integer (Word): An unsigned 16-bit quantity.

Unsigned Long Integer (Double Word): An unsigned 32-bit quantity.

Signed Quad Word: A signed 64-bit quantity.

Unsigned Quad Word: An unsigned 64-bit quantity.

Pointer: A 16 or 32-bit offset-only quantity which indirectly references another memory location.

Long Pointer: A full pointer which consists of a 16-bit segment selector and either a 16 or 32-bit offset.

Char: A byte representation of an ASCII Alphanumeric or control character.

String: A contiguous sequence of bytes, words or dwords. A string may contain between 1 byte and 4 gigabytes

BCD: A byte (unpacked) representation of decimal digits 0–9.

Packed BCD: A byte (packed) representation of two decimal digits 0–9 storing one digit in each nibble.

When the 386SX Microprocessor is coupled with its numerics coprocessor, the 80387SX, then the following common floating point types are supported:

Floating Point: A signed 32, 64, or 80-bit real number representation. Floating point numbers are supported by the 80387SX numerics coprocessor.

Figure 2.5 illustrates the data types supported by the 386SX Microprocessor and the 80387SX.

2.6 I/O Space

The 386SX Microprocessor has two distinct physical address spaces: physical memory and I/O. Generally, peripherals are placed in I/O space although the 386SX Microprocessor also supports memory-mapped peripherals. The I/O space consists of 64K bytes which can be divided into 64K 8-bit ports or 32K 16-bit ports, or any combination of ports which add up to no more than 64K bytes. The 64K I/O address space refers to physical addresses rather than linear addresses since I/O instructions do not go through the segmentation or paging hardware. The M/IO# pin acts as an additional address line, thus allowing the system designer to easily determine which address space the processor is accessing.

The I/O ports are accessed by the IN and OUT instructions, with the port address supplied as an immediate 8-bit constant in the instruction or in the DX register. All 8-bit and 16-bit port addresses are zero extended on the upper address lines. The I/O instructions cause the M/IO# pin to be driven LOW. I/O port addresses 00F8H through 00FFH are reserved for use by Intel.

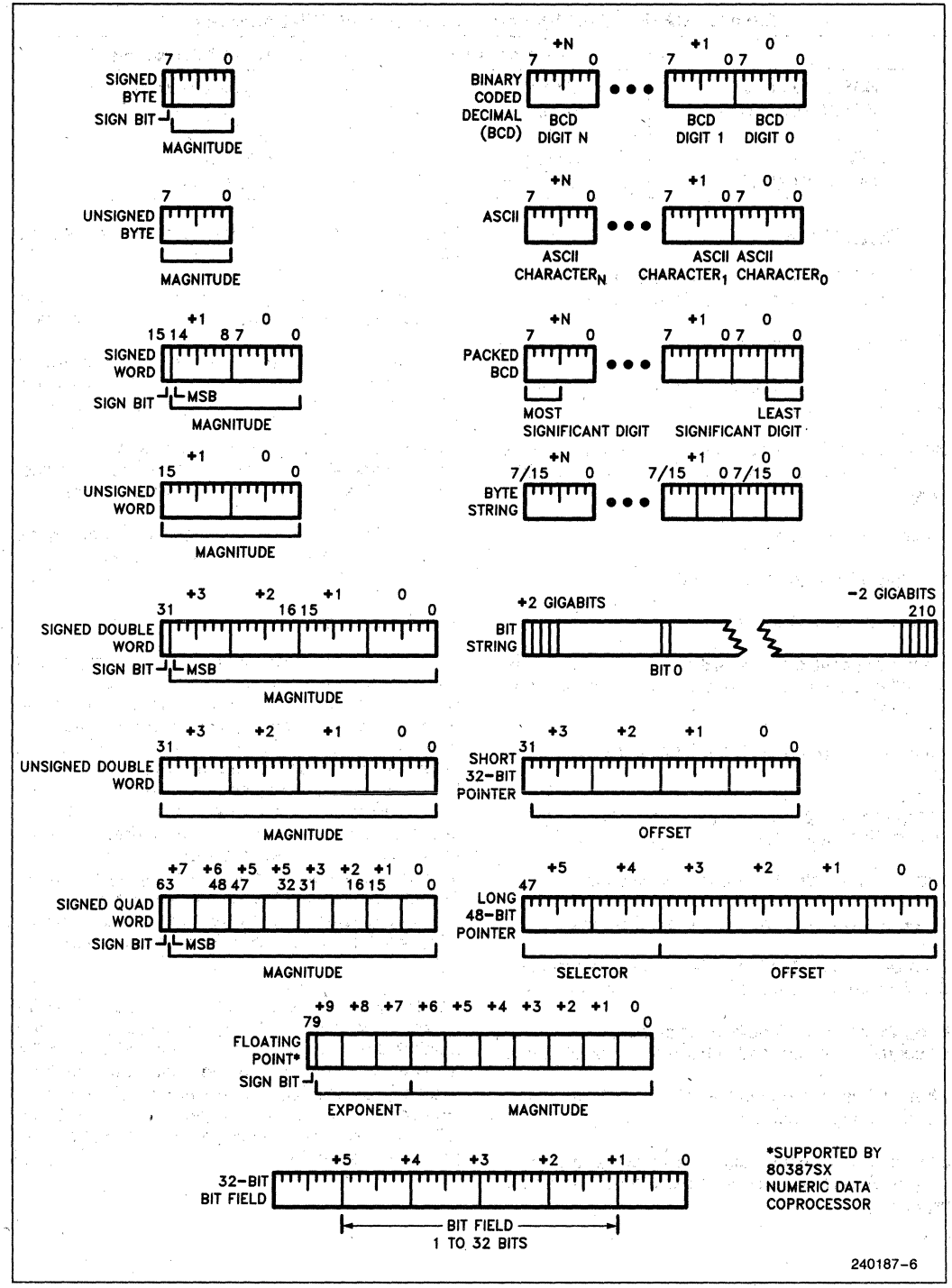


Figure 2.5. 386SX™ Microprocessor Supported Data Types

Table 2.5. Interrupt Vector Assignments

Function	Interrupt Number	Instruction Which Can Cause Exception	Return Address Points to Faulting Instruction	Type
Divide Error	0	DIV, IDIV	YES	FAULT
Debug Exception	1	any instruction	YES	TRAP*
NMI Interrupt	2	INT 2 or NMI	NO	NMI
One Byte Interrupt	3	INT	NO	TRAP
Interrupt on Overflow	4	INTO	NO	TRAP
Array Bounds Check	5	BOUND	YES	FAULT
Invalid OP-Code	6	Any illegal instruction	YES	FAULT
Device Not Available	7	ESC, WAIT	YES	FAULT
Double Fault	8	Any instruction that can generate an exception		ABORT
Coprocessor Segment Overrun	9	ESC	NO	ABORT
Invalid TSS	10	JMP, CALL, IRET, INT	YES	FAULT
Segment Not Present	11	Segment Register Instructions	YES	FAULT
Stack Fault	12	Stack References	YES	FAULT
General Protection Fault	13	Any Memory Reference	YES	FAULT
Page Fault	14	Any Memory Access or Code Fetch	YES	FAULT
Coprocessor Error	16	ESC, WAIT	YES	FAULT
Intel Reserved	17–32			
Two Byte Interrupt	0–255	INT n	NO	TRAP

*Some debug exceptions may report both traps on the previous instruction and faults on the next instruction.

2.7 Interrupts and Exceptions

Interrupts and exceptions alter the normal program flow in order to handle external events, report errors or exceptional conditions. The difference between interrupts and exceptions is that interrupts are used to handle asynchronous external events while exceptions handle instruction faults. Although a program can generate a software interrupt via an INT N instruction, the processor treats software interrupts as exceptions.

Hardware interrupts occur as the result of an external event and are classified into two types: maskable or non-maskable. Interrupts are serviced after the execution of the current instruction. After the interrupt handler is finished servicing the interrupt, execution proceeds with the instruction immediately **after** the interrupted instruction.

Exceptions are classified as faults, traps, or aborts, depending on the way they are reported and whether or not restart of the instruction causing the exception is supported. **Faults** are exceptions that are detected and serviced **before** the execution of the faulting instruction. **Traps** are exceptions that are reported immediately **after** the execution of the instruction which caused the problem. **Aborts** are exceptions which do not permit the precise location of the instruction causing the exception to be determined.

Thus, when an interrupt service routine has been completed, execution proceeds from the instruction immediately following the interrupted instruction. On the other hand, the return address from an exception fault routine will always point to the instruction causing the exception and will include any leading instruction prefixes. Table 2.5 summarizes the possible interrupts for the 386SX Microprocessor and shows where the return address points to.

The 386SX Microprocessor has the ability to handle up to 256 different interrupts/exceptions. In order to service the interrupts, a table with up to 256 interrupt vectors must be defined. The interrupt vectors are simply pointers to the appropriate interrupt service routine. In Real Mode, the vectors are 4-byte quantities, a Code Segment plus a 16-bit offset; in Protected Mode, the interrupt vectors are 8 byte quantities, which are put in an Interrupt Descriptor Table. Of the 256 possible interrupts, 32 are reserved for use by Intel and the remaining 224 are free to be used by the system designer.

INTERRUPT PROCESSING

When an interrupt occurs, the following actions happen. First, the current program address and Flags are saved on the stack to allow resumption of the interrupted program. Next, an 8-bit vector is supplied to the 386SX Microprocessor which identifies the appropriate entry in the interrupt table. The table contains the starting address of the interrupt service routine. Then, the user supplied interrupt service routine is executed. Finally, when an IRET instruction is executed the old processor state is restored and program execution resumes at the appropriate instruction.

The 8-bit interrupt vector is supplied to the 386SX Microprocessor in several different ways: exceptions supply the interrupt vector internally; software INT instructions contain or imply the vector; maskable hardware interrupts supply the 8-bit vector via the interrupt acknowledge bus sequence. Non-Maskable hardware interrupts are assigned to interrupt vector 2.

Maskable Interrupt

Maskable interrupts are the most common way to respond to asynchronous external hardware events. A hardware interrupt occurs when the INTR is pulled HIGH and the Interrupt Flag bit (IF) is enabled. The processor only responds to interrupts between instructions (string instructions have an 'interrupt window' between memory moves which allows interrupts during long string moves). When an interrupt occurs the processor reads an 8-bit vector supplied by the hardware which identifies the source of the interrupt (one of 224 user defined interrupts).

Interrupts through interrupt gates automatically reset IF, disabling INTR requests. Interrupts through Trap Gates leave the state of the IF bit unchanged. Interrupts through a Task Gate change the IF bit according to the image of the EFLAGs register in the task's Task State Segment (TSS). When an IRET instruction is executed, the original state of the IF bit is restored.

Non-Maskable Interrupt

Non-maskable interrupts provide a method of servicing very high priority interrupts. When the NMI input is pulled HIGH it causes an interrupt with an internally supplied vector value of 2. Unlike a normal hardware interrupt, no interrupt acknowledgment sequence is performed for an NMI.

While executing the NMI servicing procedure, the 386SX Microprocessor will not service any further NMI request or INT requests until an interrupt return (IRET) instruction is executed or the processor is reset. If NMI occurs while currently servicing an NMI, its presence will be saved for servicing after executing the first IRET instruction. The IF bit is cleared at the beginning of an NMI interrupt to inhibit further INTR interrupts.

Software Interrupts

A third type of interrupt/exception for the 386SX Microprocessor is the software interrupt. An INT n instruction causes the processor to execute the interrupt service routine pointed to by the n^{th} vector in the interrupt table.

A special case of the two byte software interrupt INT n is the one byte INT 3, or breakpoint interrupt. By inserting this one byte instruction in a program, the user can set breakpoints in his program as a debugging tool.

A final type of software interrupt is the single step interrupt. It is discussed in **Single Step Trap** (page 20).

INTERRUPT AND EXCEPTION PRIORITIES

Interrupts are externally generated events. Maskable Interrupts (on the INTR input) and Non-Maskable Interrupts (on the NMI input) are recognized at instruction boundaries. When NMI and maskable INTR are **both** recognized at the **same** instruction boundary, the 386SX Microprocessor invokes the NMI service routine first. If maskable interrupts are still enabled after the NMI service routine has been invoked, then the 386SX Microprocessor will invoke the appropriate interrupt service routine.

As the 386SX Microprocessor executes instructions, it follows a consistent cycle in checking for exceptions, as shown in Table 2.6. This cycle is repeated as each instruction is executed, and occurs in parallel with instruction decoding and execution.

INSTRUCTION RESTART

The 386SX Microprocessor fully supports restarting all instructions after Faults. If an exception is detected in the instruction to be executed (exception categories 4 through 10 in Table 2.6), the 386SX Microprocessor invokes the appropriate exception service routine. The 386SX Microprocessor is in a state that permits restart of the instruction, for all cases but those given in Table 2.7. Note that all such cases will be avoided by a properly designed operating system.

Table 2.6. Sequence of Exception Checking

Consider the case of the 386SX™ Microprocessor having just completed an instruction. It then performs the following checks before reaching the point where the next instruction is completed:

1. Check for Exception 1 Traps from the instruction just completed (single-step via Trap Flag, or Data Breakpoints set in the Debug Registers).
2. Check for external NMI and INTR.
3. Check for Exception 1 Faults in the next instruction (Instruction Execution Breakpoint set in the Debug Registers for the next instruction).
4. Check for Segmentation Faults that prevented fetching the entire next instruction (exceptions 11 or 13).
5. Check for Page Faults that prevented fetching the entire next instruction (exception 14).
6. Check for Faults decoding the next instruction (exception 6 if illegal opcode; exception 6 if in Real Mode or in Virtual 8086 Mode and attempting to execute an instruction for Protected Mode only; or exception 13 if instruction is longer than 15 bytes, or privilege violation in Protected Mode (i.e. not at IOPL or at CPL = 0).
7. If WAIT opcode, check if TS = 1 and MP = 1 (exception 7 if both are 1).
8. If ESCape opcode for numeric coprocessor, check if EM = 1 or TS = 1 (exception 7 if either are 1).
9. If WAIT opcode or ESCape opcode for numeric coprocessor, check ERROR# input signal (exception 16 if ERROR# input is asserted).
10. Check in the following order for each memory reference required by the instruction:
    a. Check for Segmentation Faults that prevent transferring the entire memory quantity (exceptions 11, 12, 13).
    b. Check for Page Faults that prevent transferring the entire memory quantity (exception 14).

NOTE:
Segmentation exceptions are generated before paging exceptions.

Table 2.7. Conditions Preventing Instruction Restart

1. An instruction causes a task switch to a task whose Task State Segment is **partially** 'not present' (An entirely 'not present' TSS is restartable). Partially present TSS's can be avoided either by keeping the TSS's of such tasks present in memory, or by aligning TSS segments to reside entirely within a single 4K page (for TSS segments of 4K bytes or less).
2. A coprocessor operand wraps around the top of a 64K-byte segment or a 4G-byte segment, and spans three pages, and the page holding the middle portion of the operand is 'not present'. This condition can be avoided by starting **at a page boundary** any segments containing coprocessor operands if the segments are approximately 64K-200 bytes or larger (i.e. large enough for wraparound of the coprocessor operand to possibly occur).

Note that these conditions are avoided by using the operating system designs mentioned in this table.

Table 2.8. Register Values after Reset

Flag Word (EFLAGS)	uuuu0002H	Note 1
Machine Status Word (CR0)	uuuuuuu0H	Note 2
Instruction Pointer (EIP)	0000FFF0H	
Code Segment (CS)	F000H	Note 3
Data Segment (DS)	0000H	Note 4
Stack Segment (SS)	0000H	
Extra Segment (ES)	0000H	Note 4
Extra Segment (FS)	0000H	
Extra Segment (GS)	0000H	
EAX register	0000H	Note 5
EDX register	component and stepping ID	Note 6
All other registers	undefined	Note 7

NOTES:
1. EFLAG Register. The upper 14 bits of the EFLAGS register are undefined, all defined flag bits are zero.
2. CR0: All of the defined fields in CRO are 0.
3. The Code Segment Register (CS) will have its Base Address set to 0FFFF0000H and Limit set to 0FFFFH.
4. The Data and Extra Segment Registers (DS, ES) will have their Base Address set to 000000000H and Limit set to 0FFFFH.
5. If self-test is selected, the EAX register should contain a 0 value. If a value of 0 is not found then the self-test has detected a flaw in the part.
6. EDX register always holds component and stepping identifier.
7. All undefined bits are Intel Reserved and should not be used.

DOUBLE FAULT

A Double Fault (exception 8) results when the processor attempts to invoke an exception service routine for the segment exceptions (10, 11, 12 or 13), but in the process of doing so detects an exception **other than** a Page Fault (exception 14).

One other cause of generating a Double Fault is the 386SX Microprocessor detecting any other exception when it is attempting to invoke the Page Fault (exception 14) service routine (for example, if a Page Fault is detected when the 386SX Microprocessor attempts to invoke the Page Fault service routine). Of course, in any functional system, not only in 386SX Microprocessor-based systems, the entire page fault service routine must remain 'present' in memory.

2.8 Reset and Initialization

When the processor is initialized or Reset the registers have the values shown in Table 2.8. The 386SX Microprocessor will then start executing instructions near the top of physical memory, at location 0FFFFF0H. When the first Intersegment Jump or Call is executed, address lines A_{20}–A_{23} will drop LOW for CS-relative memory cycles, and the 386SX Microprocessor will only execute instructions in the lower one megabyte of physical memory. This allows the system designer to use a shadow ROM at the top of physical memory to initialize the system and take care of Resets.

RESET forces the 386SX Microprocessor to terminate all execution and local bus activity. No instruction execution or bus activity will occur as long as Reset is active. Between 350 and 450 CLK2 periods after Reset becomes inactive, the 386SX Microprocessor will start executing instructions at the top of physical memory.

2.9 Testability

The 386SX Microprocessor, like the 386 Microprocessor, offers testability features which include a self-test and direct access to the page translation cache.

SELF-TEST

The 386SX Microprocessor has the capability to perform a self-test. The self-test checks the function of all of the Control ROM and most of the non-random logic of the part. Approximately one-half of the 386SX Microprocessor can be tested during self-test.

Self-Test is initiated on the 386SX Microprocessor when the RESET pin transitions from HIGH to LOW, and the BUSY# pin is LOW. The self-test takes about 2^{20} clocks, or approximately 33 milliseconds with a 16 MHz 386SX CPU. At the completion of self-test the processor performs reset and begins normal operation. The part has successfully passed self-test if the contents of the EAX are zero. If the results of the EAX are not zero then the self-test has detected a flaw in the part.

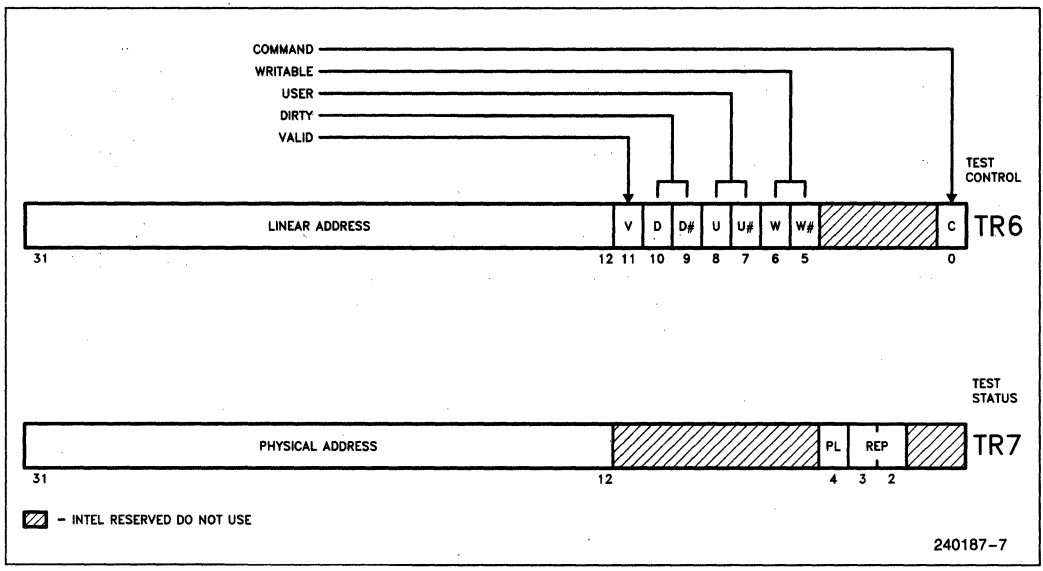


Figure 2.6. Test Registers

TLB TESTING

The 386SX Microprocessor also provides a mechanism for testing the Translation Lookaside Buffer (TLB) if desired. This particular mechanism may not be continued in the same way in future processors.

There are two TLB testing operations: 1) writing entries into the TLB, and, 2) performing TLB lookups. Two Test Registers, shown in Figure 2.6, are provided for the purpose of testing. TR6 is the "test command register", and TR7 is the "test data register". For a more detailed explanation of testing the TLB, see the 386SX™ Microprocessor Programmer's Reference Manual.

2.10 Debugging Support

The 386SX Microprocessor provides several features which simplify the debugging process. The three categories of on-chip debugging aids are:

1. The code execution breakpoint opcode (0CCH).
2. The single-step capability provided by the TF bit in the flag register.
3. The code and data breakpoint capability provided by the Debug Registers DR0–3, DR6, and DR7.

BREAKPOINT INSTRUCTION

A single-byte software interrupt (Int 3) breakpoint instruction is available for use by software debuggers. The breakpoint opcode is 0CCh, and generates an exception 3 trap when executed.

SINGLE-STEP TRAP

If the single-step flag (TF, bit 8) in the EFLAG register is found to be set at the end of an instruction, a single-step exception occurs. The single-step exception is auto vectored to exception number 1.

DEBUG REGISTERS

The Debug Registers are an advanced debugging feature of the 386SX Microprocessor. They allow data access breakpoints as well as code execution breakpoints. Since the breakpoints are indicated by on-chip registers, an instruction execution breakpoint can be placed in ROM code or in code shared by several tasks, neither of which can be supported by the INT 3 breakpoint opcode.

The 386SX Microprocessor contains six Debug Registers, consisting of four breakpoint address registers and two breakpoint control registers. Initially after reset, breakpoints are in the disabled state; therefore, no breakpoints will occur unless the debug registers are programmed. Breakpoints set up in the Debug Registers are auto-vectored to exception 1. Figure 2.7 shows the breakpoint status and control registers.

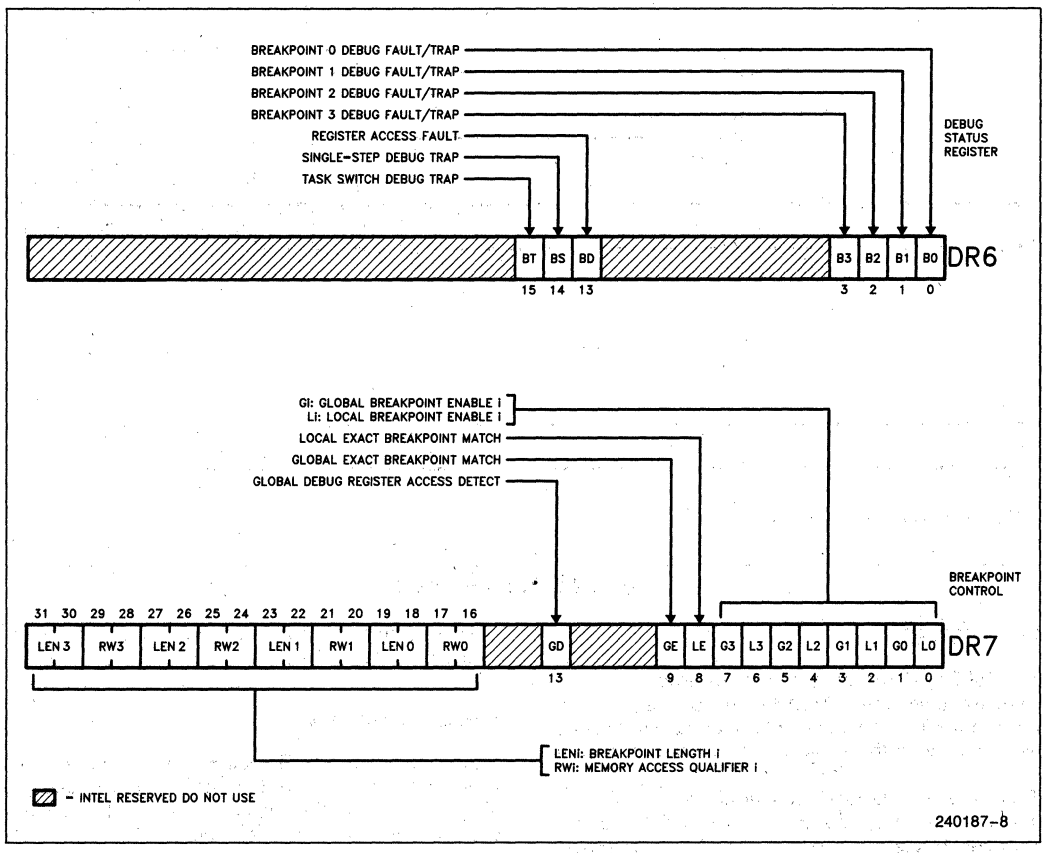


Figure 2.7. Debug Registers

3.0 REAL MODE ARCHITECTURE

When the processor is reset or powered up it is initialized in Real Mode. Real Mode has the same base architecture as the 8086, but allows access to the 32-bit register set of the 386SX Microprocessor. The addressing mechanism, memory size, and interrupt handling are all identical to the Real Mode on the 80286.

The default operand size in Real Mode is 16 bits, as in the 8086. In order to use the 32-bit registers and addressing modes, override prefixes must be used. In addition, the segment size on the 386SX Microprocessor in Real Mode is 64K bytes so 32-bit addresses must have a value less then 0000FFFFH. The primary purpose of Real Mode is to set up the processor for Protected Mode operation.

3.1 Memory Addressing

In Real Mode the linear addresses are the same as physical addresses (paging is not allowed). Physical addresses are formed in Real Mode by adding the contents of the appropriate segment register which is shifted left by four bits to an effective address. This addition results in a 20-bit physical address or a 1 megabyte address space. Since segment registers are shifted left by 4 bits, Real Mode segments always start on 16-byte boundaries.

All segments in Real Mode are exactly 64K bytes long, and may be read, written, or executed. The 386SX Microprocessor will generate an exception 13 if a data operand or instruction fetch occurs past the end of a segment.

Table 3.1. Exceptions in Real Mode

Function	Interrupt Number	Related Instructions	Return Address Location
Interrupt table limit too small	8	INT vector is not within table limit	Before Instruction
CS, DS, ES, FS, GS Segment overrun exception	13	Word memory reference with offset = 0FFFFH. an attempt to execute past the end of CS segment.	Before Instruction
SS Segment overrun exception	12	Stack Reference beyond offset = 0FFFFH	Before Instruction

3.2 Reserved Locations

There are two fixed areas in memory which are reserved in Real address mode: the system initialization area and the interrupt table area. Locations 00000H through 003FFH are reserved for interrupt vectors. Each one of the 256 possible interrupts has a 4-byte jump vector reserved for it. Locations 0FFFFF0H through 0FFFFFFH are reserved for system initialization.

3.3 Interrupts

Many of the exceptions discussed in section 2.7 are not applicable to Real Mode operation; in particular, exceptions 10, 11 and 14 do not occur in Real Mode. Other exceptions have slightly different meanings in Real Mode; Table 3.1 identifies these exceptions.

3.4 Shutdown and Halt

The HLT instruction stops program execution and prevents the processor from using the local bus until restarted. Either NMI, INTR with interrupts enabled (IF = 1), or RESET will force the 386SX Microprocessor out of halt. If interrupted, the saved CS:IP will point to the next instruction after the HLT.

Shutdown will occur when a severe error is detected that prevents further processing. In Real Mode, shutdown can occur under two conditions:

1. An interrupt or an exception occurs (Exceptions 8 or 13) and the interrupt vector is larger than the Interrupt Descriptor Table.
2. A CALL, INT or PUSH instruction attempts to wrap around the stack segment when SP is not even.

An NMI input can bring the processor out of shutdown if the Interrupt Descriptor Table limit is large enough to contain the NMI interrupt vector (at least 000FH) and the stack has enough room to contain the vector and flag information (i.e. SP is greater that 0005H). Otherwise, shutdown can only be exited by a processor reset.

3.5 LOCK operation

The LOCK prefix on the 386SX Microprocessor, even in Real Mode, is more restrictive than on the 80286. This is due to the addition of paging on the 386SX Microprocessor in Protected Mode and Virtual 8086 Mode. The LOCK prefix is not supported during repeat string instructions.

The only instruction forms where the LOCK prefix is legal on the 386SX Microprocessor are shown in Table 3.2.

Table 3.2. Legal Instructions for the LOCK Prefix

Opcode	Operands (Dest, Source)
BIT Test and SET/RESET /COMPLEMENT	Mem, Reg/Immediate
XCHG	Reg, Mem
XCHG	Mem, Reg
ADD, OR, ADC, SBB, AND, SUB, XOR	Mem, Reg/Immediate
NOT, NEG, INC, DEC	Mem

An exception 6 will be generated if a LOCK prefix is placed before any instruction form or opcode not listed above. The LOCK prefix allows indivisible read/modify/write operations on memory operands using the instructions above.

The LOCK prefix is not IOPL-sensitive on the 386SX Microprocessor. The LOCK prefix can be used at any privilege level, but only on the instruction forms listed in Table 3.2.

4.0 PROTECTED MODE ARCHITECTURE

The complete capabilities of the 386SX Microprocessor are unlocked when the processor operates in Protected Virtual Address Mode (Protected Mode). Protected Mode vastly increases the linear address space to four gigabytes (2^{32} bytes) and allows the running of virtual memory programs of almost unlimited size (64 terabytes (2^{46} bytes)). In addition, Protected Mode allows the 386SX Microprocessor to run all of the existing 386 CPU (using only 16 megabytes of physical memory), 80286 and 8086 CPU's software, while providing a sophisticated memory management and a hardware-assisted protection mechanism. Protected Mode allows the use of additional instructions specially optimized for supporting multitasking operating systems. The base architecture of the 386SX Microprocessor remains the same; the registers, instructions, and addressing modes described in the previous sections are retained. The main difference between Protected Mode and Real Mode from a programmer's viewpoint is the increased address space and a different addressing mechanism.

4.1 Addressing Mechanism

Like Real Mode, Protected Mode uses two components to form the logical address; a 16-bit selector is used to determine the linear base address of a segment, the base address is added to a 32-bit effective address to form a 32-bit linear address. The linear address is then either used as a 24-bit physical address, or if paging is enabled the paging mechanism maps the 32-bit linear address into a 24-bit physical address.

The difference between the two modes lies in calculating the base address. In Protected Mode, the selector is used to specify an index into an operating system defined table (see Figure 4.1). The table contains the 32-bit base address of a given segment. The physical address is formed by adding the base address obtained from the table to the offset.

Paging provides an additional memory management mechanism which operates only in Protected Mode. Paging provides a means of managing the very large segments of the 386SX Microprocessor, as such paging operates beneath segmentation. The page mechanism translates the protected linear address which comes from the segmentation unit into a physical address. Figure 4.2 shows the complete 386SX Microprocessor addressing mechanism with paging enabled.

4.2 Segmentation

Segmentation is one method of memory management. Segmentation provides the basis for protection. Segments are used to encapsulate regions of memory which have common attributes. For example, all of the code of a given program could be contained in a segment, or an operating system table may reside in a segment. All information about each segment is stored in an 8 byte data structure called a descriptor. All of the descriptors in a system are contained in descriptor tables which are recognized by hardware.

TERMINOLOGY

The following terms are used throughout the discussion of descriptors, privilege levels and protection:

PL: Privilege Level—One of the four hierarchical privilege levels. Level 0 is the most privileged level and level 3 is the least privileged.

RPL: Requestor Privilege Level—The privilege level of the original supplier of the selector. RPL is determined by the least two significant bits of a selector.

DPL: Descriptor Privilege Level—This is the least privileged level at which a task may access that descriptor (and the segment associated with that descriptor). Descriptor Privilege Level is determined by bits 6:5 in the Access Right Byte of a descriptor.

CPL: Current Privilege Level—The privilege level at which a task is currently executing, which equals the privilege level of the code segment being executed. CPL can also be determined by examining the lowest 2 bits of the CS register, except for conforming code segments.

EPL: Effective Privilege Level—The effective privilege level is the least privileged of the RPL and the DPL. EPL is the numerical maximum of RPL and DPL.

Task: One instance of the execution of a program. Tasks are also referred to as processes.

DESCRIPTOR TABLES

The descriptor tables define all of the segments which are used in a 386SX Microprocessor system. There are three types of tables which hold descriptors: the Global Descriptor Table, Local Descriptor Table, and the Interrupt Descriptor Table. All of the tables are variable length memory arrays and can vary in size from 8 bytes to 64K bytes. Each table can hold up to 8192 8-byte descriptors. The upper 13 bits of a selector are used as an index into the descriptor table. The tables have registers associated with them which hold the 32-bit linear base address and the 16-bit limit of each table.

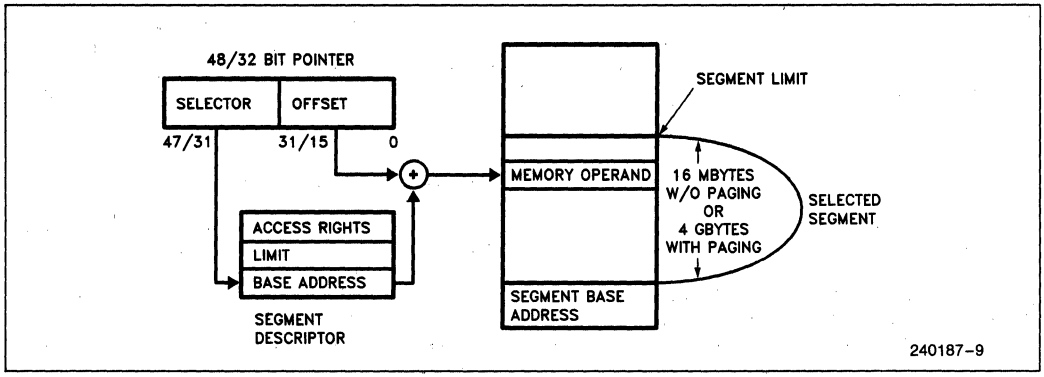


Figure 4.1. Protected Mode Addressing

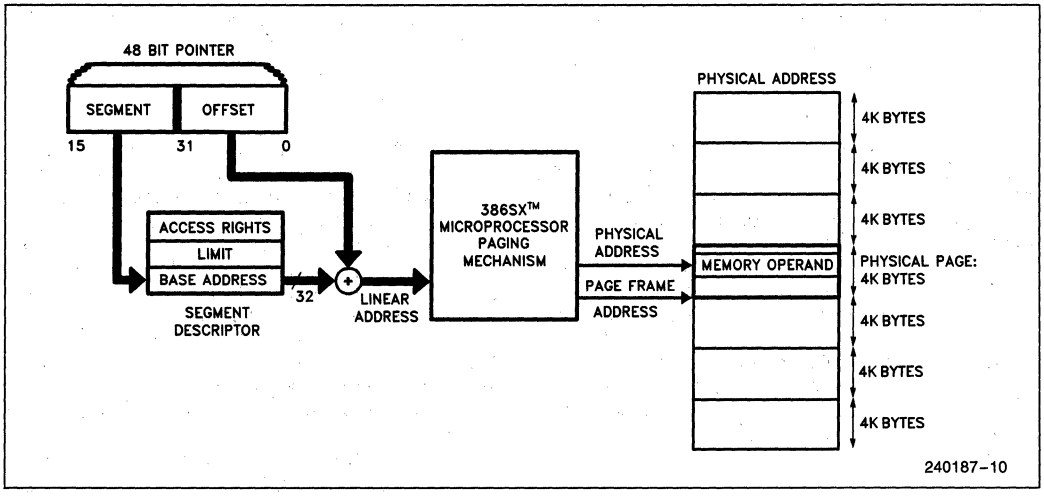


Figure 4.2. Paging and Segmentation

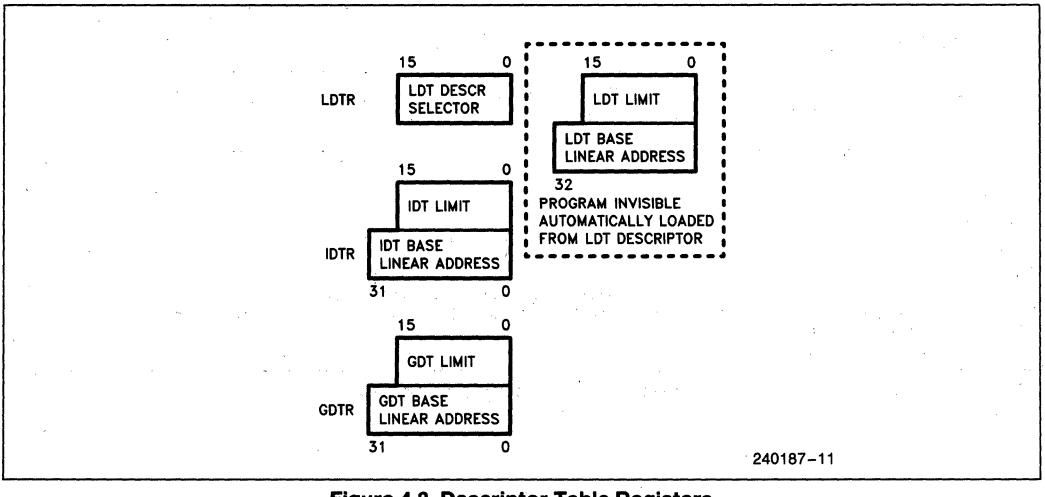


Figure 4.3. Descriptor Table Registers

Each of the tables has a register associated with it: GDTR, LDTR, and IDTR; see Figure 2.1. The LGDT, LLDT, and LIDT instructions load the base and limit of the Global, Local, and Interrupt Descriptor Tables into the appropriate register. The SGDT, SLDT, and SIDT store the base and limit values. These are privileged instructions.

Global Descriptor Table

The Global Descriptor Table (GDT) contains descriptors which are available to all of the tasks in a system. The GDT can contain any type of segment descriptor except for interrupt and trap descriptors. Every 386SX CPU system contains a GDT.

The first slot of the Global Descriptor Table corresponds to the null selector and is not used. The null selector defines a null pointer value.

Local Descriptor Table

LDTs contain descriptors which are associated with a given task. Generally, operating systems are designed so that each task has a separate LDT. The LDT may contain only code, data, stack, task gate, and call gate descriptors. LDTs provide a mechanism for isolating a given task's code and data segments from the rest of the operating system, while the GDT contains descriptors for segments which are common to all tasks. A segment cannot be accessed by a task if its segment descriptor does not exist in either the current LDT or the GDT. This provides both isolation and protection for a task's segments while still allowing global data to be shared among tasks.

Unlike the 6-byte GDT or IDT registers which contain a base address and limit, the visible portion of the LDT register contains only a 16-bit selector. This selector refers to a Local Descriptor Table descriptor in the GDT (see figure 2.1).

Interrupt Descriptor Table

The third table needed for 386SX Microprocessor systems is the Interrupt Descriptor Table. The IDT contains the descriptors which point to the location of the up to 256 interrupt service routines. The IDT may contain only task gates, interrupt gates, and trap gates. The IDT should be at least 256 bytes in size in order to hold the descriptors for the 32 Intel Reserved Interrupts. Every interrupt used by a system must have an entry in the IDT. The IDT entries are referenced by INT instructions, external interrupt vectors, and exceptions.

DESCRIPTORS

The object to which the segment selector points to is called a descriptor. Descriptors are eight byte quantities which contain attributes about a given region of linear address space. These attributes include the 32-bit base linear address of the segment, the 20-bit length and granularity of the segment, the protection level, read, write or execute privileges, the default size of the operands (16-bit or 32-bit), and the type of segment. All of the attribute information about a segment is contained in 12 bits in the segment descriptor. Figure 4.4 shows the general format of a descriptor. All segments on the 386SX Microprocessor have three attribute fields in common: the P bit, the DPL bit, and the S bit. The P

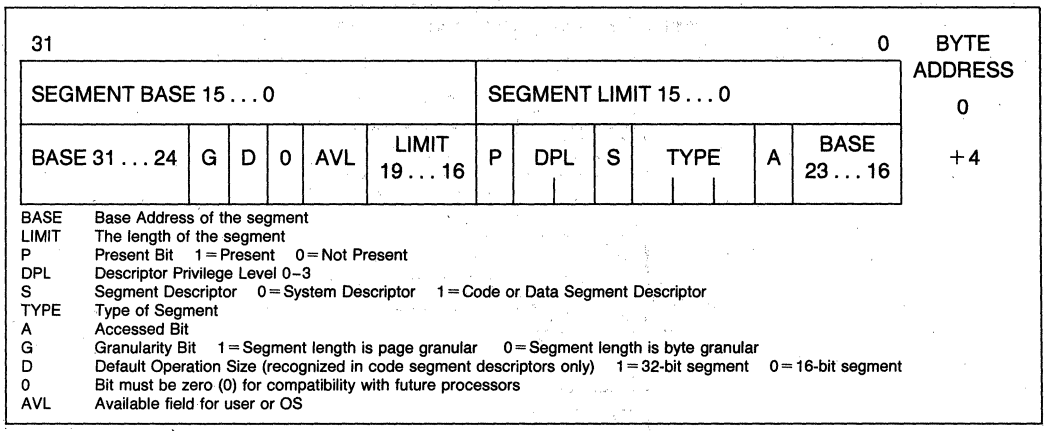


Figure 4.4. Segment Descriptors

(Present) Bit is 1 if the segment is loaded in physical memory. If P=0 then any attempt to access this segment causes a not present exception (exception 11). The Descriptor Privilege Level, DPL, is a two bit field which specifies the protection level, 0–3, associated with a segment.

The 386SX Microprocessor has two main categories of segments: system segments and non-system segments (for code and data). The segment bit, S, determines if a given segment is a system segment or a code or data segment. If the S bit is 1 then the segment is either a code or data segment; if it is 0 then the segment is a system segment.

Code and Data Descriptors (S = 1)

Figure 4.5 shows the general format of a code and data descriptor and Table 4.1 illustrates how the bits in the Access Right Byte are interpreted.

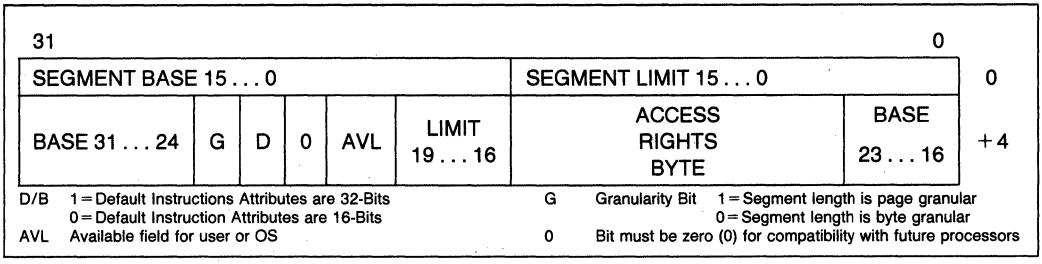
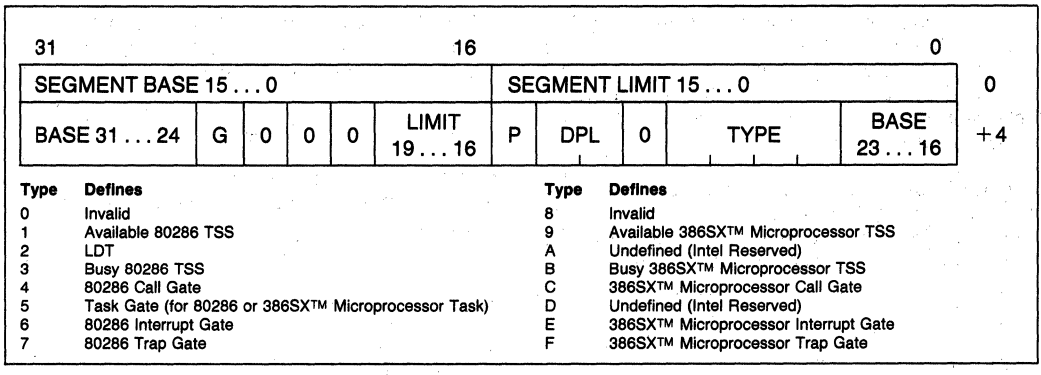


Figure 4.5. Code and Data Descriptors

Table 4.1. Access Rights Byte Definition for Code and Data Descriptors

Bit Position	Name	Function	
7	Present (P)	P = 1 Segment is mapped into physical memory. P = 0 No mapping to physical memory exists, base and limt are not used.	
6–5	Descriptor Privilege Level (DPL)	Segment privilege attribute used in privilege tests.	
4	Segment Descriptor (S)	S = 1 Code or Data (includes stacks) segment descriptor S = 0 System Segment Descriptor or Gate Descriptor	
3	Executable (E)	E = 0 Descriptor type is data segment:	If Data Segment (S = 1, E = 0)
2	Expansion Direction (ED)	ED = 0 Expand up segment, offsets must be ≤ limit. ED = 1 Expand down segment, offsets must be > limit.	
1	Writeable (W)	W = 0 Data segment may not be written into. W = 1 Data segment may be written into.	
3	Executable (E)	E = 1 Descriptor type is code segment:	If Code Segment (S = 1, E = 1)
2	Conforming (C)	C = 1 Code segment may only be executed when CPL ≥ DPL and CPL remains unchanged.	
1	Readable (R)	R = 0 Code segment may not be read. R = 1 Code segment may be read.	
0	Accessed (A)	A = 0 Segment has not been accessed. A = 1 Segment selector has been loaded into segment register or used by selector test instructions.	



Type	Defines
0	Invalid
1	Available 80286 TSS
2	LDT
3	Busy 80286 TSS
4	80286 Call Gate
5	Task Gate (for 80286 or 386SX™ Microprocessor Task)
6	80286 Interrupt Gate
7	80286 Trap Gate

Type	Defines
8	Invalid
9	Available 386SX™ Microprocessor TSS
A	Undefined (Intel Reserved)
B	Busy 386SX™ Microprocessor TSS
C	386SX™ Microprocessor Call Gate
D	Undefined (Intel Reserved)
E	386SX™ Microprocessor Interrupt Gate
F	386SX™ Microprocessor Trap Gate

Figure 4.6. System Descriptors

Code and data segments have several descriptor fields in common. The accessed bit, A, is set whenever the processor accesses a descriptor. The granularity bit, G, specifies if a segment length is byte-granular or page-granular.

System Descriptor Formats (S = 0)

System segments describe information about operating system tables, tasks, and gates. Figure 4.6 shows the general format of system segment descriptors, and the various types of system segments. 386SX system descriptors (which are the same as 386 CPU system descriptors) contain a 32-bit base linear address and a 20-bit segment limit. 80286 system descriptors have a 24-bit base address and a 16-bit segment limit. 80286 system descriptors are identified by the upper 16 bits being all zero.

Differences Between 386SX™ Microprocessor and 80286 Descriptors

In order to provide operating system compatibility with the 80286 the 386SX CPU supports all of the 80286 segment descriptors. The 80286 system segment descriptors contain a 24-bit base address and 16-bit limit, while the 386SX CPU system segment descriptors have a 32-bit base address, a 20-bit limit field, and a granularity bit. The word count field specifies the number of 16-bit quantities to copy for 80286 call gates and 32-bit quantities for 386SX CPU call gates.

Selector Fields

A selector in Protected Mode has three fields: Local or Global Descriptor Table indicator (TI), Descriptor Entry Index (Index), and Requestor (the selector's) Privilege Level (RPL) as shown in Figure 4.7. The TI bit selects either the Global Descriptor Table or the Local Descriptor Table. The Index selects one of 8k descriptors in the appropriate descriptor table. The RPL bits allow high speed testing of the selector's privilege attributes.

Segment Descriptor Cache

In addition to the selector value, every segment register has a segment descriptor cache register associated with it. Whenever a segment register's contents are changed, the 8-byte descriptor associated with that selector is automatically loaded (cached) on the chip. Once loaded, all references to that segment use the cached descriptor information instead of reaccessing the descriptor. The contents of the descriptor cache are not visible to the programmer. Since descriptor caches only change when a segment register is changed, programs which modify the descriptor tables must reload the appropriate segment registers after changing a descriptor's value.

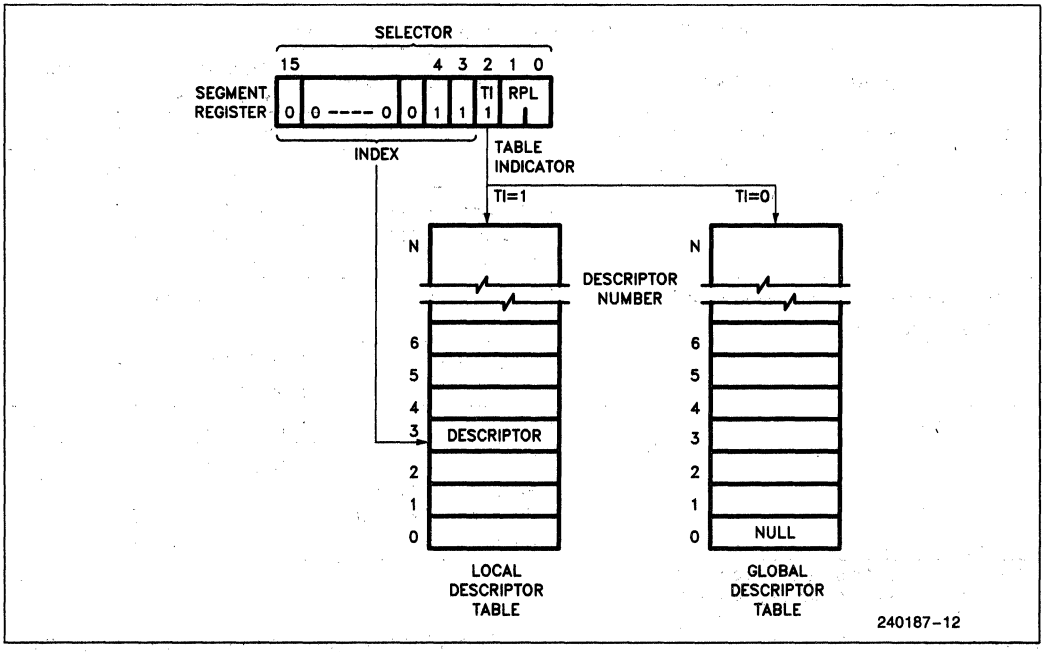


Figure 4.7. Example Descriptor Selection

4.3 Protection

The 386SX Microprocessor has four levels of protection which are optimized to support a multi-tasking operating system and to isolate and protect user programs from each other and the operating system. The privilege levels control the use of privileged instructions, I/O instructions, and access to segments and segment descriptors. The 386SX Microprocessor also offers an additional type of protection on a page basis when paging is enabled.

The four-level hierarchical privilege system is an extension of the user/supervisor privilege mode commonly used by minicomputers. The user/supervisor mode is fully supported by the 386SX Microprocessor paging mechanism. The privilege levels (PL) are numbered 0 through 3. Level 0 is the most privileged level.

RULES OF PRIVILEGE

The 386SX Microprocessor controls access to both data and procedures between levels of a task, according to the following rules.

— Data stored in a segment with privilege level **p** can be accessed only by code executing at a privilege level at least as privileged as **p**.

— A code segment/procedure with privilege level **p** can only be called by a task executing at the same or a lesser privilege level than **p**.

PRIVILEGE LEVELS

At any point in time, a task on the 386SX Microprocessor always executes at one of the four privilege levels. The Current Privilege Level (CPL) specifies what the task's privilege level is. A task's CPL may only be changed by control transfers through gate descriptors to a code segment with a different privilege level. Thus, an application program running at PL=3 may call an operating system routine at PL=1 (via a gate) which would cause the task's CPL to be set to 1 until the operating system routine was finished.

Selector Privilege (RPL)

The privilege level of a selector is specified by the RPL field. The selector's RPL is only used to establish a less trusted privilege level than the current privilege level of the task for the use of a segment. This level is called the task's effective privilege level (EPL). The EPL is defined as being the least privileged (numerically larger) level of a task's CPL and a selector's RPL. The RPL is most commonly used to verify that pointers passed to an operating system procedure do not access data that is of higher privilege than the procedure that originated the pointer. Since the originator of a selector can specify any RPL value, the Adjust RPL (ARPL) instruction is provided to force the RPL bits to the originator's CPL.

Table 4.2. Descriptor Types Used for Control Transfer

Control Transfer Types	Operation Types	Descriptor Referenced	Descriptor Table
Intersegment within the same privilege level	JMP, CALL RET, IRET*	Code Segment	GDT/LDT
Intersegment to the same or higher privilege level Interrupt within task may change CPL	CALL	Call Gate	GDT/LDT
	Interrupt instruction Exception External Interrupt	Trap or Interrupt Gate	IDT
Intersegment to a lower privilege level (changes task CPL)	RET, IRET*	Code Segment	GDT/LDT
	CALL, JMP	Task State Segment	GDT
Task Switch	CALL, JMP	Task Gate	GDT/LDT
	IRET** Interrupt instruction, Exception, External Interrupt	Task Gate	IDT

*NT (Nested Task bit of flag register) = 0
**NT (Nested Task bit of flag register) = 1

I/O Privilege

The I/O privilege level (IOPL) lets the operating system code executing at CPL = 0 define the least privileged level at which I/O instructions can be used. An exception 13 (General Protection Violation) is generated if an I/O instruction is attempted when the CPL of the task is less privileged then the IOPL. The IOPL is stored in bits 13 and 14 of the EFLAGS register. The following instructions cause an exception 13 if the CPL is greater than IOPL: IN, INS, OUT, OUTS, STI, CLI, LOCK prefix.

Descriptor Access

There are basically two types of segment accesses: those involving code segments such as control transfers, and those involving data accesses. Determining the ability of a task to access a segment involves the type of segment to be accessed, the instruction used, the type of descriptor used and CPL, RPL, and DPL as described above.

Any time an instruction loads a data segment register (DS, ES, FS, GS) the 386SX Microprocessor makes protection validation checks. Selectors loaded in the DS, ES, FS, GS registers must refer only to data segment or readable code segments.

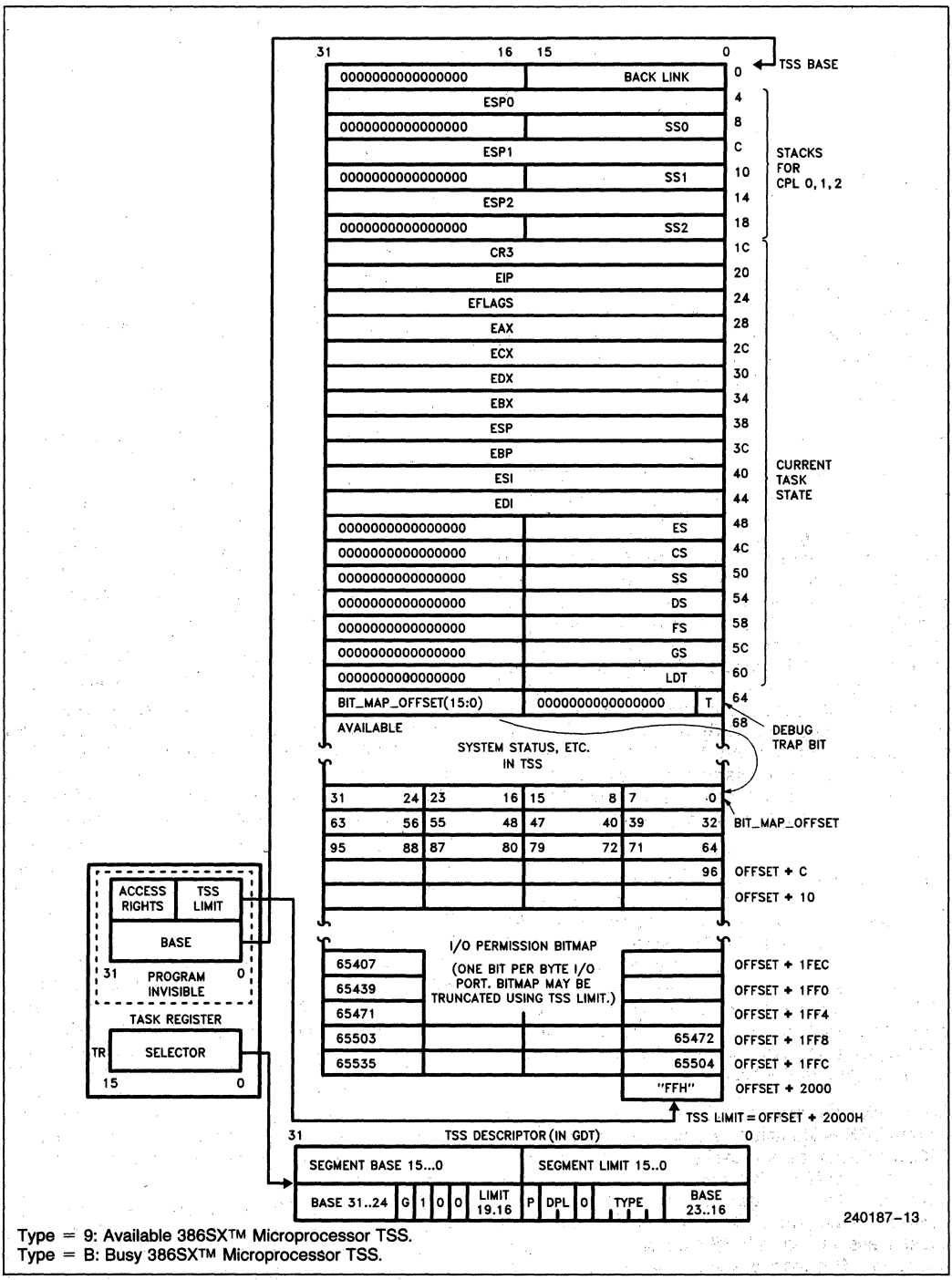
Finally the privilege validation checks are performed. The CPL is compared to the EPL and if the EPL is more privileged than the CPL, an exception 13 (general protection fault) is generated.

The rules regarding the stack segment are slightly different than those involving data segments. Instructions that load selectors into SS must refer to data segment descriptors for writeable data segments. The DPL and RPL must equal the CPL of all other descriptor types or a privilege level violation will cause an exception 13. A stack not present fault causes an exception 12.

PRIVILEGE LEVEL TRANSFERS

Inter-segment control transfers occur when a selector is loaded in the CS register. For a typical system most of these transfers are simply the result of a call or a jump to another routine. There are five types of control transfers which are summarized in Table 4.2. Many of these transfers result in a privilege level transfer. Changing privilege levels is done only by control transfers, using gates, task switches, and interrupt or trap gates.

Control transfers can only occur if the operation which loaded the selector references the correct descriptor type. Any violation of these descriptor usage rules will cause an exception 13.



Type = 9: Available 386SX™ Microprocessor TSS.
Type = B: Busy 386SX™ Microprocessor TSS.

Figure 4.8. 386SX™ Microprocessor TSS and TSS Registers

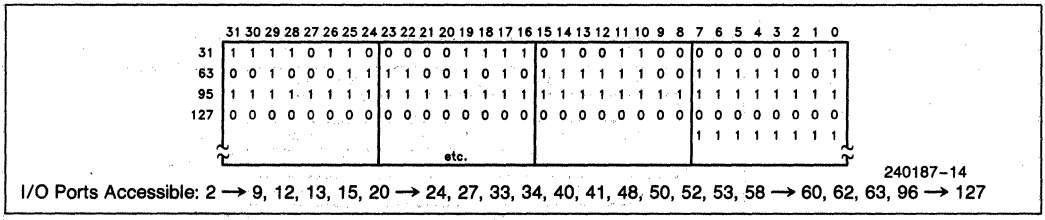


Figure 4.9. Sample I/O Permission Bit Map

CALL GATES

Gates provide protected indirect CALLs. One of the major uses of gates is to provide a secure method of privilege transfers within a task. Since the operating system defines all of the gates in a system, it can ensure that all gates only allow entry into a few trusted procedures.

TASK SWITCHING

A very important attribute of any multi-tasking/multi-user operating system is its ability to rapidly switch between tasks or processes. The 386SX Microprocessor directly supports this operation by providing a task switch instruction in hardware. The task switch operation saves the entire state of the machine (all of the registers, address space, and a link to the previous task), loads a new execution state, performs protection checks, and commences execution in the new task. Like transfer of control by gates, the task switch operation is invoked by executing an inter-segment JMP or CALL instruction which refers to a Task State Segment (TSS), or a task gate descriptor in the GDT or LDT. An INT n instruction, exception, trap, or external interrupt may also invoke the task switch operation if there is a task gate descriptor in the associated IDT descriptor slot.

The TSS descriptor points to a segment (see Figure 4.8) containing the entire execution state. A task gate descriptor contains a TSS selector. The 386SX Microprocessor supports both 286 and 386SX CPU TSSs. The limit of a 386SX Microprocessor TSS must be greater than 64H (2BH for a 286 TSS), and can be as large as 16 megabytes. In the additional TSS space, the operating system is free to store additional information such as the reason the task is inactive, time the task has spent running, or open files belonging to the task.

Each task must have a TSS associated with it. The current TSS is identified by a special register in the 386SX Microprocessor called the Task State Segment Register (TR). This register contains a selector referring to the task state segment descriptor that defines the current TSS. A hidden base and limit register associated with TSS descriptor are loaded whenever TR is loaded with a new selector. Returning from a task is accomplished by the IRET instruction. When IRET is executed, control is returned to the task which was interrupted. The currently executing task's state is saved in the TSS and the old task state is restored from its TSS.

Several bits in the flag register and machine status word (CR0) give information about the state of a task which is useful to the operating system. The Nested Task bit, NT, controls the function of the IRET instruction. If NT = 0 the IRET instruction performs the regular return. If NT = 1 IRET performs a task switch operation back to the previous task. The NT bit is set or reset in the following fashion:

> When a CALL or INT instruction initiates a task switch, the new TSS will be marked busy and the back link field of the new TSS set to the old TSS selector. The NT bit of the new task is set by CALL or INT initiated task switches. An interrupt that does not cause a task switch will clear NT (The NT bit will be restored after execution of the interrupt handler). NT may also be set or cleared by POPF or IRET instructions.

The 386SX Microprocessor task state segment is marked busy by changing the descriptor type field from TYPE 9 to TYPE 0BH. A 286 TSS is marked busy by changing the descriptor type field from TYPE 1 to TYPE 3. Use of a selector that references a busy task state segment causes an exception 13.

The VM (Virtual Mode) bit is used to indicate if a task is a virtual 8086 task. If VM = 1 then the tasks will use the Real Mode addressing mechanism. The virtual 8086 environment is only entered and exited by a task switch.

The coprocessor's state is not automatically saved when a task switch occurs. The Task Switched Bit, TS, in the CR0 register helps deal with the coprocessor's state in a multi-tasking environment. Whenever the 386SX Microprocessor switches task, it sets the TS bit. The 386SX Microprocessor detects the first use of a processor extension instruction after a task switch and causes the processor extension not available exception 7. The exception handler for exception 7 may then decide whether to save the state of the coprocessor.

The T bit in the 386SX Microprocessor TSS indicates that the processor should generate a debug exception when switching to a task. If T = 1 then upon entry to a new task a debug exception 1 will be generated.

INITIALIZATION AND TRANSITION TO PROTECTED MODE

Since the 386SX Microprocessor begins executing in Real Mode immediately after RESET it is necessary to initialize the system tables and registers with the appropriate values. The GDT and IDT registers must refer to a valid GDT and IDT. The IDT should be at least 256 bytes long, and the GDT must contain descriptors for the initial code and data segments.

Protected Mode is enabled by loading CR0 with PE bit set. This can be accomplished by using the **MOV CR0, R/M** instruction. After enabling Protected Mode, the next instruction should execute an intersegment JMP to load the CS register and flush the instruction decode queue. The final step is to load all of the data segment registers with the initial selector values.

An alternate approach to entering Protected Mode is to use the built in task-switch to load all of the registers. In this case the GDT would contain two TSS descriptors in addition to the code and data descriptors needed for the first task. The first JMP instruction in Protected Mode would jump to the TSS causing a task switch and loading all of the registers with the values stored in the TSS. The Task State Segment Register should be initialized to point to a valid TSS descriptor.

4.4 Paging

Paging is another type of memory management useful for virtual memory multi-tasking operating systems. Unlike segmentation, which modularizes programs and data into variable length segments, paging divides programs into multiple uniform size pages. Pages bear no direct relation to the logical structure of a program. While segment selectors can be considered the logical 'name' of a program module or data structure, a page most likely corresponds to only a portion of a module or data structure.

PAGE ORGANIZATION

The 386SX Microprocessor uses two levels of tables to translate the linear address (from the segmentation unit) into a physical address. There are three components to the paging mechanism of the 386SX Microprocessor: the page directory, the page tables, and the page itself (page frame). All memory-resident elements of the 386SX Microprocessor paging mechanism are the same size, namely 4K bytes. A uniform size for all of the elements simplifies memory allocation and reallocation schemes, since there is no problem with memory fragmentation. Figure 4.10 shows how the paging mechanism works.

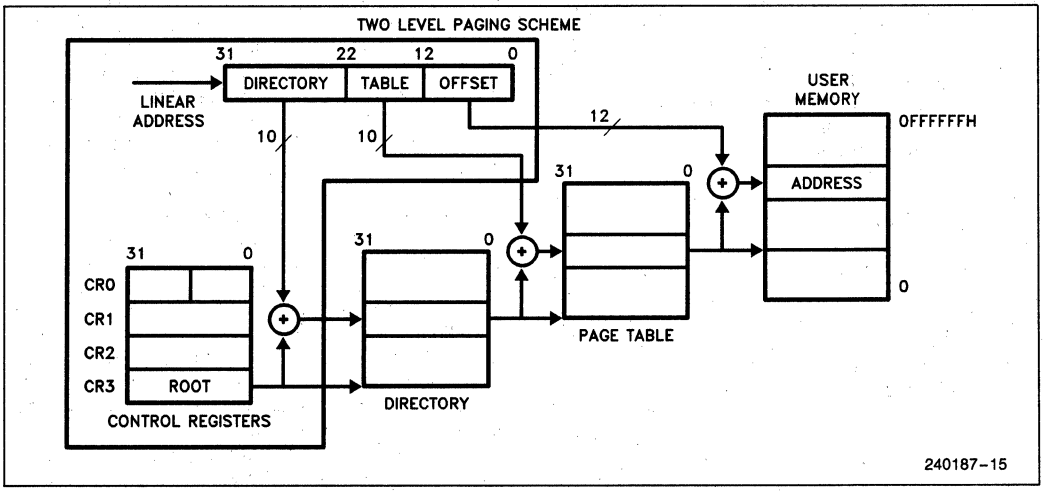


Figure 4.10. Paging Mechanism

31 12	11 10	9	8	7	6	5	4	3	2	1	0
PAGE TABLE ADDRESS 31..12	System Software Defineable		0	0	D	A	0	0	U/S	R/W	P

Figure 4.11. Page Directory Entry (Points to Page Table)

31 — 12	11 — 9	8	7	6	5	4	3	2	1	0
PAGE FRAME ADDRESS 31..12	System Software Defineable	0	0	D	A	0	0	U/S	R/W	P

Figure 4.12. Page Table Entry (Points to Page)

Page Fault Register

CR2 is the Page Fault Linear Address register. It holds the 32-bit linear address which caused the last Page Fault detected.

Page Descriptor Base Register

CR3 is the Page Directory Physical Base Address Register. It contains the physical starting address of the Page Directory (this value is truncated to a 24-bit value associated with the 386SX CPU's 16 megabyte physical memory limitation). The lower 12 bits of CR3 are always zero to ensure that the Page Directory is always page aligned. Loading it with a **MOV CR3, reg** instruction causes the page table entry cache to be flushed, as will a task switch through a TSS which changes the value of CR0.

Page Directory

The Page Directory is 4k bytes long and allows up to 1024 page directory entries. Each page directory entry contains information about the page table and the address of the next level of tables, the Page Tables. The contents of a Page Directory Entry are shown in figure 4.11. The upper 10 bits of the linear address (A_{31}–A_{22}) are used as an index to select the correct Page Directory Entry.

The page table address contains the upper 20 bits of a 32-bit physical address that is used as the base address for the next set of tables, the page tables. The lower 12 bits of the page table address are zero so that the page table addresses appear on 4 kbyte boundaries. For a 386 CPU system the upper 20 bits will select one of 2^{20} page tables, but for an 386SX Microprocessor system the upper 20 bits only select one of 2^{12} page tables. Again, this is because the 386SX Microprocessor is limited to a 24-bit physical address and the upper 8 bits (A_{24}–A_{31}) are truncated when the address is output on its 24 address pins.

Page Tables

Each Page Table is 4K bytes long and allows up to 1024 Page table Entries. Each page table entry contains information about the Page Frame and its address. The contents of a Page Table Entry are shown in figure 4.12. The middle 10 bits of the linear address (A_{21}–A_{12}) are used as an index to select the correct Page Table Entry.

The Page Frame Address contains the upper 20 bits of a 32-bit physical address that is used as the base address for the Page Frame. The lower 12 bits of the Page Frame Address are zero so that the Page Frame addresses appear on 4 kbyte boundaries. For an 386 CPU system the upper 20 bits will select one of 2^{20} Page Frames, but for an 386SX Microprocessor system the upper 20 bits only select one of 2^{12} Page Frames. Again, this is because the 386SX Microprocessor is limited to a 24-bit physical address space and the upper 8 bits (A_{24}–A_{31}) are truncated when the address is output on its 24 address pins.

Page Directory/Table Entries

The lower 12 bits of the Page Table Entries and Page Directory Entries contain statistical information about pages and page tables respectively. The P (Present) bit indicates if a Page Directory or Page Table entry can be used in address translation. If P = 1, the entry can be used for address translation. If P = 0, the entry cannot be used for translation. All of the other bits are available for use by the software. For example, the remaining 31 bits could be used to indicate where on disk the page is stored.

The A (Accessed) bit is set by the 386SX CPU for both types of entries before a read or write access occurs to an address covered by the entry. The D (Dirty) bit is set to 1 before a write to an address covered by that page table entry occurs. The D bit is undefined for Page Directory Entries. When the P, A and D bits are updated by the 386SX CPU, the processor generates a Read- Modify-Write cycle which locks the bus and prevents conflicts with other processors or peripherals. Software which modifies these bits should use the LOCK prefix to ensure the integrity of the page tables in multi-master systems.

The 3 bits marked system software definable in Figures 4.11 and Figure 4.12 are software definable. System software writers are free to use these bits for whatever purpose they wish.

PAGE LEVEL PROTECTION (R/W, U/S BITS)

The 386SX Microprocessor provides a set of protection attributes for paging systems. The paging mechanism distinguishes between two levels of protection: User, which corresponds to level 3 of the segmentation based protection, and supervisor which encompasses all of the other protection levels (0, 1, 2). Programs executing at Level 0, 1 or 2 bypass the page protection, although segmentation-based protection is still enforced by the hardware.

The U/S and R/W bits are used to provide User/Supervisor and Read/Write protection for individual pages or for all pages covered by a Page Table Directory Entry. The U/S and R/W bits in the second level Page Table Entry apply only to the page described by that entry. While the U/S and R/W bits in the first level Page Directory Table apply to all pages described by the page table pointed to by that directory entry. The U/S and R/W bits for a given page are obtained by taking the most restrictive of the U/S and R/W from the Page Directory Table Entries and using these bits to address the page.

TRANSLATION LOOKASIDE BUFFER

The 386SX Microprocessor paging hardware is designed to support demand paged virtual memory systems. However, performance would degrade substantially if the processor was required to access two levels of tables for every memory reference. To solve this problem, the 386SX Microprocessor keeps a cache of the most recently accessed pages, this cache is called the Translation Lookaside Buffer (TLB). The TLB is a four-way set associative 32-entry page table cache. It automatically keeps the most commonly used page table entries in the processor. The 32-entry TLB coupled with a 4K page size results in coverage of 128K bytes of memory addresses. For many common multi-tasking systems, the TLB will have a hit rate of greater than 98%. This means that the processor will only have to access the two-level page structure for less than 2% of all memory references.

PAGING OPERATION

The paging hardware operates in the following fashion. The paging unit hardware receives a 32-bit linear address from the segmentation unit. The upper 20 linear address bits are compared with all 32 entries in the TLB to determine if there is a match. If there is a match (i.e. a TLB hit), then the 24-bit physical address is calculated and is placed on the address bus.

If the page table entry is not in the TLB, the 386SX Microprocessor will read the appropriate Page Directory Entry. If P = 1 on the Page Directory Entry, indicating that the page table is in memory, then the 386SX Microprocessor will read the appropriate Page Table Entry and set the Access bit. If P = 1 on the Page Table Entry, indicating that the page is in memory, the 386SX Microprocessor will update the Access and Dirty bits as needed and fetch the operand. The upper 20 bits of the linear address, read from the page table, will be stored in the TLB for future accesses. If P = 0 for either the Page Directory Entry or the Page Table Entry, then the processor will generate a page fault Exception 14.

The processor will also generate a Page Fault (Exception 14) if the memory reference violated the page protection attributes. CR2 will hold the linear address which caused the page fault. Since Exception 14 is classified as a fault, CS:EIP will point to the instruction causing the page-fault. The 16-bit error code pushed as part of the page fault handler will contain status bits which indicate the cause of the page fault.

The 16-bit error code is used by the operating system to determine how to handle the Page Fault. Figure 4.13 shows the format of the Page Fault error code and the interpretation of the bits. Even though the bits in the error code (U/S, W/R, and P) have similar names as the bits in the Page Directory/Table Entries, the interpretation of the error code bits is different. Figure 4.14 indicates what type of access caused the page fault.

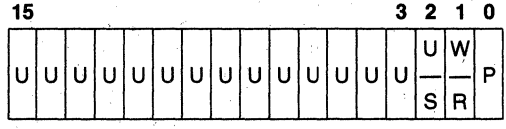


Figure 4.13. Page Fault Error Code Format

U/S: The U/S bit indicates whether the access causing the fault occurred when the processor was executing in User Mode (U/S = 1) or in Supervisor mode (U/S = 0)

W/R: The W/R bit indicates whether the access causing the fault was a Read (W/R = 0) or a Write (W/R = 1)

P: The P bit indicates whether a page fault was caused by a not-present page (P = 0), or by a page level protection violation (P = 1)

U = Undefined

U/S	W/R	Access Type
0	0	Supervisor* Read
0	1	Supervisor Write
1	0	User Read
1	1	User Write

*Descriptor table access will fault with U/S = 0, even if the program is executing at level 3.

Figure 4.14. Type of Access Causing Page Fault

OPERATING SYSTEM RESPONSIBILITIES

When the operating system enters or exits paging mode (by setting or resetting bit 31 in the CR0 register) a short JMP must be executed to flush the 386SX Microprocessor's prefetch queue. This ensures that all instructions executed after the address mode change will generate correct addresses.

The 386SX Microprocessor takes care of the page address translation process, relieving the burden from an operating system in a demand-paged system. The operating system is responsible for setting up the initial page tables and handling any page faults. The operating system also is required to invalidate (i.e. flush) the TLB when any changes are made to any of the page table entries. The operating system must reload CR3 to cause the TLB to be flushed.

Setting up the tables is simply a matter of loading CR3 with the address of the Page Directory, and allocating space for the Page Directory and the Page Tables. The primary responsibility of the operating system is to implement a swapping policy and handle all of the page faults.

A final concern of the operating system is to ensure that the TLB cache matches the information in the paging tables. In particular, any time the operating systems sets the P (Present) bit of page table entry to zero. The TLB must be flushed by reloading CR3. Operating systems may want to take advantage of the fact that CR3 is stored as part of a TSS, to give every task or group of tasks its own set of page tables.

4.5 Virtual 8086 Environment

The 386SX Microprocessor allows the execution of 8086 application programs in both Real Mode and in the Virtual 8086 Mode. The Virtual 8086 Mode allows the execution of 8086 applications, while still allowing the system designer to take full advantage of the 386SX CPU's protection mechanism.

VIRTUAL 8086 ADDRESSING MECHANISM

One of the major differences between 386SX CPU Real and Protected modes is how the segment selectors are interpreted. When the processor is executing in Virtual 8086 Mode, the segment registers are used in a fashion identical to Real Mode. The contents of the segment register are shifted left 4 bits and added to the offset to form the segment base linear address.

The 386SX Microprocessor allows the operating system to specify which programs use the 8086 address mechanism and which programs use Protected Mode addressing on a per task basis. Through the use of paging, the one megabyte address space of the Virtual Mode task can be mapped to anywhere in the 4 gigabyte linear address space of the 386SX Microprocessor. Like Real Mode, Virtual Mode addresses that exceed one megabyte will cause an exception 13. However, these restrictions should not prove to be important, because most tasks running in Virtual 8086 Mode will simply be existing 8086 application programs.

PAGING IN VIRTUAL MODE

The paging hardware allows the concurrent running of multiple Virtual Mode tasks, and provides protection and operating system isolation. Although it is not strictly necessary to have the paging hardware enabled to run Virtual Mode tasks, it is needed in order to run multiple Virtual Mode tasks or to relocate the address space of a Virtual Mode task to physical address space greater than one megabyte.

The paging hardware allows the 20-bit linear address produced by a Virtual Mode program to be divided into as many as 256 pages. Each one of the pages can be located anywhere within the maximum 16 megabyte physical address space of the 386SX Microprocessor. In addition, since CR3 (the Page Directory Base Register) is loaded by a task switch, each Virtual Mode task can use a different mapping scheme to map pages to different physical locations. Finally, the paging hardware allows the sharing of the 8086 operating system code between multiple 8086 applications.

PROTECTION AND I/O PERMISSION BIT MAP

All Virtual Mode programs execute at privilege level 3. As such, Virtual Mode programs are subject to all of the protection checks defined in Protected Mode. This is different than Real Mode, which implicitly is executing at privilege level 0. Thus, an attempt to execute a privileged instruction in Virtual Mode will cause an exception 13 fault.

The following are privileged instructions, which may be executed only at Privilege Level 0. Attempting to execute these instructions in Virtual 8086 Mode (or anytime CPL≥0) causes an exception 13 fault:

```
LIDT;   MOV DRn,REG;   MOV reg,DRn;
LGDT;   MOV TRn,reg;   MOV reg,TRn;
LMSW;   MOV CRn,reg;   MOV reg,CRn;

CLTS;
HLT;
```

Several instructions, particularly those applying to the multitasking and the protection model, are available only in Protected Mode. Therefore, attempting to execute the following instructions in Real Mode or in Virtual 8086 Mode generates an exception 6 fault:

```
LTR;    STR;
LLDT;   SLDT;
LAR;    VERR;
LSL;    VERW;
ARPL;
```

The instructions which are IOPL sensitive in Protected Mode are:

```
IN;     STI;
OUT;    CLI
INS;
OUTS;
REP INS;
REP OUTS;
```

In Virtual 8086 Mode the following instructions are IOPL-sensitive:

```
INT n;  STI;
PUSHF;  CLI;
POPF;   IRET;
```

The PUSHF, POPF, and IRET instructions are IOPL-sensitive in Virtual 8086 Mode only. This provision allows the IF flag to be virtualized to the virtual 8086 Mode program. The INT n software interrupt instruction is also IOPL-sensitive in Virtual 8086 mode. Note that the INT 3, INTO, and BOUND instructions are not IOPL-sensitive in Virtual 8086 Mode.

The I/O instructions that directly refer to addresses in the processor's I/O space are IN, INS, OUT, and OUTS. The 386SX Microprocessor has the ability to selectively trap references to specific I/O addresses. The structure that enables selective trapping is the *I/O Permission Bit Map* in the TSS segment (see Figures 4.8 and 4.9). The I/O permission map is a bit vector. The size of the map and its location in the TSS segment are variable. The processor locates the I/O permission map by means of the **I/O map base** field in the fixed portion of the TSS. The **I/O map base** field is 16 bits wide and contains the offset of the beginning of the I/O permission map.

In protected mode when an I/O instruction (IN, INS, OUT or OUTS) is encountered, the processor first checks whether CPL ≤ IOPL. If this condition is true, the I/O operation may proceed. If not true, the processor checks the I/O permission map (in Virtual 8086 Mode, the processor consults the map without regard for the IOPL).

Each bit in the map corresponds to an I/O port byte address; for example, the bit for port 41 is found at **I/O map base** + 5, bit offset 1. The processor tests all the bits that correspond to the I/O addresses spanned by an I/O operation; for example, a double word operation tests four bits corresponding to four adjacent byte addresses. If any tested bit is set, the processor signals a general protection exception. If all the tested bits are zero, the I/O operations may proceed.

It is not necessary for the I/O permission map to represent all the I/O addresses. I/O addresses not spanned by the map are treated as if they had one-bits in the map. The **I/O map base** should be at least one byte less than the TSS limit, the last byte beyond the I/O mapping information must contain all 1's.

Because the I/O permission map is in the TSS segment, different tasks can have different maps. Thus, the operating system can allocate ports to a task by changing the I/O permission map in the task's TSS.

IMPORTANT IMPLEMENTATION NOTE: Beyond the last byte of I/O mapping information in the I/O permission bit map **must** be a byte containing all 1's. The byte of all 1's must be within the limit of the 386SX CPU TSS segment (see Figure 4.8).

Interrupt Handling

In order to fully support the emulation of an 8086 machine, interrupts in Virtual 8086 Mode are handled in a unique fashion. When running in Virtual Mode all interrupts and exceptions involve a privilege change back to the host 386SX Microprocessor operating system. The 386SX Microprocessor operating system determines if the interrupt comes from a Protected Mode application or from a Virtual Mode program by examining the VM bit in the EFLAGS image stored on the stack.

When a Virtual Mode program is interrupted and execution passes to the interrupt routine at level 0, the VM bit is cleared. However, the VM bit is still set in the EFLAG image on the stack.

The 386SX Microprocessor operating system in turn handles the exception or interrupt and then returns control to the 8086 program. The 386SX Microprocessor operating system may choose to let the 8086 operating system handle the interrupt or it may emulate the function of the interrupt handler. For example, many 8086 operating system calls are accessed by PUSHing parameters on the stack, and then executing an INT n instruction. If the IOPL is set to 0 then all INT n instructions will be intercepted by the 386SX Microprocessor operating system.

An 386SX Microprocessor operating system can provide a Virtual 8086 Environment which is totally transparent to the application software by intercepting and then emulating 8086 operating system's calls, and intercepting IN and OUT instructions.

Entering and Leaving Virtual 8086 Mode

Virtual 8086 mode is entered by executing a 32-bit IRET instruction at CPL = 0 where the stack has a 1 in the VM bit of its EFLAGS image, or a Task Switch (at any CPL) to a 386SX Microprocessor task whose 386SX CPU TSS has a EFLAGS image containing a 1 in the VM bit position while the processor is executing in the Protected Mode. POPF does not affect the VM bit but a PUSHF always pushes a 0 in the VM bit.

The transition out of virtual 8086 mode to protected mode occurs only on receipt of an interrupt or exception. In Virtual 8086 mode, all interrupts and exceptions vector through the protected mode IDT, and enter an interrupt handler in protected mode. As part of the interrupt processing the VM bit is cleared.

Because the matching IRET must occur from level 0, Interrupt or Trap Gates used to field an interrupt or exception out of Virtual 8086 mode must perform an inter-level interrupt only to level 0. Interrupt or Trap Gates through conforming segments, or through segments with DPL > 0, will raise a GP fault with the CS selector as the error code.

Task Switches To/From Virtual 8086 Mode

Tasks which can execute in virtual 8086 mode must be described by a TSS with the 386SX CPU format (type 9 or 11 descriptor). A task switch out of virtual 8086 mode will operate exactly the same as any other task switch out of a task with a 386SX CPU TSS. All of the programmer visible state, including the EFLAGS register with the VM bit set to 1, is stored in the TSS. The segment registers in the TSS will contain 8086 segment base values rather than selectors.

A task switch into a task described by a 386SX CPU TSS will have an additional check to determine if the incoming task should be resumed in virtual 8086 mode. Tasks described by 286 format TSSs cannot be resumed in virtual 8086 mode, so no check is required there (the FLAGS image in 286 format TSS has only the low order 16 FLAGS bits). Before loading the segment register images from a 386SX CPU TSS, the FLAGS image is loaded, so that the segment registers are loaded from the TSS image as 8086 segment base values. The task is now ready to resume in virtual 8086 mode.

Transitions Through Trap and Interrupt Gates, and IRET

A task switch is one way to enter or exit virtual 8086 mode. The other method is to exit through a Trap or Interrupt gate, as part of handling an interrupt, and to enter as part of executing an IRET instruction. The transition out must use a 386SX CPU Trap Gate (Type 14), or 386SX CPU Interrupt Gate (Type 15), which must point to a non-conforming level 0 segment (DPL = 0) in order to permit the trap handler to IRET back to the Virtual 8086 program. The Gate must point to a non-conforming level 0 segment to perform a level switch to level 0 so that the matching IRET can change the VM bit. 386SX CPU gates must be used since 286 gates save only the low 16 bits of the EFLAGS register (the VM bit will not be saved). Also, the 16-bit IRET used to terminate the 286 interrupt handler will pop only the lower 16 bits from FLAGS, and will not affect the VM bit. The action taken for a 386SX CPU Trap or Interrupt gate if an interrupt occurs while the task is executing in virtual 8086 mode is given by the following sequence:

1. Save the FLAGS register in a temp to push later. Turn off the VM, TF, and IF bits.
2. Interrupt and Trap gates must perform a level switch from 3 (where the Virtual 8086 Mode program executes) to level 0 (so IRET can return).
3. Push the 8086 segment register values onto the new stack, in this order: GS, FS, DS, ES. These are pushed as 32-bit quantities. Then load these 4 registers with null selectors (0).
4. Push the old 8086 stack pointer onto the new stack by pushing the SS register (as 32-bits), then pushing the 32-bit ESP register saved above.
5. Push the 32-bit EFLAGS register saved in step 1.
6. Push the old 8086 instruction onto the new stack by pushing the CS register (as 32-bits), then pushing the 32-bit EIP register.
7. Load up the new CS:EIP value from the interrupt gate, and begin execution of the interrupt routine in protected mode.

The transition out of V86 mode performs a level change and stack switch, in addition to changing back to protected mode. Also all of the 8086 segment register images are stored on the stack (behind the SS:ESP image), and then loaded with null (0) selectors before entering the interrupt handler. This will permit the handler to safely save and restore the DS, ES, FS, and GS registers as 286 selectors. This is needed so that interrupt handlers which don't care about the mode of the interrupted program can use the same prologue and epilogue code for state saving regardless of whether or not a 'native' mode or Virtual 8086 Mode program was interrupted. Restoring null selectors to these registers

before executing the IRET will cause a trap in the interrupt handler. Interrupt routines which expect or return values in the segment registers will have to obtain/return values from the 8086 register images pushed onto the new stack. They will need to know the mode of the interrupted program in order to know where to find/return segment registers, and also to know how to interpret segment register values.

The IRET instruction will perform the inverse of the above sequence. Only the extended IRET instruction (operand size=32) can be used and must be executed at level 0 to change the VM bit to 1.

1. If the NT bit in the FLAGS register is on, an inter-task return is performed. The current state is stored in the current TSS, and the link field in the current TSS is used to locate the TSS for the interrupted task which is to be resumed. Otherwise, continue with the following sequence:
2. Read the FLAGS image from SS:8[ESP] into the FLAGS register. This will set VM to the value active in the interrupted routine.
3. Pop off the instruction pointer CS:EIP. EIP is popped first, then a 32-bit word is popped which contains the CS value in the lower 16 bits. If VM=0, this CS load is done as a protected mode segment load. If VM=1, this will be done as an 8086 segment load.
4. Increment the ESP register by 4 to bypass the FLAGS image which was 'popped' in step 1.
5. If VM=1, load segment registers ES, DS, FS, and GS from memory locations SS:[ESP+8], SS:[ESP+12], SS:[ESP+16], and SS:[ESP=20], respectively, where the new value of ESP stored in step 4 is used. Since VM=1, these are done as 8086 segment register loads.

   Else if VM=0, check that the selectors in ES, DS, FS, and GS are valid in the interrupted routine. Null out invalid selectors to trap if an attempt is made to access through them.
6. If RPL(CS)>CPL, pop the stack pointer SS:ESP from the stack. The ESP register is popped first, followed by 32-bits containing SS in the lower 16 bits. If VM=0, SS is loaded as a protected mode segment register load. If VM=1, an 8086 segment register load is used.
7. Resume execution of the interrupted routine. The VM bit in the FLAGS register (restored from the interrupt routine's stack image in step 1) determines whether the processor resumes the interrupted routine in Protected mode or Virtual 8086 Mode.

5.0 FUNCTIONAL DATA

The 386SX Microprocessor features a straightforward functional interface to the external hardware. The 386SX Microprocessor has separate parallel buses for data and address. The data bus is 16-bits in width, and bi-directional. The address bus outputs 24-bit address values using 23 address lines and two byte enable signals.

The 386SX Microprocessor has two selectable address bus cycles: address pipelined and non-address pipelined. The address pipelining option allows as much time as possible for data access by starting the pending bus cycle before the present bus cycle is finished. A non-pipelined bus cycle gives the highest bus performance by executing every bus cycle in two processor CLK cycles. For maximum design flexibility, the address pipelining option is selectable on a cycle-by-cycle basis.

The processor's bus cycle is the basic mechanism for information transfer, either from system to processor, or from processor to system. 386SX Microprocessor bus cycles perform data transfer in a minimum of only two clock periods. The maximum transfer bandwidth at 16 MHz is therefore 16 Mbytes/sec. However, any bus cycle will be extended for more than two clock periods if external hardware withholds acknowledgement of the cycle.

The 386SX Microprocessor can relinquish control of its local buses to allow mastership by other devices, such as direct memory access (DMA) channels. When relinquished, HLDA is the only output pin driven by the 386SX Microprocessor, providing near-complete isolation of the processor from its system (all other output pins are in a float condition).

5.1 Signal Description Overview

Ahead is a brief description of the 386SX Microprocessor input and output signals arranged by functional groups. Note the # symbol at the end of a signal name indicates the active, or asserted, state occurs when the signal is at a LOW voltage. When no # is present after the signal name, the signal is asserted when at the HIGH voltage level.

Example signal: M/IO# — HIGH voltage indicates Memory selected
— LOW voltage indicates I/O selected

The signal descriptions sometimes refer to AC timing parameters, such as 't_{25} Reset Setup Time' and 't_{26} Reset Hold Time.' The values of these parameters can be found in Table 7.4.

CLOCK (CLK2)

CLK2 provides the fundamental timing for the 386SX Microprocessor. It is divided by two internally to generate the internal processor clock used for instruction execution. The internal clock is comprised of two phases, 'phase one' and 'phase two'. Each CLK2 period is a phase of the internal clock. Figure 5.2 illustrates the relationship. If desired, the phase of the internal processor clock can be synchronized to a known phase by ensuring the falling edge of the RESET signal meets the applicable setup and hold times t_{25} and t_{26}.

DATA BUS (D_{15}–D_0)

These three-state bidirectional signals provide the general purpose data path between the 386SX Microprocessor and other devices. The data bus outputs are active HIGH and will float during bus hold acknowledge. Data bus reads require that read-data setup and hold times t_{21} and t_{22} be met relative to CLK2 for correct operation.

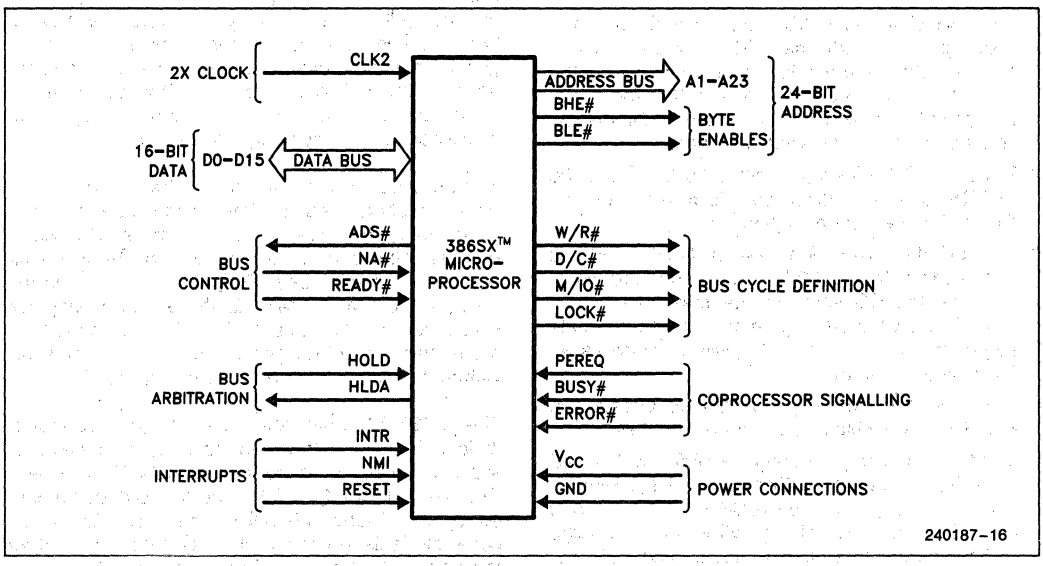


Figure 5.1. Functional Signal Groups

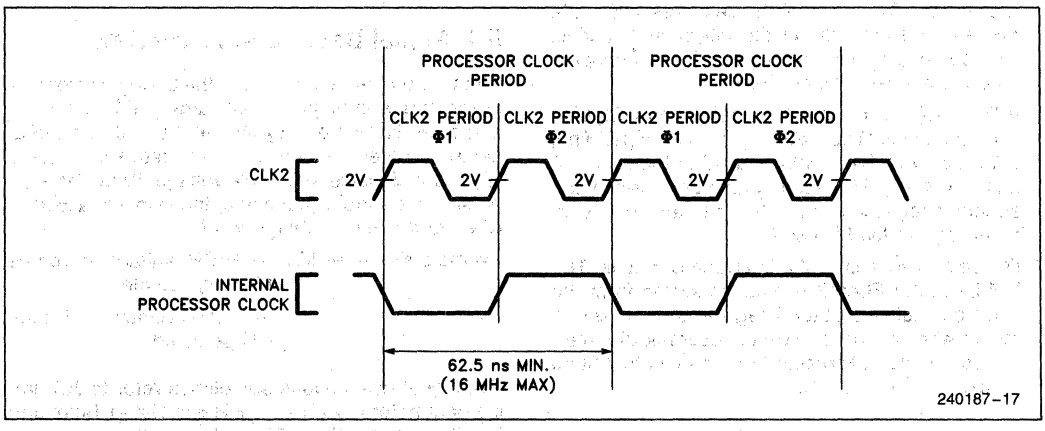


Figure 5.2. CLK2 Signal and Internal Processor Clock

ADDRESS BUS (A_{23}–A_1, BHE#, BLE#)

These three-state outputs provide physical memory addresses or I/O port addresses. A_{23}–A_{16} are LOW during I/O transfers except for I/O transfers automatically generated by coprocessor instructions. During coprocessor I/O transfers, A_{22}–A_{16} are driven LOW, and A_{23} is driven HIGH so that this address line can be used by external logic to generate the coprocessor select signal. Thus, the I/O address driven by the 386SX Microprocessor for coprocessor commands is 8000F8H, the I/O addresses driven by the 386SX Microprocessor for coprocessor data are 8000FCH or 8000FEH for cycles to the 80387SX.

The address bus is capable of addressing 16 megabytes of physical memory space (000000H through FFFFFFH), and 64 kilobytes of I/O address space (000000H through 00FFFFH) for programmed I/O. The address bus is active HIGH and will float during bus hold acknowledge.

The Byte Enable outputs, BHE# and BLE#, directly indicate which bytes of the 16-bit data bus are involved with the current transfer. BHE# applies to D_{15}–D_8 and BLE# applies to D_7–D_0. If both BHE# and BLE# are asserted, then 16 bits of data are being transferred. See Table 5.1 for a complete decoding of these signals. The byte enables are active LOW and will float during bus hold acknowledge.

BUS CYCLE DEFINITION SIGNALS (W/R#, D/C#, M/IO#, LOCK#)

These three-state outputs define the type of bus cycle being performed: W/R# distinguishes between write and read cycles, D/C# distinguishes between data and control cycles, M/IO# distinguishes between memory and I/O cycles, and LOCK# distinguishes between locked and unlocked bus cycles. All of these signals are active LOW and will float during bus acknowledge.

The primary bus cycle definition signals are W/R#, D/C# and M/IO#, since these are the signals driven valid as ADS# (Address Status output) becomes active. The LOCK# is driven valid at the same time the bus cycle begins, which due to address pipelining, could be after ADS# becomes active. Exact bus cycle definitions, as a function of W/R#, D/C#, and M/IO# are given in Table 5.2.

LOCK# indicates that other system bus masters are not to gain control of the system bus while it is active. LOCK# is activated on the CLK2 edge that begins the first locked bus cycle (i.e., it is not active at the same time as the other bus cycle definition pins) and is deactivated when ready is returned at the end of the last bus cycle which is to be locked. The beginning of a bus cycle is determined when READY# is returned in a previous bus cycle and another is pending (ADS# is active) or the clock in which ADS# is driven active if the bus was idle. This means that it follows more closely with the write data rules when it is valid, but may cause the bus to be locked longer than desired. The LOCK# signal may be explicitly activated by the LOCK prefix on certain instructions. LOCK# is always asserted when executing the XCHG instruction, during descriptor updates, and during the interrupt acknowledge sequence.

Table 5.1. Byte Enable Definitions

BHE#	BLE#	Function
0	0	Word Transfer
0	1	Byte transfer on upper byte of the data bus, D_{15}–D_8
1	0	Byte transfer on lower byte of the data bus, D_7–D_0
1	1	Never occurs

Table 5.2. Bus Cycle Definition

M/IO#	D/C#	W/R#	Bus Cycle Type	Locked?
0	0	0	Interrupt Acknowledge	Yes
0	0	1	does not occur	—
0	1	0	I/O Data Read	No
0	1	1	I/O Data Write	No
1	0	0	Memory Code Read	No
1	0	1	Halt: Address = 2, BHE# = 1, BLE# = 0 / Shutdown: Address = 0, BHE# = 1, BLE# = 0	No
1	1	0	Memory Data Read	Some Cycles
1	1	1	Memory Data Write	Some Cycles

BUS CONTROL SIGNALS (ADS#, READY#, NA#)

The following signals allow the processor to indicate when a bus cycle has begun, and allow other system hardware to control address pipelining and bus cycle termination.

Address Status (ADS#)

This three-state output indicates that a valid bus cycle definition and address (W/R#, D/C#, M/IO#, BHE#, BLE# and A_{23}–A_1) are being driven at the 386SX Microprocessor pins. ADS# is an active LOW output. Once ADS# is driven active, valid address, byte enables, and definition signals will not change. In addition, ADS# will remain active until its associated bus cycle begins (when READY# is returned for the previous bus cycle when running pipelined bus cycles). When address pipelining is utilized, maximum throughput is achieved by initiating bus cycles when ADS# and READY# are active in the same clock cycle. ADS# will float during bus hold acknowledge. See sections **Non-Pipelined Address** (page 49) and **Pipelined Address** (page 50) for additional information on how ADS# is asserted for different bus states.

Transfer Acknowledge (READY#)

This input indicates the current bus cycle is complete, and the active bytes indicated by BHE# and BLE# are accepted or provided. When READY# is sampled active during a read cycle or interrupt acknowledge cycle, the 386SX Microprocessor latches the input data and terminates the cycle. When READY# is sampled active during a write cycle, the processor terminates the bus cycle.

READY# is ignored on the first bus state of all bus cycles, and sampled each bus state thereafter until asserted. READY# must eventually be asserted to acknowledge every bus cycle, including Halt Indication and Shutdown Indication bus cycles. When being sampled, READY# must always meet setup and hold times t_{19} and t_{20} for correct operation.

Next Address Request (NA#)

This is used to request address pipelining. This input indicates the system is prepared to accept new values of BHE#, BLE#, A_{23}–A_1, W/R#, D/C# and M/IO# from the 386SX Microprocessor even if the end of the current cycle is not being acknowledged on READY#. If this input is active when sampled, the next address is driven onto the bus, provided the next bus request is already pending internally. NA# is ignored in CLK cycles in which ADS# or READY# is activated. This signal is active LOW and must satisfy setup and hold times t_{15} and t_{16} for correct operation. See **Pipelined Address** (page 50) and **Read and Write Cycles** (page 47) for additional information.

BUS ARBITRATION SIGNALS (HOLD, HLDA)

This section describes the mechanism by which the processor relinquishes control of its local buses when requested by another bus master device. See **Entering and Exiting Hold Acknowledge** (page 57) for additional information.

Bus Hold Request (HOLD)

This input indicates some device other than the 386SX Microprocessor requires bus mastership. When control is granted, the 386SX Microprocessor floats A_{23}–A_1, BHE#, BLE#, D_{15}–D_0, LOCK#, M/IO#, D/C#, W/R# and ADS#, and then activates HLDA, thus entering the bus hold acknowledge state. The local bus will remain granted to the requesting master until HOLD becomes inactive. When HOLD becomes inactive, the 386SX Microprocessor will deactivate HLDA and drive the local bus (at the same time), thus terminating the hold acknowledge condition.

HOLD must remain asserted as long as any other device is a local bus master. External pull-up resistors may be required when in the hold acknowledge state since none of the 386SX Microprocessor floated outputs have internal pull-up resistors. See **Resistor Recommendations** (page 64) for additional information. HOLD is not recognized while RESET is active. If RESET is asserted while HOLD is asserted, RESET has priority and places the bus into an idle state, rather than the hold acknowledge (high-impedance) state.

HOLD is a level-sensitive, active HIGH, synchronous input. HOLD signals must always meet setup and hold times t_{23} and t_{24} for correct operation.

Bus Hold Acknowledge (HLDA)

When active (HIGH), this output indicates the 386SX Microprocessor has relinquished control of its local bus in response to an asserted HOLD signal, and is in the bus Hold Acknowledge state.

The Bus Hold Acknowledge state offers near-complete signal isolation. In the Hold Acknowledge state, HLDA is the only signal being driven by the 386SX Microprocessor. The other output signals or bidirectional signals (D_{15}–D_0, BHE#, BLE#, A_{23}–A_1, W/R#, D/C#, M/IO#, LOCK# and ADS#) are in a high-impedance state so the requesting bus

master may control them. These pins remain OFF throughout the time that HLDA remains active (see Table 5.3)). Pull-up resistors may be desired on several signals to avoid spurious activity when no bus master is driving them. See **Resistor Recommendations** (page 64) for additional information.

When the HOLD signal is made inactive, the 386SX Microprocessor will deactivate HLDA and drive the bus. One rising edge on the NMI input is remembered for processing after the HOLD input is negated.

Table 5.3. Output pin State During HOLD

Pin Value	Pin Names
1	HLDA
Float	LOCK#, M/IO#, D/C#, W/R#, ADS#, A_{23}–A_1, BHE#, BLE#, D_{15}–D_0

In addition to the normal usage of Hold Acknowledge with DMA controllers or master peripherals, the near-complete isolation has particular attractiveness during system test when test equipment drives the system, and in hardware fault-tolerant applications.

HOLD Latencies

The maximum possible HOLD latency depends on the software being executed. The actual HOLD latency at any time depends on the current bus activity, the state of the LOCK# signal (internal to the CPU) activated by the LOCK# prefix, and interrupts. The 386SX Microprocessor will not honor a HOLD request until the current bus operation is complete. Table 5.4 shows the types of bus operations that can affect HOLD latency, and indicates the types of delays that these operations may introduce. When considering maximum HOLD latencies, designers must select which of these bus operations are possible, and then select the maximum latency from among them.

The 386SX Microprocessor breaks 32-bit data or I/O accesses into 2 internally locked 16-bit bus cycles; the LOCK# signal is not asserted. The 386SX Microprocessor breaks unaligned 16-bit or 32-bit data or I/O accesses into 2 or 3 internally locked 16-bit bus cycles. Again, the LOCK# signal is not asserted but a HOLD request will not be recognized until the end of the entire transfer.

As indicated in Table 5.4, wait states affect HOLD latency. The 386SX Microprocessor will not honor a HOLD request until the end of the current bus operation, no matter how many wait states are required. Systems with DMA where data transfer is critical must insure that READY# returns sufficiently soon.

{*** **Not Available At This Time** ***}

Table 5.4. Locked Bus Operations Affecting HOLD Latency in Systems Clocks

COPROCESSOR INTERFACE SIGNALS (PEREQ, BUSY#, ERROR#)

In the following sections are descriptions of signals dedicated to the numeric coprocessor interface. In addition to the data bus, address bus, and bus cycle definition signals, these following signals control communication between the 386SX Microprocessor and its 80387SX processor extension.

Coprocessor Request (PEREQ)

When asserted (HIGH), this input signal indicates a coprocessor request for a data operand to be transferred to/from memory by the 386SX Microprocessor. In response, the 386SX Microprocessor transfers information between the coprocessor and memory. Because the 386SX Microprocessor has internally stored the coprocessor opcode being executed, it performs the requested data transfer with the correct direction and memory address.

PEREQ is a level-sensitive active HIGH asynchronous signal. Setup and hold times, t_{29} and t_{30}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. This signal is provided with a weak internal pull-down resistor of around 20 K-ohms to ground so that it will not float active when left unconnected.

Coprocessor Busy (BUSY#)

When asserted (LOW), this input indicates the coprocessor is still executing an instruction, and is not yet able to accept another. When the 386SX Microprocessor encounters any coprocessor instruction which operates on the numerics stack (e.g. load, pop, or arithmetic operation), or the WAIT instruction, this input is first automatically sampled until it is seen to be inactive. This sampling of the BUSY# input prevents overrunning the execution of a previous coprocessor instruction.

The FNINIT, FNSTENV, FNSAVE, FNSTSW, FNSTCW and FNCLEX coprocessor instructions are allowed to execute even if BUSY# is active, since these instructions are used for coprocessor initialization and exception-clearing.

BUSY# is an active LOW, level-sensitive asynchronous signal. Setup and hold times, t_{29} and t_{30}, rela-

tive to the CLK2 signal must be met to guarantee recognition at a particular clock edge. This pin is provided with a weak internal pull-up resistor of around 20 K-ohms to Vcc so that it will not float active when left unconnected.

BUSY# serves an additional function. If BUSY# is sampled LOW at the falling edge of RESET, the 386SX Microprocessor performs an internal self-test (see **Bus Activity During and Following Reset**, page 58). If BUSY# is sampled HIGH, no self-test is performed.

Coprocessor Error (ERROR#)

When asserted (LOW), this input signal indicates that the previous coprocessor instruction generated a coprocessor error of a type not masked by the coprocessor's control register. This input is automatically sampled by the 386SX Microprocessor when a coprocessor instruction is encountered, and if active, the 386SX Microprocessor generates exception 16 to access the error-handling software.

Several coprocessor instructions, generally those which clear the numeric error flags in the coprocessor or save coprocessor state, do execute without the 386SX Microprocessor generating exception 16 even if ERROR# is active. These instructions are FNINIT, FNCLEX, FNSTSW, FNSTSWAX, FNSTCW, FNSTENV and FNSAVE.

ERROR# is an active LOW, level-sensitive asynchronous signal. Setup and hold times, t_{29} and t_{30}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. This pin is provided with a weak internal pull-up resistor of around 20 K-ohms to Vcc so that it will not float active when left unconnected.

INTERRUPT SIGNALS (INTR, NMI, RESET)

The following descriptions cover inputs that can interrupt or suspend execution of the processor's current instruction stream.

Maskable Interrupt Request (INTR)

When asserted, this input indicates a request for interrupt service, which can be masked by the 386SX CPU Flag Register IF bit. When the 386SX Microprocessor responds to the INTR input, it performs two interrupt acknowledge bus cycles and, at the end of the second, latches an 8-bit interrupt vector on D_7–D_0 to identify the source of the interrupt.

INTR is an active HIGH, level-sensitive asynchronous signal. Setup and hold times, t_{27} and t_{28}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. To assure recognition of an INTR request, INTR should remain active until the first interrupt acknowledge bus cycle begins. INTR is sampled at the beginning of every instruction in the 386SX Microprocessor's Execution Unit. In order to be recognized at a particular instruction boundary, INTR must be active at least eight CLK2 clock periods before the beginning of the instruction. If recognized, the 386SX Microprocessor will begin execution of the interrupt.

Non-Maskable Interrupt Request (NMI))

This input indicates a request for interrupt service which cannot be masked by software. The non-maskable interrupt request is always processed according to the pointer or gate in slot 2 of the interrupt table. Because of the fixed NMI slot assignment, no interrupt acknowledge cycles are performed when processing NMI.

NMI is an active HIGH, rising edge-sensitive asynchronous signal. Setup and hold times, t_{27} and t_{28}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. To assure recognition of NMI, it must be inactive for at least eight CLK2 periods, and then be active for at least eight CLK2 periods before the beginning of the instruction boundary in the 386SX Microprocessor's Execution Unit.

Once NMI processing has begun, no additional NMI's are processed until after the next IRET instruction, which is typically the end of the NMI service routine. If NMI is re-asserted prior to that time, however, one rising edge on NMI will be remembered for processing after executing the next IRET instruction.

Interrupt Latency

The time that elapses before an interrupt request is serviced (interrupt latency) varies according to several factors. This delay must be taken into account by the interrupt source. Any of the following factors can affect interrupt latency:

1. If interrupts are masked, an INTR request will not be recognized until interrupts are reenabled.
2. If an NMI is currently being serviced, an incoming NMI request will not be recognized until the 386SX Microprocessor encounters the IRET instruction.
3. An interrupt request is recognized only on an instruction boundary of the 386SX Microprocessor's Execution Unit except for the following cases:
   — Repeat string instructions can be interrupted after each iteration.

— If the instruction loads the Stack Segment register, an interrupt is not processed until after the following instruction, which should be an ESP. This allows the entire stack pointer to be loaded without interruption.

— If an instruction sets the interrupt flag (enabling interrupts), an interrupt is not processed until after the next instruction.

The longest latency occurs when the interrupt request arrives while the 386SX Microprocessor is executing a long instruction such as multiplication, division, or a task-switch in the protected mode.

4. Saving the Flags register and CS:EIP registers.
5. If interrupt service routine requires a task switch, time must be allowed for the task switch.
6. If the interrupt service routine saves registers that are not automatically saved by the 386SX Microprocessor.

RESET

This input signal suspends any operation in progress and places the 386SX Microprocessor in a known reset state. The 386SX Microprocessor is reset by asserting RESET for 15 or more CLK2 periods (80 or more CLK2 periods before requesting self-test). When RESET is active, all other input pins are ignored, and all other bus pins are driven to an idle bus state as shown in Table 5.5. If RESET and HOLD are both active at a point in time, RESET takes priority even if the 386SX Microprocessor was in a Hold Acknowledge state prior to RESET active.

RESET is an active HIGH, level-sensitive synchronous signal. Setup and hold times, t_{25} and t_{26}, must be met in order to assure proper operation of the 386SX Microprocessor.

Table 5.5. Pin State (Bus Idle) During Reset

Pin Name	Signal Level During Reset
ADS#	1
D_{15}–D_0	Float
BHE#, BLE#	0
A_{23}–A_1	1
W/R#	0
D/C#	1
M/IO#	0
LOCK#	1
HLDA	0

5.2 Bus Transfer Mechanism

All data transfers occur as a result of one or more bus cycles. Logical data operands of byte and word lengths may be transferred without restrictions on physical address alignment. Any byte boundary may be used, although two physical bus cycles are performed as required for unaligned operand transfers.

The 386SX Microprocessor address signals are designed to simplify external system hardware. Higher-order address bits are provided by A_{23}–A_1. BHE# and BLE# provide linear selects for the two bytes of the 16-bit data bus.

Byte Enable outputs BHE# and BLE# are asserted when their associated data bus bytes are involved with the present bus cycle, as listed in Table 5.6.

Table 5.6. Byte Enables and Associated Data and Operand Bytes

Byte Enable Signal	Associated Data Bus Signals	
BLE#	D_7–D_0	(byte 0 — least significant)
BHE#	D_{15}–D_8	(byte 1 — most significant)

Each bus cycle is composed of at least two bus states. Each bus state requires one processor clock period. Additional bus states added to a single bus cycle are called wait states. See section **5.4 Bus Functional Description**.

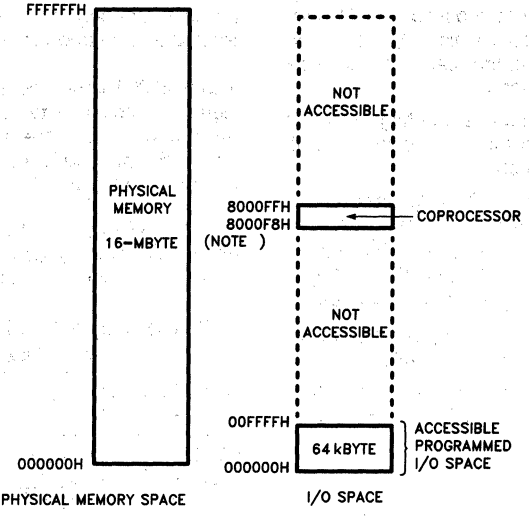
5.3 Memory and I/O Spaces

Bus cycles may access physical memory space or I/O space. Peripheral devices in the system may either be memory-mapped, or I/O-mapped, or both. As shown in Figure 5.3, physical memory addresses range from 000000H to 0FFFFFFH (16 megabytes) and I/O addresses from 000000H to 00FFFFH (64 kilobytes). Note the I/O addresses used by the automatic I/O cycles for coprocessor communication are 8000F8H to 8000FFH, beyond the address range of programmed I/O, to allow easy generation of a coprocessor chip select signal using the A_{23} and M/IO# signals.

5.4 Bus Functional Description

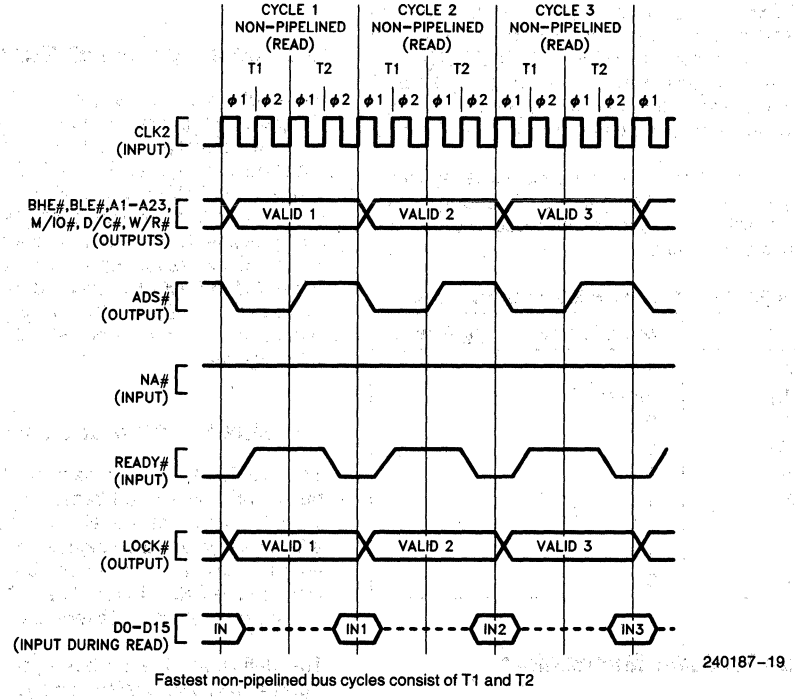
The 386SX Microprocessor has separate, parallel buses for data and address. The data bus is 16-bits in width, and bidirectional. The address bus provides a 24-bit value using 23 signals for the 23 upper-order address bits and 2 Byte Enable signals to directly indicate the active bytes. These buses are interpreted and controlled by several definition signals.

The definition of each bus cycle is given by three signals: M/IO#, W/R# and D/C#. At the same time, a valid address is present on the byte enable signals, BHE# and BLE#, and the other address signals A_{23}–A_1. A status signal, ADS#, indicates



NOTE:
Since A23 is HIGH during automatic communication with coprocessor, A23 HIGH and M/IO# LOW can be used to easily generate a coprocessor select signal.

Figure 5.3. Physical Memory and I/O Spaces



Fastest non-pipelined bus cycles consist of T1 and T2

Figure 5.4. Fastest Read Cycles with Non-pipelined Address Timing

when the 386SX Microprocessor issues a new bus cycle definition and address.

Collectively, the address bus, data bus and all associated control signals are referred to simply as 'the bus'. When active, the bus performs one of the bus cycles below:

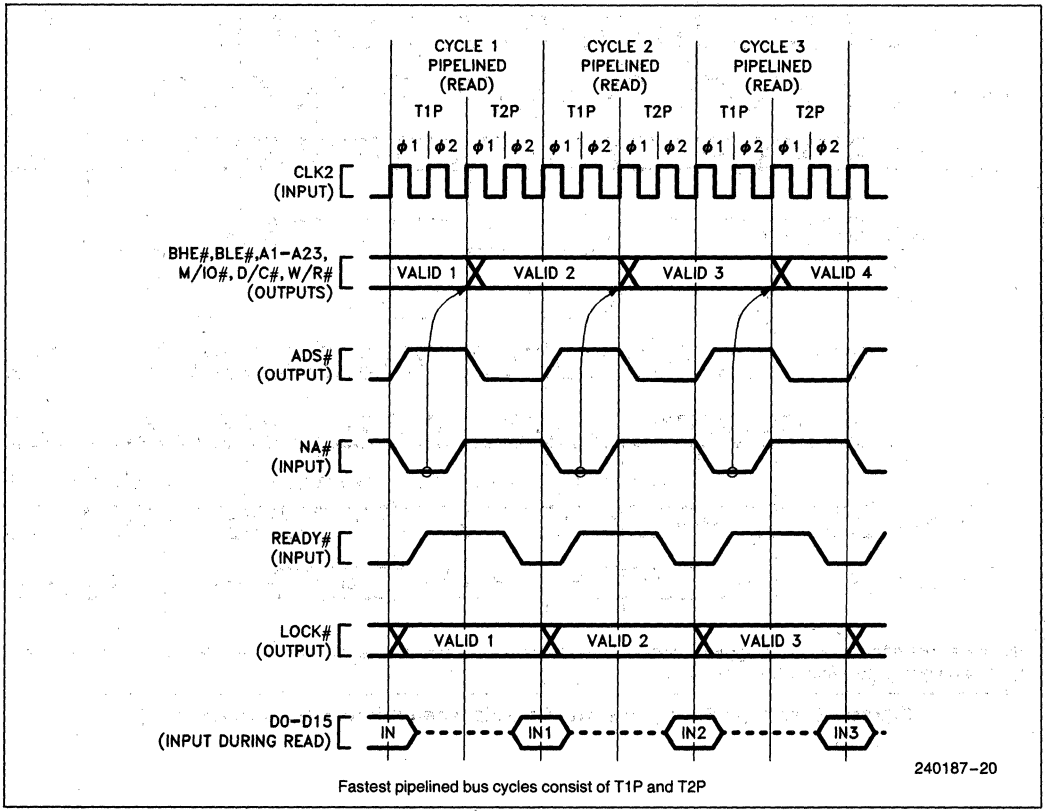
1. Read from memory space
2. Locked read from memory space
3. Write to memory space
4. Locked write to memory space
5. Read from I/O space (or coprocessor)
6. Write to I/O space (or coprocessor)
7. Interrupt acknowledge (always locked)
8. Indicate halt, or indicate shutdown

Table 5.2 shows the encoding of the bus cycle definition signals for each bus cycle. See **Bus Cycle Definition Signals** (page 40) for additional information.

When the 386SX Microprocessor bus is not performing one of the activities listed above, it is either Idle or in the Hold Acknowledge state, which may be detected externally. The idle state can be identified by the 386SX Microprocessor giving no further assertions on its address strobe output (ADS#) since the beginning of its most recent bus cycle, and the most recent bus cycle having been terminated. The hold acknowledge state is identified by the 386SX Microprocessor asserting its hold acknowledge (HLDA) output.

The shortest time unit of bus activity is a bus state. A bus state is one processor clock period (two CLK2 periods) in duration. A complete data transfer occurs during a bus cycle, composed of two or more bus states.

The fastest 386SX Microprocessor bus cycle requires only two bus states. For example, three consecutive bus read cycles, each consisting of two bus states, are shown by Figure 5.4. The bus states in each cycle are named T1 and T2. Any memory or I/O address may be accessed by such a two-state bus cycle, if the external hardware is fast enough.



Fastest pipelined bus cycles consist of T1P and T2P

Figure 5.5. Fastest Read Cycles with Pipelined Address Timing

Every bus cycle continues until it is acknowledged by the external system hardware, using the 386SX Microprocessor READY# input. Acknowledging the bus cycle at the end of the first T2 results in the shortest bus cycle, requiring only T1 and T2. If READY# is not immediately asserted however, T2 states are repeated indefinitely until the READY# input is sampled active.

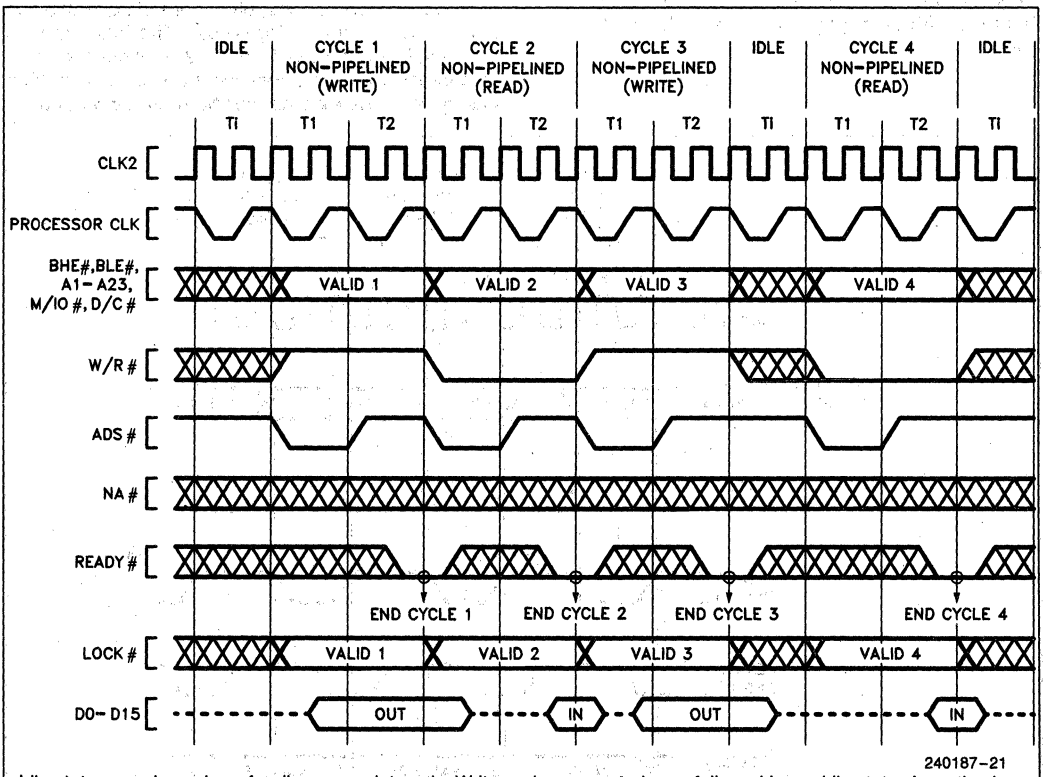
The address pipelining option provides a choice of bus cycle timings. Pipelined or non-pipelined address timing is selectable on a cycle-by-cycle basis with the Next Address (NA#) input.

When address pipelining is selected the address (BHE#, BLE# and A_{23}–A_1) and definition (W/R#, D/C#, M/IO# and LOCK#) of the next cycle are available before the end of the current cycle. To signal their availability, the 386SX Microprocessor address status output (ADS#) is asserted. Figure 5.5 illustrates the fastest read cycles with pipelined address timing.

Note from Figure 5.5 the fastest bus cycles using pipelined address require only two bus states, named **T1P** and **T2P**. Therefore cycles with pipelined address timing allow the same data bandwidth as non-pipelined cycles, but address-to-data access time is increased by one T-state time compared to that of a non-pipelined cycle.

READ AND WRITE CYCLES

Data transfers occur as a result of bus cycles, classified as read or write cycles. During read cycles, data is transferred from an external device to the processor. During write cycles, data is transferred from the processor to an external device.



Idle states are shown here for diagram variety only. Write cycles are **not** always followed by an idle state. An active bus cycle can immediately follow the write cycle.

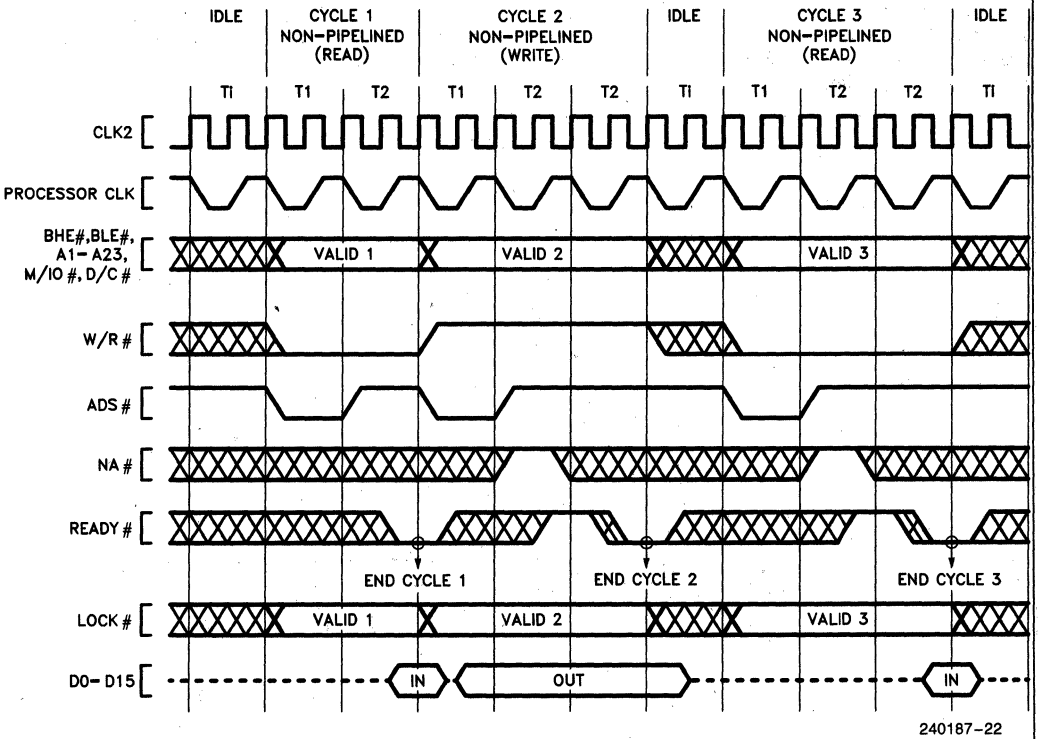
Figure 5.6. Various Bus Cycles with Non-Pipelined Address (zero wait states)

Two choices of address timing are dynamically selectable: non-pipelined or pipelined. After an idle bus state, the processor always uses non-pipelined address timing. However the NA# (Next Address) input may be asserted to select pipelined address timing for the next bus cycle. When pipelining is selected and the 386SX Microprocessor has a bus request pending internally, the address and definition of the next cycle is made available even before the current bus cycle is acknowledged by READY#.

Terminating a read or write cycle, like any bus cycle, requires acknowledging the cycle by asserting the READY# input. Until acknowledged, the processor inserts wait states into the bus cycle, to allow adjustment for the speed of any external device. External hardware, which has decoded the address and bus cycle type, asserts the READY# input at the appropriate time.

At the end of the second bus state within the bus cycle, READY# is sampled. At that time, if external hardware acknowledges the bus cycle by asserting READY#, the bus cycle terminates as shown in Figure 5.6. If READY# is negated as in Figure 5.7, the 386SX Microprocessor executes another bus state (a wait state) and READY# is sampled again at the end of that state. This continues indefinitely until the cycle is acknowledged by READY# asserted.

When the current cycle is acknowledged, the 386SX Microprocessor terminates it. When a read cycle is acknowledged, the 386SX Microprocessor latches the information present at its data pins. When a write cycle is acknowledged, the 386SX CPU's write data remains valid throughout phase one of the next bus state, to provide write data hold time.



Idle states are shown here for diagram variety only. Write cycles are **not** always followed by an idle state. An active bus cycle can immediately follow the write cycle.

Figure 5.7. Various Bus Cycles with Non-Pipelined Address (various number of wait states)

Non-Pipelined Address

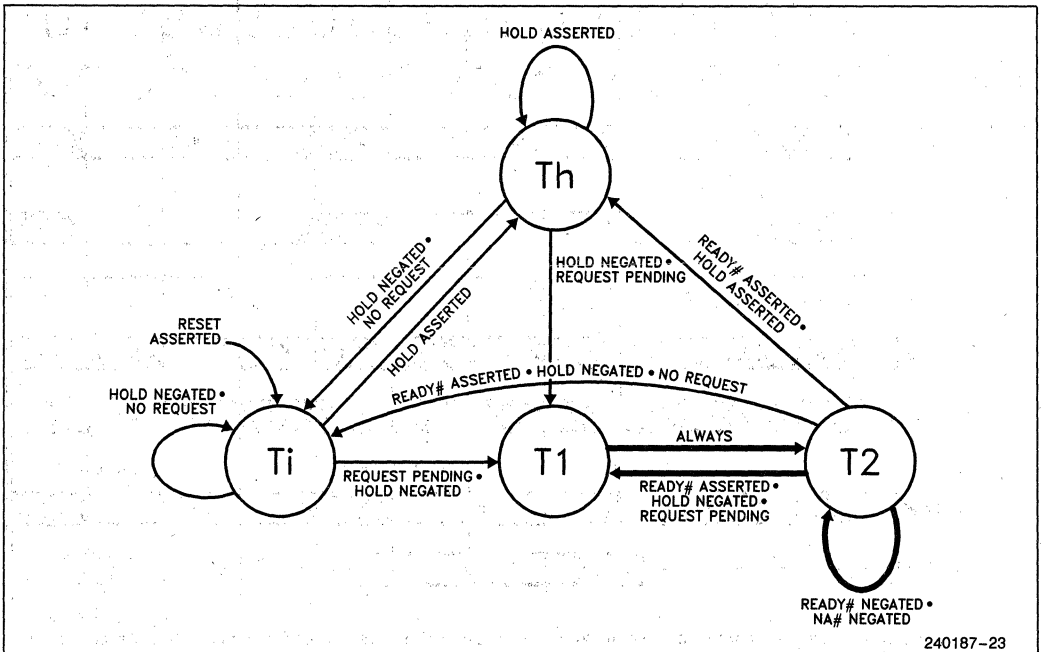
Any bus cycle may be performed with non-pipelined address timing. For example, Figure 5.6 shows a mixture of read and write cycles with non-pipelined address timing. Figure 5.6 shows that the fastest possible cycles with non-pipelined address have two bus states per bus cycle. The states are named T1 and T2. In phase one of T1, the address signals and bus cycle definition signals are driven valid and, to signal their availability, address strobe (ADS#) is simultaneously asserted.

During read or write cycles, the data bus behaves as follows. If the cycle is a read, the 386SX Microprocessor floats its data signals to allow driving by the external device being addressed. **The 386SX Microprocessor requires that all data bus pins be at a valid logic state (HIGH or LOW) at the end of each read cycle, when READY# is asserted. The system MUST be designed to meet this requirement.** If the cycle is a write, data signals are driven by the 386SX Microprocessor beginning in phase two of T1 until phase one of the bus state following cycle acknowledgment.

Figure 5.7 illustrates non-pipelined bus cycles with one wait state added to Cycles 2 and 3. READY# is sampled inactive at the end of the first T2 in Cycles 2 and 3. Therefore Cycles 2 and 3 have T2 repeated again. At the end of the second T2, READY# is sampled active.

When address pipelining is not used, the address and bus cycle definition remain valid during all wait states. When wait states are added and it is desirable to maintain non-pipelined address timing, it is necessary to negate NA# during each T2 state except the last one, as shown in Figure 5.7 Cycles 2 and 3. If NA# is sampled active during a T2 other than the last one, the next state would be T2I or T2P instead of another T2.

When address pipelining is not used, the bus states and transitions are completely illustrated by Figure 5.8. The bus transitions between four possible states, T1, T2, T_i, and T_h. Bus cycles consist of T1 and T2, with T2 being repeated for wait states. Otherwise the bus may be idle, T_i, or in the hold acknowledge state T_h.



Bus States:
T1—first clock of a non-pipelined bus cycle (386SX™ CPU drives new address and asserts ADS#).
T2—subsequent clocks of a bus cycle when NA# has not been sampled asserted in the current bus cycle.
Ti—idle state.
Th—hold acknowledge state (386SX™ CPU asserts HLDA).
The fastest bus cycle consists of two states T1 and T2.
Four basic bus states describe bus operation when not using pipelined address.

Figure 5.8. Bus States (not using pipelined address)

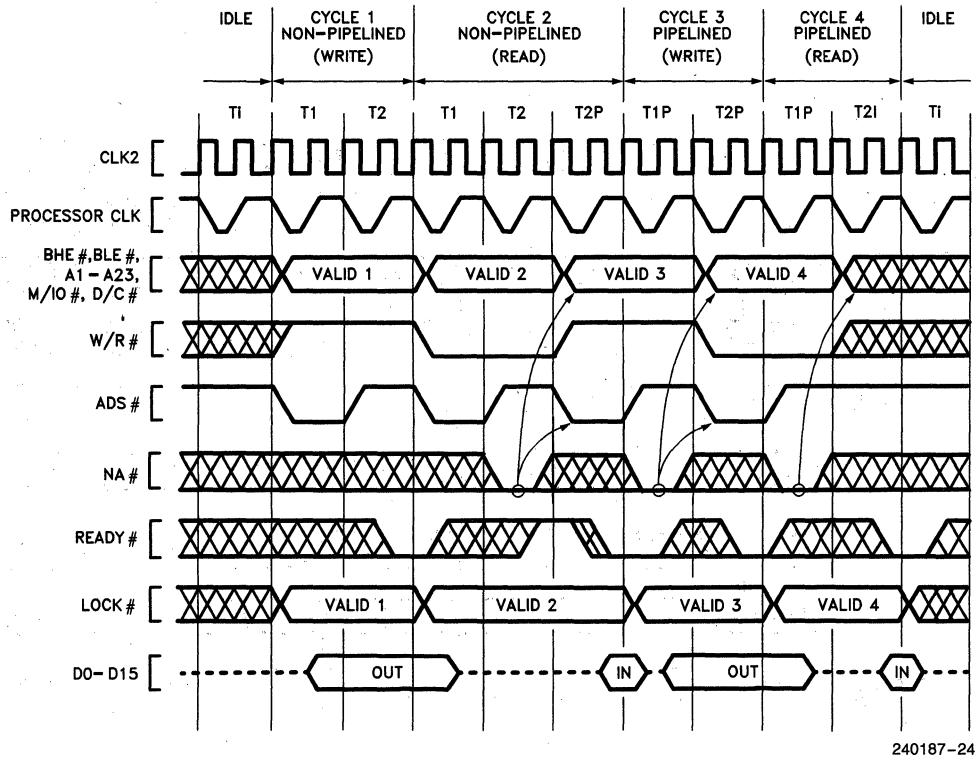
Bus cycles always begin with T1. T1 always leads to T2. If a bus cycle is not acknowledged during T2 and NA# is inactive, T2 is repeated. When a cycle is acknowledged during T2, the following state will be T1 of the next bus cycle if a bus request is pending internally, or T_i if there is no bus request pending, or T_h if the HOLD input is being asserted.

Use of pipelined address allows the 386SX Microprocessor to enter three additional bus states not shown in Figure 5.8. Figure 5.12 on page 53 is the complete bus state diagram, including pipelined address cycles.

Pipelined Address

Address pipelining is the option of requesting the address and the bus cycle definition of the next internally pending bus cycle before the current bus cycle is acknowledged with READY# asserted. ADS# is asserted by the 386SX Microprocessor when the next address is issued. The address pipelining option is controlled on a cycle-by-cycle basis with the NA# input signal.

Once a bus cycle is in progress and the current address has been valid for at least one entire bus state, the NA# input is sampled at the end of every phase one until the bus cycle is acknowledged. During non-pipelined bus cycles NA# is sampled at the end of phase one in every T2. An example is Cycle 2 in Figure 5.9, during which NA# is sampled at the end of phase one of every T2 (it was asserted once during the first T2 and has no further effect during that bus cycle).



240187-24

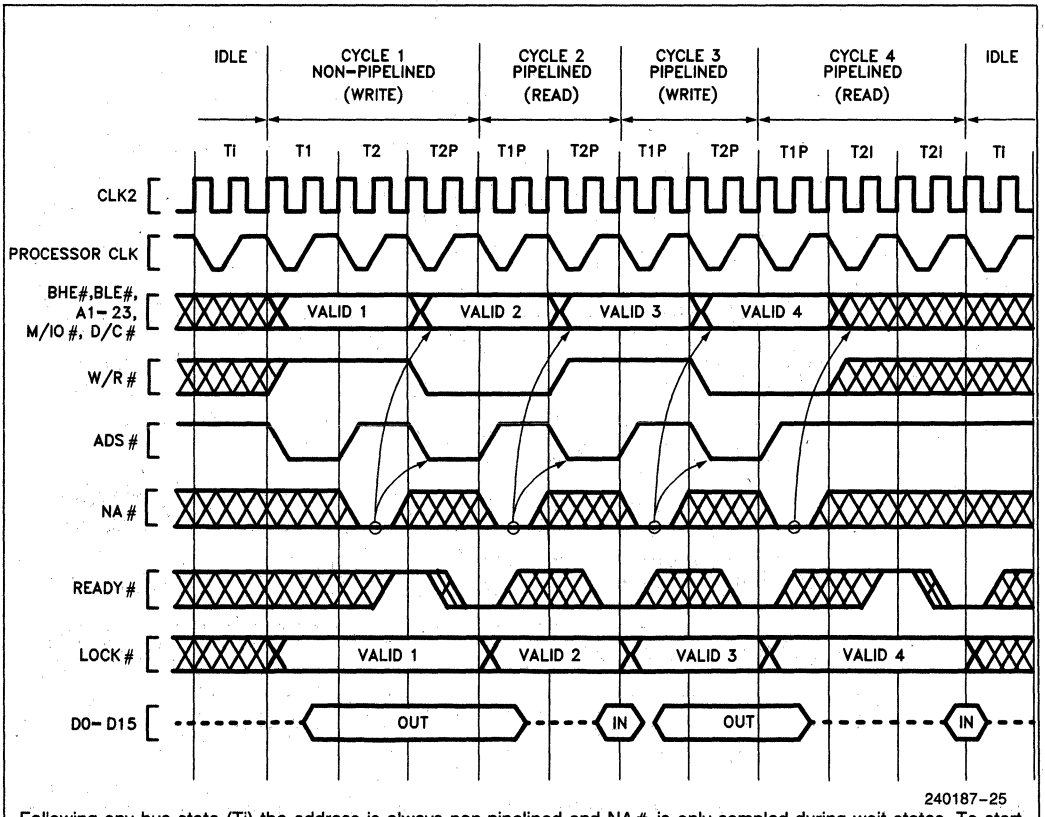
Following any idle bus state (Ti), addresses are non-pipelined. Within non-pipelined bus cycles, NA# is only sampled during wait states. Therefore, to begin address pipelining during a group of non-pipelined bus cycles requires a non-pipelined cycle with at least one wait state (Cycle 2 above).

Figure 5.9. Transitioning to Pipelined Address During Burst of Bus Cycles

If NA# is sampled active, the 386SX Microprocessor is free to drive the address and bus cycle definition of the next bus cycle, and assert ADS#, as soon as it has a bus request internally pending. It may drive the next address as early as the next bus state, whether the current bus cycle is acknowledged at that time or not.

Regarding the details of address pipelining, the 386SX Microprocessor has the following characteristics:

1. The next address may appear as early as the bus state after NA# was sampled active (see Figures 5.9 or 5.10). In that case, state T2P is entered immediately. However, when there is not an internal bus request already pending, the next address will not be available immediately after NA# is asserted and T2I is entered instead of T2P (see Figure 5.11 Cycle 3). Provided the current bus cycle isn't yet acknowledged by READY# asserted, T2P will be entered as soon as the 386SX Microprocessor does drive the next address. External hardware should therefore observe the ADS# output as confirmation the next address is actually being driven on the bus.
2. Any address which is validated by a pulse on the ADS# output will remain stable on the address pins for at least two processor clock periods. The 386SX Microprocessor cannot produce a new address more frequently than every two processor clock periods (see Figures 5.9, 5.10, and 5.11).
3. Only the address and bus cycle definition of the very next bus cycle is available. The pipelining capability cannot look further than one bus cycle ahead (see Figure 5.11 Cycle 1).



Following any bus state (Ti) the address is always non-pipelined and NA# is only sampled during wait states. To start address pipelining after an idle state requires a non-pipelined cycle with at least one wait state (cycle 1 above) The pipelined cycles (2, 3, 4 above) are shown with various numbers of wait states.

Figure 5.10. Fastest Transition to Pipelined Address Following Idle Bus State

The complete bus state transition diagram, including operation with pipelined address is given by Figure 5.12. Note it is a superset of the diagram for non-pipelined address only, and the three additional bus states for pipelined address are drawn in bold.

The fastest bus cycle with pipelined address consists of just two bus states, T1P and T2P (recall for non-pipelined address it is T1 and T2). T1P is the first bus state of a pipelined cycle.

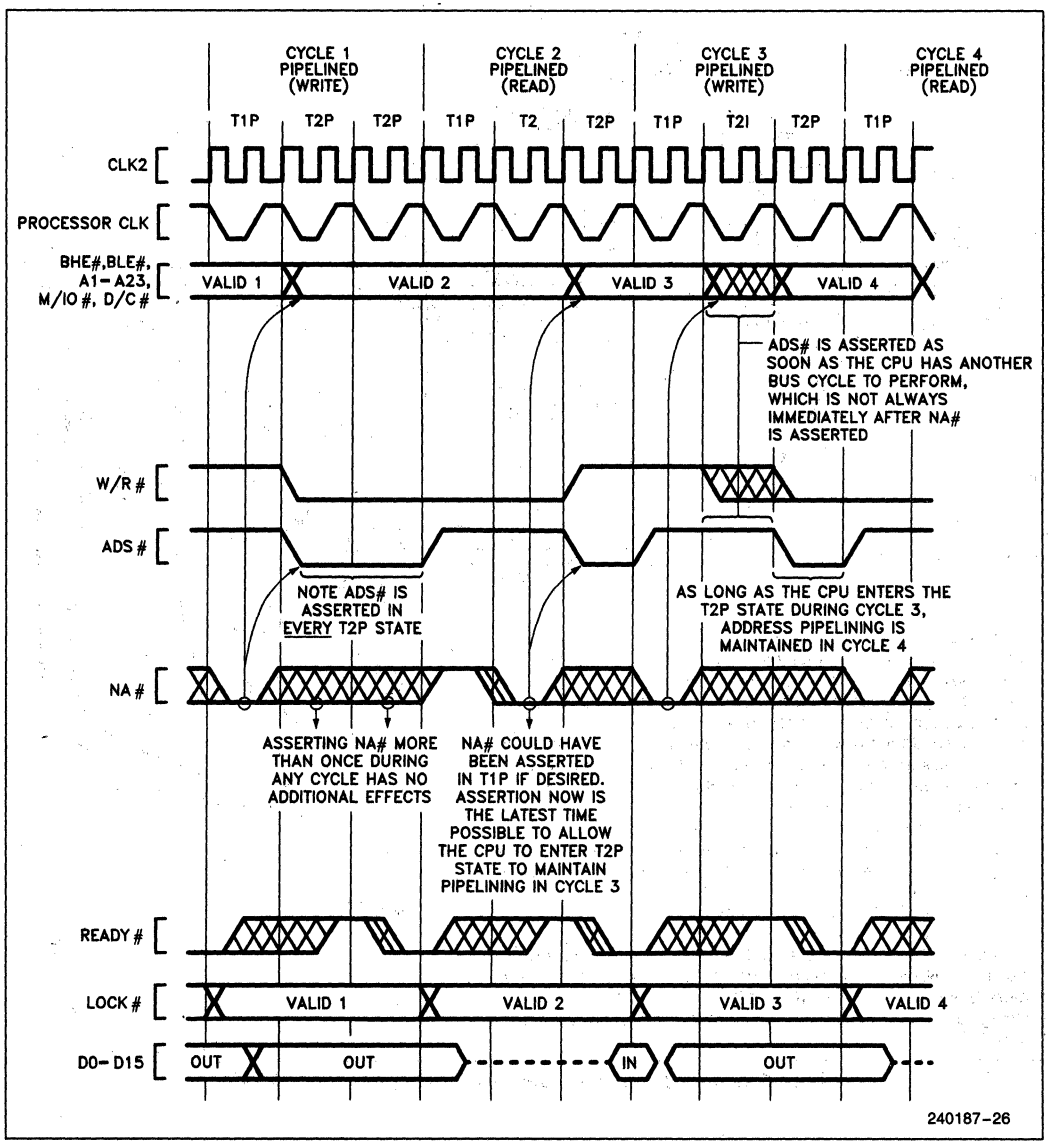
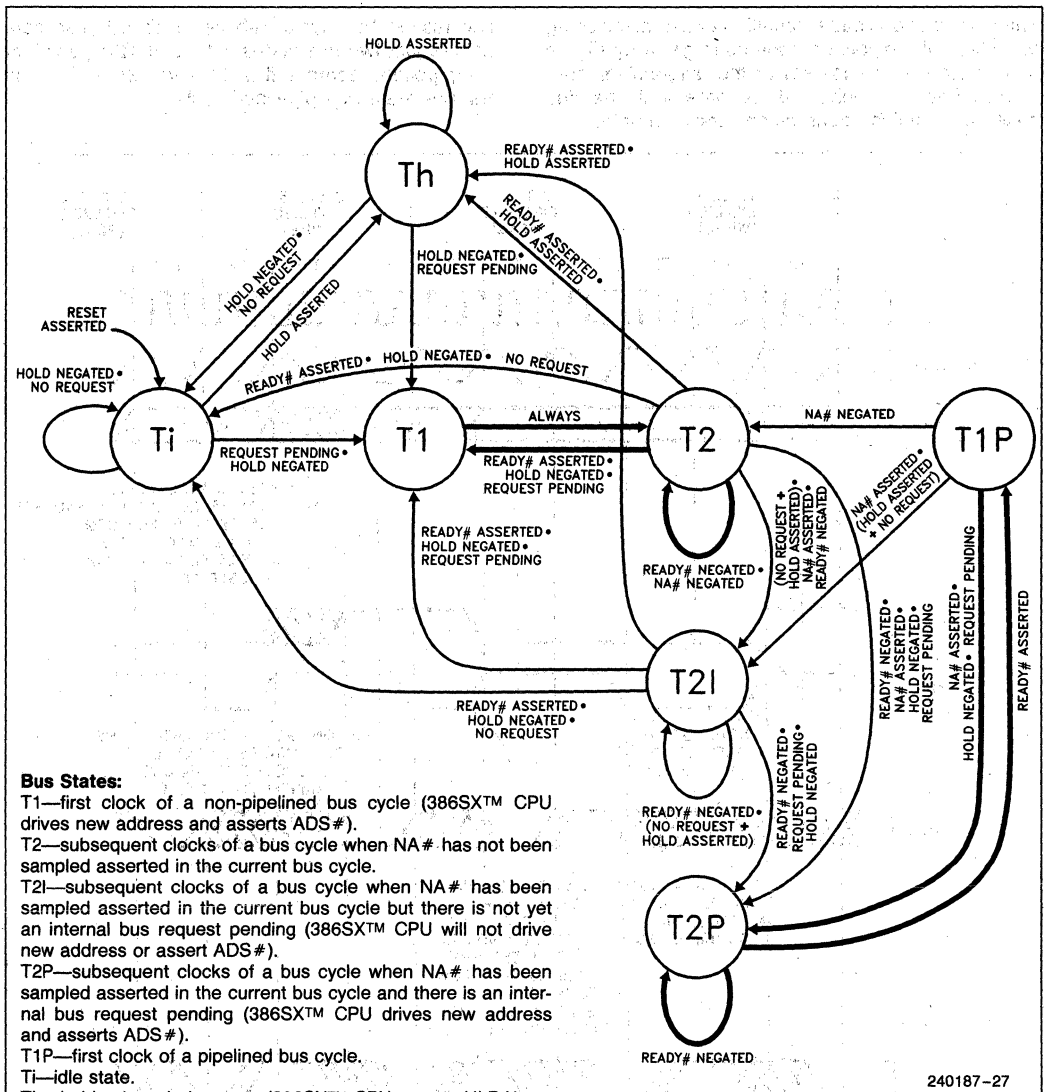


Figure 5.11. Details of Address Pipelining During Cycles with Wait States



Bus States:

T1—first clock of a non-pipelined bus cycle (386SX™ CPU drives new address and asserts ADS#).

T2—subsequent clocks of a bus cycle when NA# has not been sampled asserted in the current bus cycle.

T2I—subsequent clocks of a bus cycle when NA# has been sampled asserted in the current bus cycle but there is not yet an internal bus request pending (386SX™ CPU will not drive new address or assert ADS#).

T2P—subsequent clocks of a bus cycle when NA# has been sampled asserted in the current bus cycle and there is an internal bus request pending (386SX™ CPU drives new address and asserts ADS#).

T1P—first clock of a pipelined bus cycle.

Ti—idle state.

Th—hold acknowledge state (386SX™ CPU asserts HLDA).

Asserting NA# for pipelined address gives access to three more bus states: T2I, T2P and T1P.

Using pipelined address, the fastest bus cycle consists of T1P and T2P.

Figure 5.12. Complete Bus States (including pipelined address)

Initiating and Maintaining Pipelined Address

Using the state diagram Figure 5.12, observe the transitions from an idle state, T_i, to the beginning of a pipelined bus cycle T1P. From an idle state, T_i, the first bus cycle must begin with T1, and is therefore a non-pipelined bus cycle. The next bus cycle will be pipelined, however, provided NA# is asserted and the first bus cycle ends in a T2P state (the address for the next bus cycle is driven during T2P). The fastest path from an idle state to a bus cycle with pipelined address is shown in bold below:

T_I, T_I, T_I,	**T1 - T2 - T2P,**	**T1P - T2P,**
idle states	non-pipelined cycle	pipelined cycle

T1-T2-T2P are the states of the bus cycle that establish address pipelining for the next bus cycle, which begins with T1P. The same is true after a bus hold state, shown below:

T_h, T_h, T_h,	**T1 - T2 - T2P,**	**T1P - T2P,**
hold acknowledge states	non-pipelined cycle	pipelined cycle

The transition to pipelined address is shown functionally by Figure 5.10 Cycle 1. Note that Cycle 1 is used to transition into pipelined address timing for the subsequent Cycles 2, 3 and 4, which are pipelined. The NA# input is asserted at the appropriate time to select address pipelining for Cycles 2, 3 and 4.

Once a bus cycle is in progress and the current address has been valid for one entire bus state, the NA# input is sampled at the end of every phase one until the bus cycle is acknowledged. Sampling begins in T2 during Cycle 1 in Figure 5.10. Once NA# is sampled active during the current cycle, the 386SX Microprocessor is free to drive a new address and bus cycle definition on the bus as early as the next bus state. In Figure 5.10 Cycle 1 for example, the next address is driven during state T2P. Thus Cycle 1 makes the transition to pipelined address timing, since it begins with T1 but ends with T2P. Because the address for Cycle 2 is available before Cycle 2 begins, Cycle 2 is called a pipelined bus cycle, and it begins with T1P. Cycle 2 begins as soon as READY# asserted terminates Cycle 1.

Examples of transition bus cycles are Figure 5.10 Cycle 1 and Figure 5.9 Cycle 2. Figure 5.10 shows transition during the very first cycle after an idle bus state, which is the fastest possible transition into address pipelining. Figure 5.9 Cycle 2 shows a transition cycle occurring during a burst of bus cycles. In any case, a transition cycle is the same whenever it occurs: it consists at least of T1, T2 (NA# is asserted at that time), and T2P (provided the 386SX Microprocessor has an internal bus request already pending, which it almost always has). T2P states are repeated if wait states are added to the cycle.

Note that only three states (T1, T2 and T2P) are required in a bus cycle performing a **transition** from non-pipelined address into pipelined address timing, for example Figure 5.10 Cycle 1. Figure 5.10 Cycles 2, 3 and 4 show that address pipelining can be maintained with two-state bus cycles consisting only of T1P and T2P.

Once a pipelined bus cycle is in progress, pipelined timing is maintained for the next cycle by asserting NA# and detecting that the 386SX Microprocessor enters T2P during the current bus cycle. The current bus cycle must end in state T2P for pipelining to be maintained in the next cycle. T2P is identified by the assertion of ADS#. Figures 5.9 and 5.10 however, each show pipelining ending after Cycle 4 because Cycle 4 ends in T2I. This indicates the 386SX Microprocessor didn't have an internal bus request prior to the acknowledgement of Cycle 4. If a cycle ends with a T2 or T2I, the next cycle will not be pipelined.

Realistically, address pipelining is almost always maintained as long as NA# is sampled asserted. This is so because in the absence of any other request, a code prefetch request is always internally pending until the instruction decoder and code prefetch queue are completely full. Therefore, address pipelining is maintained for long bursts of bus cycles, if the bus is available (i.e., HOLD inactive) and NA# is sampled active in each of the bus cycles.

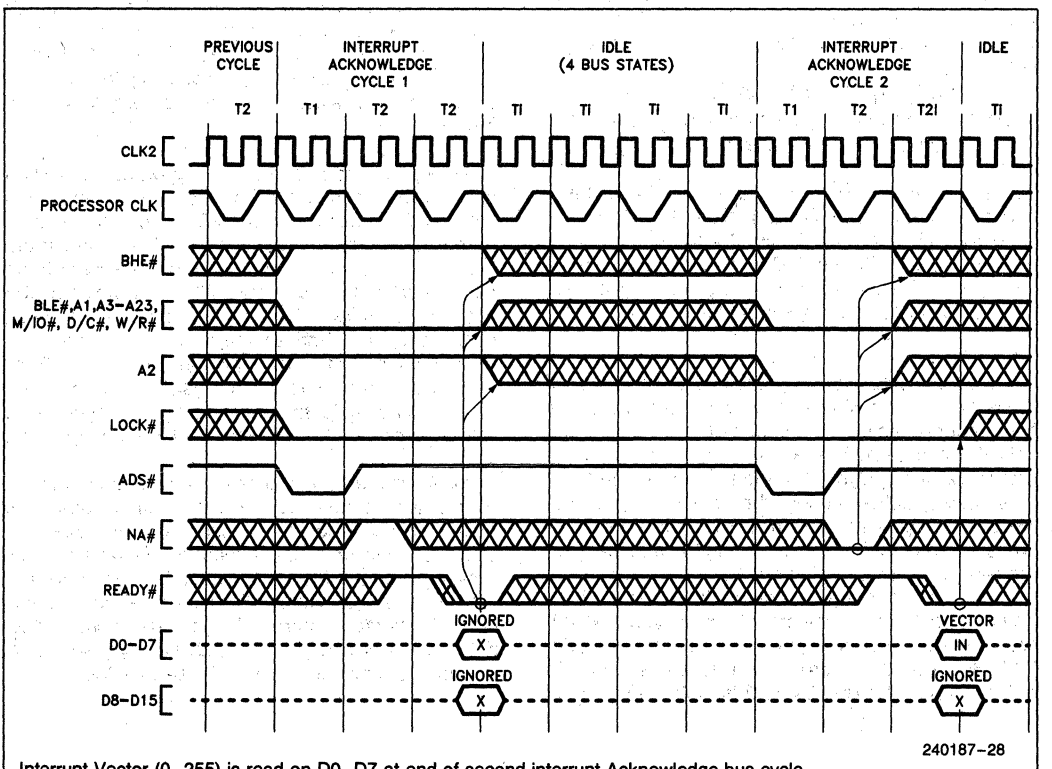
INTERRUPT ACKNOWLEDGE (INTA) CYCLES

In response to an interrupt request on the INTR input when interrupts are enabled, the 386SX Microprocessor performs two interrupt acknowledge cycles. These bus cycles are similar to read cycles in that bus definition signals define the type of bus activity taking place, and each cycle continues until acknowledged by READY# sampled active.

The state of A_2 distinguishes the first and second interrupt acknowledge cycles. The byte address driven during the first interrupt acknowledge cycle is 4 (A_{23}–A_3, A_1, BLE# LOW, A_2 and BHE# HIGH). The byte address driven during the second interrupt acknowledge cycle is 0 (A_{23}–A_1, BLE# LOW, and BHE# HIGH).

The LOCK# output is asserted from the beginning of the first interrupt acknowledge cycle until the end of the second interrupt acknowledge cycle. Four idle bus states, T_i, are inserted by the 386SX Microprocessor between the two interrupt acknowledge cycles for compatibility with spec TRHRL of the 8259A Interrupt Controller.

During both interrupt acknowledge cycles, D_{15}–D_0 float. No data is read at the end of the first interrupt acknowledge cycle. At the end of the second interrupt acknowledge cycle, the 386SX Microprocessor will read an external interrupt vector from D_7–D_0 of the data bus. The vector indicates the specific interrupt number (from 0–255) requiring service.



Interrupt Vector (0–255) is read on D0–D7 at end of second interrupt Acknowledge bus cycle.
Because each Interrupt Acknowledge bus cycle is followed by idle bus states. asserting NA# has no practical effect. Choose the approach which is simplest for your system hardware design.

Figure 5.13. Interrupt Acknowledge Cycles

HALT INDICATION CYCLE

The execution unit halts as a result of executing a HLT instruction. Signaling its entrance into the halt state, a halt indication cycle is performed. The halt indication cycle is identified by the state of the bus definition signals shown on page 40, **Bus Cycle Definition Signals**, and an address of 2. The halt indication cycle must be acknowledged by READY# asserted. A halted 386SX Microprocessor resumes execution when INTR (if interrupts are enabled), NMI or RESET is asserted.

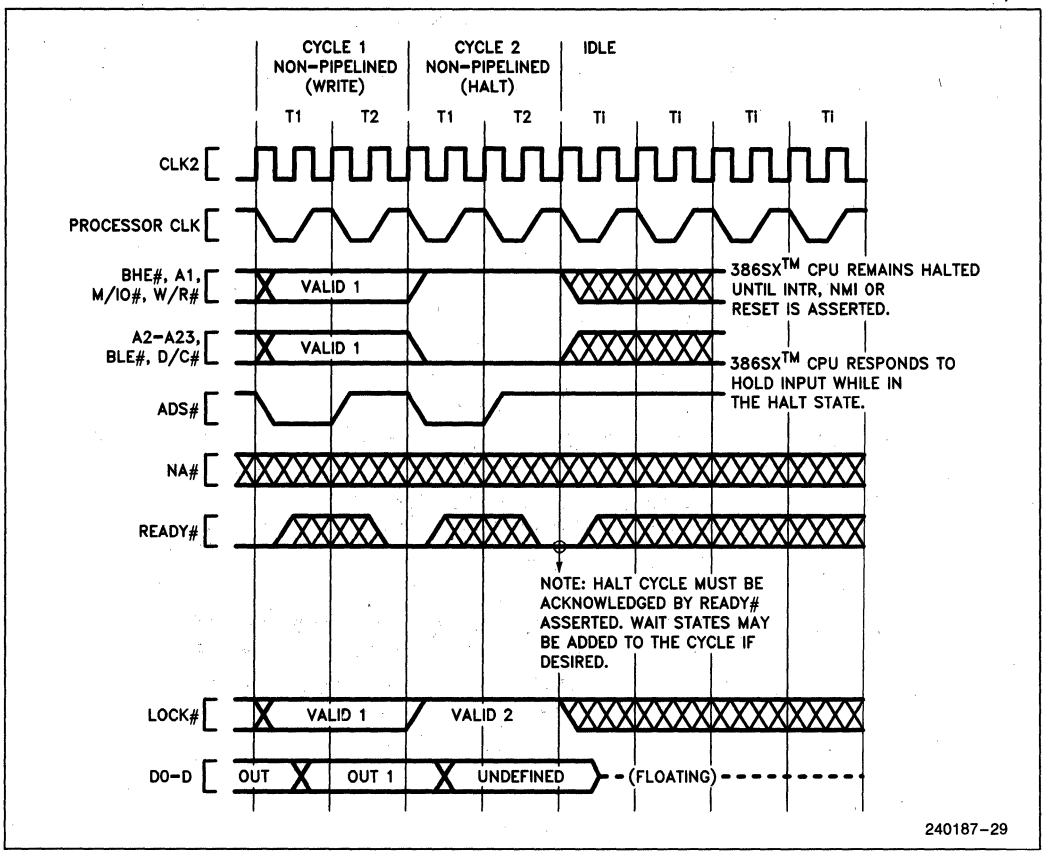


Figure 5.14. Example Halt Indication Cycle from Non-Pipelined Cycle

SHUTDOWN INDICATION CYCLE

The 386SX Microprocessor shuts down as a result of a protection fault while attempting to process a double fault. Signaling its entrance into the shutdown state, a shutdown indication cycle is performed. The shutdown indication cycle is identified by the state of the bus definition signals shown in **Bus Cycle Definition Signals** (page 40) and an address of 0. The shutdown indication cycle must be acknowledged by READY# asserted. A shutdown 386SX Microprocessor resumes execution when NMI or RESET is asserted.

ENTERING AND EXITING HOLD ACKNOWLEDGE

The bus hold acknowledge state, T_h, is entered in response to the HOLD input being asserted. In the bus hold acknowledge state, the 386SX Microprocessor floats all outputs or bidirectional signals, except for HLDA. HLDA is asserted as long as the 386SX Microprocessor remains in the bus hold acknowledge state. In the bus hold acknowledge state, all inputs except HOLD and RESET are ignored.

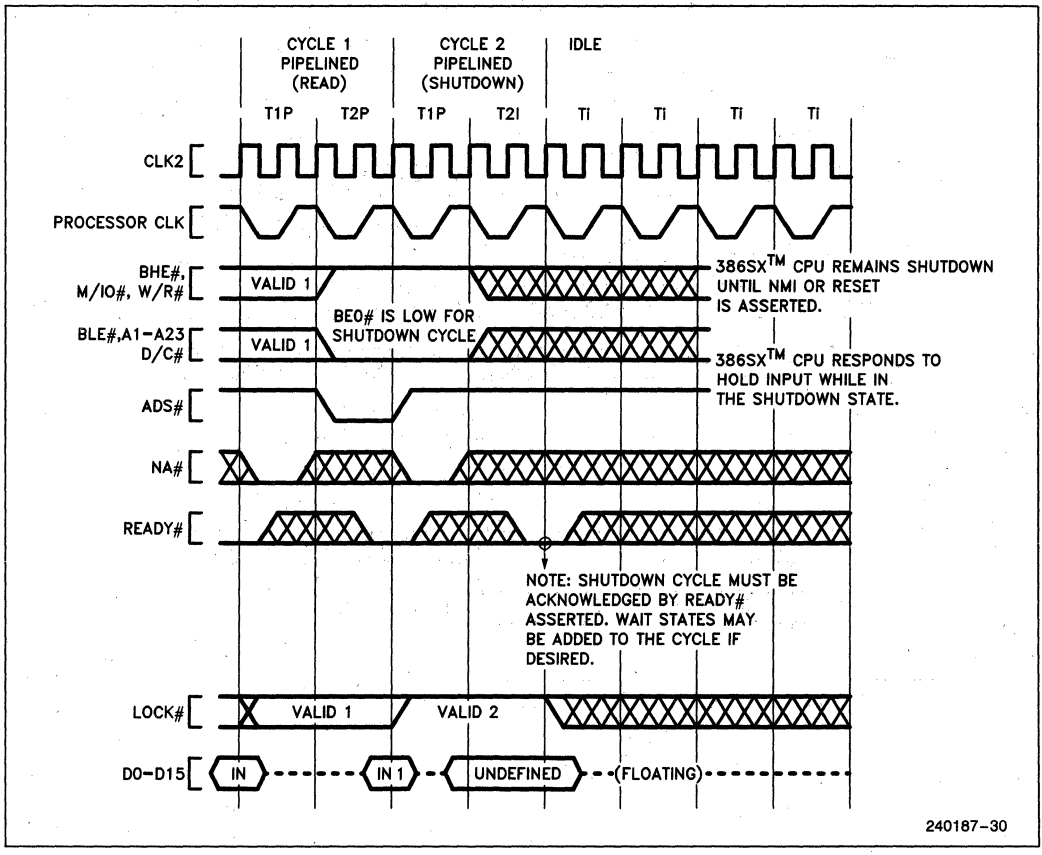


Figure 5.15. Example Shutdown Indication Cycle from Non-Pipelined Cycle

T_h may be entered from a bus idle state as in Figure 5.16 or after the acknowledgement of the current physical bus cycle if the LOCK# signal is not asserted, as in Figures 5.17 and 5.18.

T_h is exited in response to the HOLD input being negated. The following state will be T_i as in Figure 5.16 if no bus request is pending. The following bus state will be T1 if a bus request is internally pending, as in Figures 5.17 and 5.18. T_h is exited in response to RESET being asserted.

If a rising edge occurs on the edge-triggered NMI input while in T_h, the event is remembered as a nonmaskable interrupt 2 and is serviced when T_h is exited unless the 386SX Microprocessor is reset before T_h is exited.

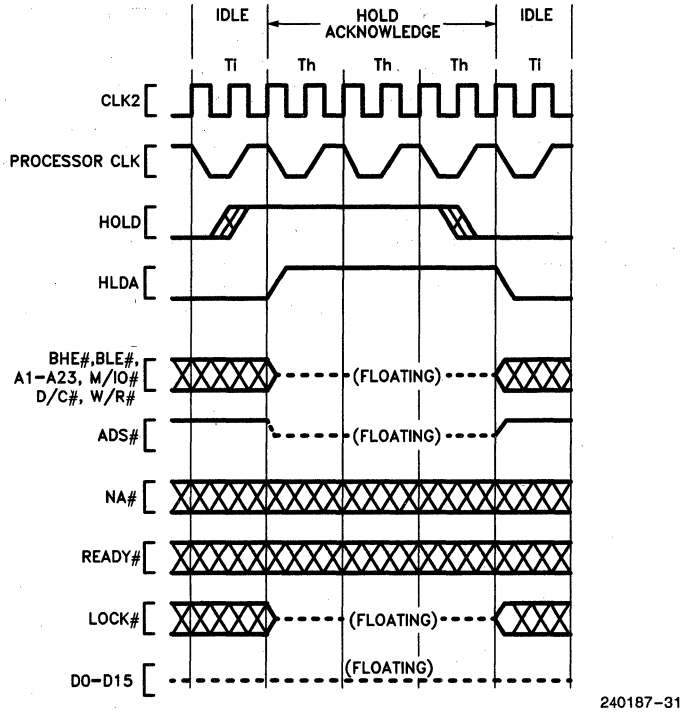
RESET DURING HOLD ACKNOWLEDGE

RESET being asserted takes priority over HOLD being asserted. If RESET is asserted while HOLD remains asserted, the 386SX Microprocessor drives its pins to defined states during reset, as in **Table 5.5 Pin State During Reset**, and performs internal reset activity as usual.

If HOLD remains asserted when RESET is inactive, the 386SX Microprocessor enters the hold acknowledge state before performing its first bus cycle, provided HOLD is still asserted when the 386SX Microprocessor would otherwise perform its first bus cycle.

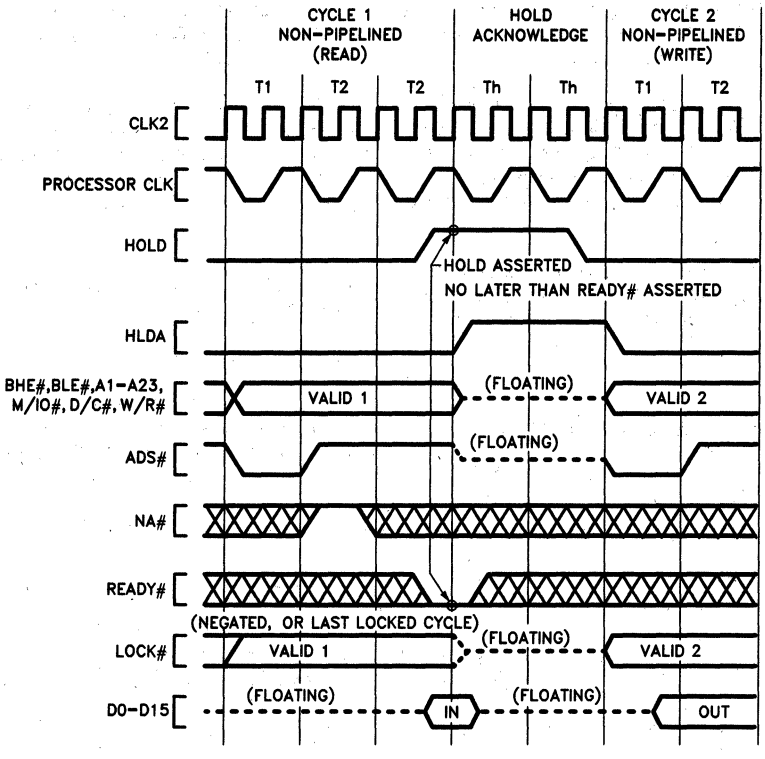
BUS ACTIVITY DURING AND FOLLOWING RESET

RESET is the highest priority input signal, capable of interrupting any processor activity when it is asserted. A bus cycle in progress can be aborted at any stage, or idle states or bus hold acknowledge states discontinued so that the reset state is established.



NOTE:
For maximum design flexibility the 386SX™ CPU has no internal pullup resistors on its outputs. Your design may require an external pullup on ADS# and other outputs to keep them negated during float periods.

Figure 5.16. Requesting Hold from Idle Bus



NOTE:
HOLD is a synchronous input and can be asserted at any CLK2 edge, provided setup and hold (t_{23} and t_{24}) requirements are met. This waveform is useful for determining Hold Acknowledge latency.

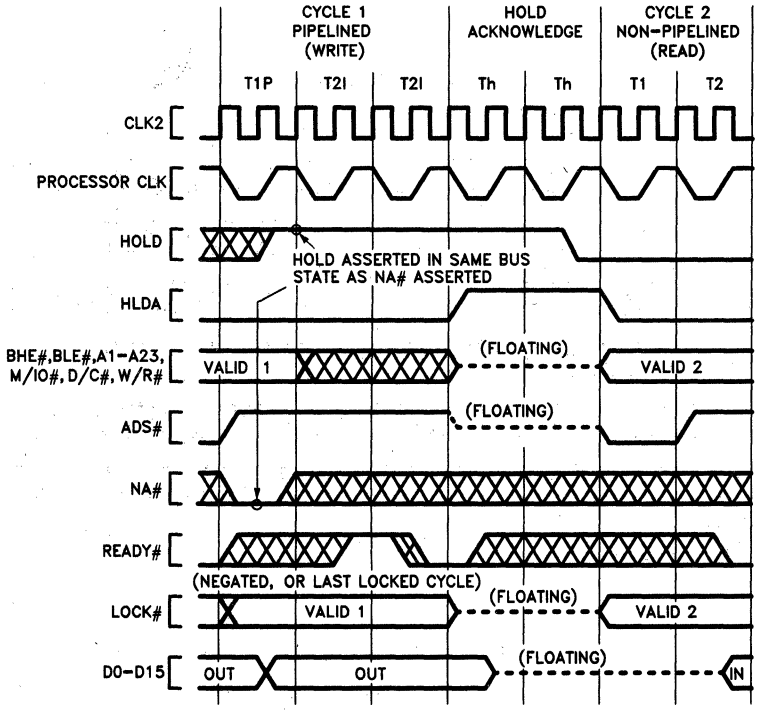
Figure 5.17. Requesting Hold from Active Bus (NA# inactive)

RESET should remain asserted for at least 15 CLK2 periods to ensure it is recognized throughout the 386SX Microprocessor, and at least 80 CLK2 periods if self-test is going to be requested at the falling edge. RESET asserted pulses less than 15 CLK2 periods may not be recognized. RESET pulses less than 80 CLK2 periods followed by a self-test may cause the self-test to report a failure when no true failure exists.

Provided the RESET falling edge meets setup and hold times t_{25} and t_{26}, the internal processor clock phase is defined at that time as illustrated by Figure 5.19 and Figure 7.7.

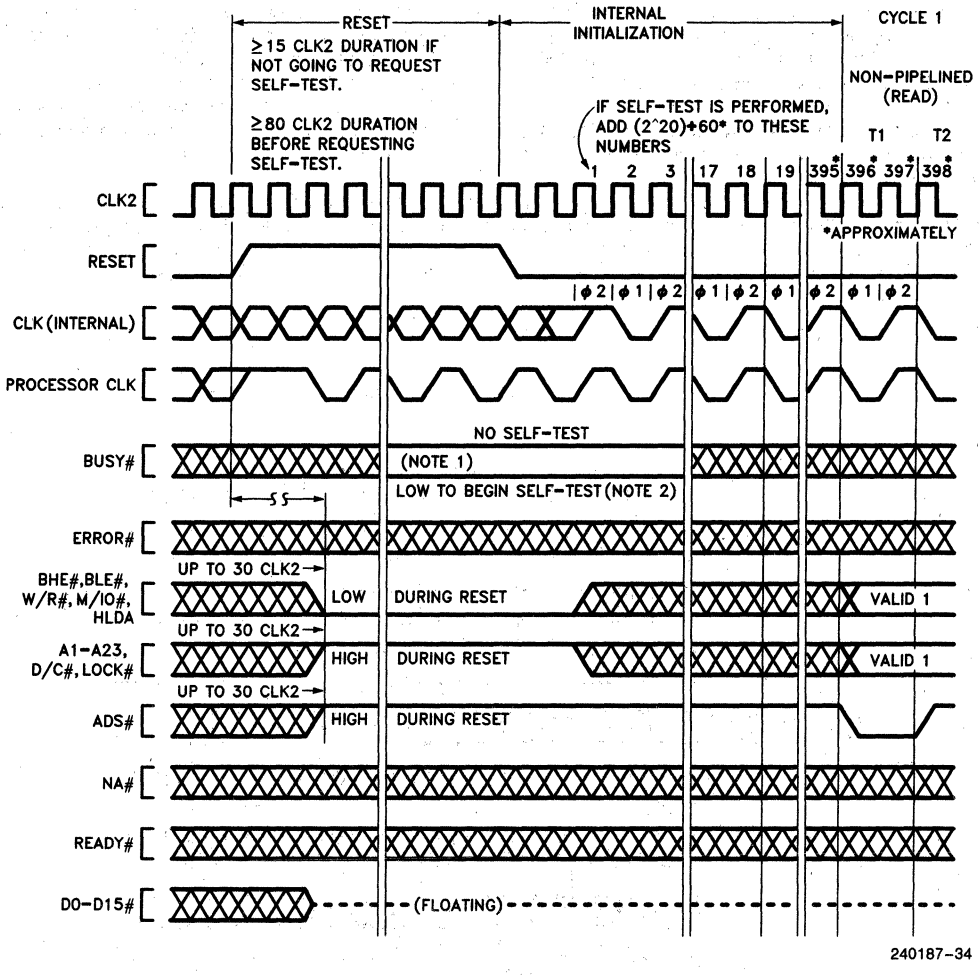
A self-test may be requested at the time RESET goes inactive by having the BUSY# input at a LOW level as shown in Figure 5.19. The self-test requires (2^{20} + approximately 60) CLK2 periods to complete. The self-test duration is not affected by the test results. Even if the self-test indicates a problem, the 386SX Microprocessor attempts to proceed with the reset sequence afterwards.

After the RESET falling edge (and after the self-test if it was requested) the 386SX Microprocessor performs an internal initialization sequence for approximately 350 to 450 CLK2 periods.



NOTE:
HOLD is a synchronous input and can be asserted at any CLK2 edge, provided setup and hold (t23 and t24) requirements are met. This waveform is useful for determining Hold Acknowledge latency.

Figure 5.18. Requesting Hold from Idle Bus (NA# active)



NOTES:
1. BUSY# should be held stable for 8 CLK2 periods before and after the CLK2 period in which RESET falling edge occurs.
2. If self-test is requested the outputs remain in their reset state as shown here.

Figure 5.19. Bus Activity from Reset Until First Code Fetch

5.5 Self-test Signature

Upon completion of self-test (if self-test was requested by driving BUSY# LOW at the falling edge of RESET) the EAX register will contain a signature of 00000000H indicating the 386SX Microprocessor passed its self-test of microcode and major PLA contents with no problems detected. The passing signature in EAX, 00000000H, applies to all revision levels. Any non-zero signature indicates the unit is faulty.

5.6 Component and Revision Identifiers

To assist users, the 386SX Microprocessor after reset holds a component identifier and revision identifier in its DX register. The upper 8 bits of DX hold 23H as identification of the 386SX Microprocessor (the lower nibble, 03H, refers to the Intel386 Architecture. The upper nibble, 02H, refers to the second member of the Intel386 Family). The lower 8 bits of DX hold an 8-bit unsigned binary number related to the component revision level. The revision identifier will, in general, chronologically track those component steppings which are intended to have certain improvements or distinction from previous steppings. The 386SX Microprocessor revision identifier will track that of the 386 CPU where possible.

The revision identifier is intended to assist users to a practical extent. However, the revision identifier value is not guaranteed to change with every stepping revision, or to follow a completely uniform numerical sequence, depending on the type or intention of revision, or manufacturing materials required to be changed. Intel has sole discretion over these characteristics of the component.

Table 5.7. Component and Revision Identifier History

Stepping	Revision Identifier
A0	04H
B	05H

5.7 Coprocessor Interfacing

The 386SX Microprocessor provides an automatic interface for the Intel 80387SX numeric floating-point coprocessor. The 80387SX coprocessor uses an I/O mapped interface driven automatically by the 386SX Microprocessor and assisted by three dedicated signals: BUSY#, ERROR# and PEREQ.

As the 386SX Microprocessor begins supporting a coprocessor instruction, it tests the BUSY# and ERROR# signals to determine if the coprocessor can accept its next instruction. Thus, the BUSY# and ERROR# inputs eliminate the need for any 'preamble' bus cycles for communication between processor and coprocessor. The 80387SX can be given its command opcode immediately. The dedicated signals provide instruction synchronization, and eliminate the need of using the WAIT opcode (9BH) for 80387SX instruction synchronization (the WAIT opcode was required when the 8086 or 8088 was used with the 8087 coprocessor).

Custom coprocessors can be included in 386SX Microprocessor based systems by memory-mapped or I/O-mapped interfaces. Such coprocessor interfaces allow a completely custom protocol, and are not limited to a set of coprocessor protocol 'primitives'. Instead, memory-mapped or I/O-mapped interfaces may use all applicable instructions for high-speed coprocessor communication. The BUSY# and ERROR# inputs of the 386SX Microprocessor may also be used for the custom coprocessor interface, if such hardware assist is desired. These signals can be tested by the WAIT opcode (9BH). The WAIT instruction will wait until the BUSY# input is inactive (interruptable by an NMI or enabled INTR input), but generates an exception 16 fault if the ERROR# pin is active when the BUSY# goes (or is) inactive. If the custom coprocessor interface is memory-mapped, protection of the addresses used for the interface can be provided with the 386SX CPU's on-chip paging or segmentation mechanisms. If the custom interface is I/O-mapped, protection of the interface can be provided with the IOPL (I/O Privilege Level) mechanism.

The 80387SX numeric coprocessor interface is I/O mapped as shown in Table 5.8. Note that the 80387SX coprocessor interface addresses are beyond the 0H-0FFFFH range for programmed I/O. When the 386SX Microprocessor supports the 80387SX coprocessor, the 386SX Microprocessor automatically generates bus cycles to the coprocessor interface addresses.

Table 5.8. Numeric Coprocessor Port Addresses

Address in 386SX™ CPU I/O Space	80387SX Coprocessor Register
8000F8H	Opcode Register
8000FCH/8000FEH*	Operand Register

*Generated as 2nd bus cycle during Dword transfer.

To correctly map the 80387SX registers to the appropriate I/O addresses, connect the CMD0 and CMD1 lines of the 80387SX as listed in Table 5.9.

Table 5.9. Connections for CMD0 and CMD1 Inputs for the 80387SX

Signal	Connection
CMD0	Connect to latched version of 386SX™ CPU A2 signal
CMD1	Connect to ground.

Software Testing for Coprocessor Presence

When software is used to test for coprocessor (80387SX) presence, it should use only the following coprocessor opcodes: FINIT, FNINIT, FSTCW mem, FSTSW mem and FSTSW AX. To use other coprocessor opcodes when a coprocessor is known to be not present, first set EM = 1 in the 386SX CPU's CR0 register.

6.0 PACKAGE THERMAL SPECIFICATIONS

The 386SX Microprocessor is specified for operation when case temperature is within the range of 0°C–85°C. The case temperature may be measured in any environment, to determine whether the 386SX Microprocessor is within specified operating range. The case temperature should be measured at the center of the top surface opposite the pins.

The ambient temperature is guaranteed as long as T_c is not violated. The ambient temperature can be calculated from the θ_{jc} and θ_{ja} from the following equations:

$$T_j = T_c + P^*\theta_{jc}$$

$$T_a = T_j - P^*\theta_{ja}$$

$$T_c = T_a + P^*[\theta_{ja} - \theta_{jc}]$$

Values for θ_{ja} and θ_{jc} are given in table 6.1 for the 100 lead fine pitch. θ_{ja} is given at various airflows. Table 6.2 shows the maximum T_a allowable (without exceeding T_c) at various airflows. Note that T_a can be improved further by attaching 'fins' or a 'heat sink' to the package.

7.0 ELECTRICAL SPECIFICATIONS

The following sections describe recommended electrical connections for the 386SX Microprocessor, and its electrical specifications.

7.1 Power and Grounding

The 386SX Microprocessor is implemented in CHMOS III technology and has modest power requirements. However, its high clock frequency and 47 output buffers (address, data, control, and HLDA) can cause power surges as multiple output buffers drive new signal levels simultaneously. For clean on-chip power distribution at high frequency, 14 Vcc and 18 Vss pins separately feed functional units of the 386SX Microprocessor.

Power and ground connections must be made to all external Vcc and GND pins of the 386SX Microprocessor. On the circuit board, all Vcc pins should be connected on a Vcc plane and all Vss pins should be connected on a GND plane.

POWER DECOUPLING RECOMMENDATIONS

Liberal decoupling capacitors should be placed near the 386SX Microprocessor. The 386SX Microprocessor driving its 24-bit address bus and 16-bit data bus at high frequencies can cause transient power surges, particularly when driving large capacitive loads. Low inductance capacitors and interconnects are recommended for best high frequency electrical performance. Inductance can be reduced by shortening circuit board traces between the 386SX Microprocessor and decoupling capacitors as much as possible.

Table 6.1. Thermal Resistances (°C/Watt) θ_{jc} and θ_{ja}.

Package	θjc	θ_{ja} versus Airflow - ft/min (m/sec)					
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
100 Lead Fine Pitch	7	33	27	24	21	18	17

Table 6.2: Maximum T_a at various airflows.

Package	T_A(°C) versus Airflow - ft/min (m/sec)					
	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
100 Lead Fine Pitch	33	45	51	57	63	65

Max. T_A calculated at 5.0V and max I_{CC}.

Table 7.1. Recommended Resistor Pull-ups to Vcc

Pin	Signal	Pull-up Value	Purpose
16	ADS#	20 K-Ohm ± 10%	Lightly pull ADS# inactive during 386SX™ CPU hold acknowledge states
26	LOCK#	20 K-Ohm ± 10%	Lightly pull LOCK# inactive during 386SX™ CPU hold acknowledge states

RESISTOR RECOMMENDATIONS

The ERROR# and BUSY# inputs have internal pull-up resistors of approximately 20 K-Ohms and the PEREQ input has an internal pull-down resistor of approximately 20 K-Ohms built into the 386SX Microprocessor to keep these signals inactive when the 80387SX is not present in the system (or temporarily removed from its socket).

In typical designs, the external pull-up resistors shown in Table 7.1 are recommended. However, a particular design may have reason to adjust the resistor values recommended here, or alter the use of pull-up resistors in other ways.

OTHER CONNECTION RECOMMENDATIONS

For reliable operation, always connect unused inputs to an appropriate signal level. N/C pins should always remain **unconnected. Connection of N/C pins to Vcc or Vss will result in component malfunction or incompatibility with future steppings of the 386SX Microprocessor**.

Particularly when not using interrupts or bus hold (as when first prototyping), prevent any chance of spurious activity by connecting these associated inputs to GND:

Pin	Signal
40	INTR
38	NMI
4	HOLD

If not using address pipelining, connect pin 6, NA#, through a pull-up in the range of 20 K-Ohms to Vcc.

7.2 Maximum Ratings

Table 7.2. Maximum Ratings

Parameter	Maximum Rating
Storage temperature	−65 °C to 150 °C
Case temperature under bias	−65 °C to 110 °C
Supply voltage with respect to Vss	−.5V to 6.5V
Voltage on other pins	−.5V to (Vcc+.5)V

Table 7.2 gives stress ratings only, and functional operation at the maximums is not guaranteed. Functional operating conditions are given in section **7.3, D.C. Specifications**, and section **7.4, A.C. Specifications**.

Extended exposure to the Maximum Ratings may affect device reliability. Furthermore, although the 386SX Microprocessor contains protective circuitry to resist damage from static electric discharge, always take precautions to avoid high static voltages or electric fields.

7.3 D.C. Specifications

Functional operating range: Vcc = 5V ± 10%; T_{CASE} = 0°C to 85°C

Table 7.3. D.C. Characteristics

Symbol	Parameter	Min	Max	Unit	Notes
V_{IL}	Input LOW Voltage	−0.3	+0.8	V	
V_{IH}	Input HIGH Voltage	2.0	Vcc+0.3	V	
V_{ILC}	CLK2 Input LOW Voltage	−0.3	+0.8	V	
V_{IHC}	CLK2 Input HIGH Voltage	Vcc−0.8	Vcc+0.3	V	
V_{OL}	Output LOW Voltage I_{OL} = 4mA: A_{23}–A_1, D_{15}–D_0 I_{OL} = 5mA: BHE#, BLE#, W/R#, D/C#, M/IO#, LOCK#, ADS#, HLDA		0.45 0.45	V V	
V_{OH}	Output high voltage I_{OH} = −1mA: A_{23}–A_1, D_{15}–D_0 I_{OH} = −0.2 mA: A_{23}–A_1, D_{15}–D_0 I_{OH} = −0.9mA: BHE#, BLE#, W/R#, D/C#, M/IO#, LOCK#, ADS#, HLDA I_{OH} = −0.18 mA: BHE#, BLE#, W/R#, D/C#, M/IO#, LOCK#, ADS#, HLDA	2.4 Vcc−0.5 2.4 Vcc−0.5		V V V	
I_{LI}	Input leakage current (for all pins except PEREQ, BUSY# and ERROR#)		±15	μA	0V ≤ V_{IN} ≤ Vcc
I_{IH}	Input Leakage Current (PEREQ pin)		200	μA	V_{IH} = 2.4V, Note 1
I_{IL}	Input Leakage Current (BUSY# and ERROR# pins)		−400	μA	V_{IL} = 0.45V, Note 2
I_{LO}	Output leakage current		±15	μA	0.45V ≤ V_{OUT} ≤ Vcc
Icc	Supply current (CLK2 = 32 MHz)		400	mA	Icc typ = 300mA, Note 3
C_{IN}	Input capacitance		10	pF	Fc = 1 MHz, Note 4
C_{OUT}	Output or I/O capacitance		12	pF	Fc = 1 MHz, Note 4
C_{CLK}	CLK2 Capacitance		20	pF	Fc = 1 MHz, Note 4

Tested at the minimum operating frequency of the part.

NOTES:
1. PEREQ input has an internal pull-down resistor.
2. BUSY# and ERROR# inputs each have an internal pull-up resistor.
3. Icc max measurement at worst case load, frequency, Vcc and temperature.
4. Not 100% tested.

7.4 A.C. Specifications

The A.C. specifications given in Table 7.4 consist of output delays, input setup requirements and input hold requirements. All A.C. specifications are relative to the CLK2 rising edge crossing the 2.0V level.

A.C. spec measurement is defined by Figure 7.1. Inputs must be driven to the voltage levels indicated by Figure 7.1 when A.C. specifications are measured. Output delays are specified with minimum and maximum limits measured as shown. The minimum delay times are hold times provided to external circuitry. Input setup and hold times are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct operation.

Outputs NA#, W/R#, D/C#, M/IO#, LOCK#, BHE#, BLE#, A_{23}–A_1 and HLDA only change at the beginning of phase one. D_{15}–D_0 (write cycles) only change at the beginning of phase two. The READY#, HOLD, BUSY#, ERROR#, PEREQ and D_{15}–D_0 (read cycles) inputs are sampled at the beginning of phase one. The NA#, INTR and NMI inputs are sampled at the beginning of phase two.

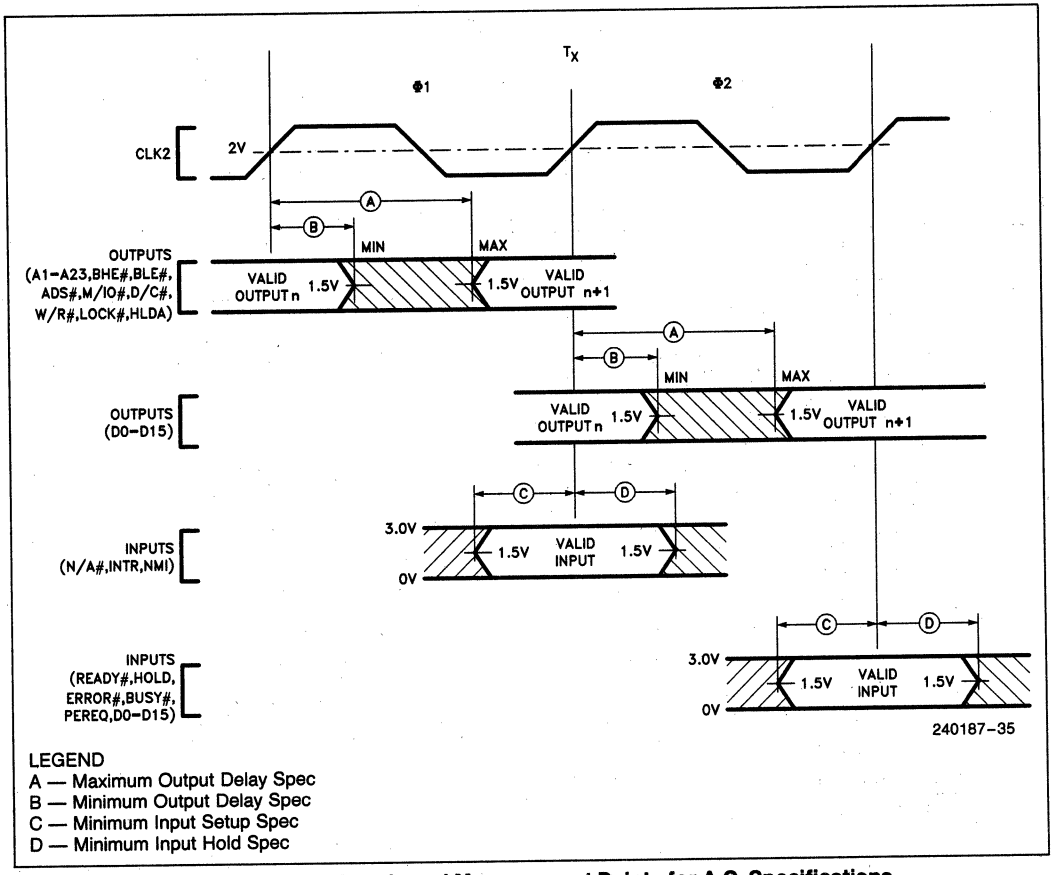


Figure 7.1. Drive Levels and Measurement Points for A.C. Specifications

A.C. SPECIFICATONS TABLES

Functional operating range: Vcc = 5V ± 10%; T_{CASE} = 0°C to 85°C

Table 7.4. A.C. Characteristics at 16 MHz

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Operating frequency	4	16	MHz		Half CLK2 Freq
t_1	CLK2 period	31	125	ns	7.3	
t_{2a}	CLK2 HIGH time	9		ns	7.3	at 2V(3)
t_{2b}	CLK2 HIGH time	5		ns	7.3	at (Vcc−0.8)V(3)
t_{3a}	CLK2 LOW time	9		ns	7.3	at 2V(3)
t_{3b}	CLK2 LOW time	7		ns	7.3	at 0.8V(3)
t_4	CLK2 fall time		8	ns	7.3	(Vcc−0.8)V to 0.8V(3)
t_5	CLK2 rise time		8	ns	7.3	0.8V to (Vcc−0.8)V(3)
t_6	A_{23}–A_1 valid delay	4	36	ns	7.5	C_L = 120pF(4)
t_7	A_{23}–A_1 float delay	4	40	ns	7.6	(Note 1)
t_8	BHE#, BLE#, LOCK# valid delay	4	36	ns	7.5	C_L = 75pF(4)
t_9	BHE#, BLE#, LOCK# float delay	4	40	ns	7.6	(Note 1)
t_{10}	W/R#, M/IO#, D/C#, ADS# valid delay	6	33	ns	7.5	C_L = 75pF(4)
t_{11}	W/R#, M/IO#, D/C#, ADS# float delay	6	35	ns	7.6	(Note 1)
t_{12}	D_{15}–D_0 Write Data Valid Delay	4	40	ns	7.5	C_L = 120pF(4)
t_{13}	D_{15}–D_0 Write Data Float Delay	4	35	ns	7.6	(Note 1)
t_{14}	HLDA valid delay	6	33	ns	7.5	C_L = 75pF(4)
t_{15}	NA# setup time	5		ns	7.4	
t_{16}	NA# hold time	21		ns	7.4	
t_{19}	READY# setup time	19		ns	7.4	
t_{20}	READY# hold time	4		ns	7.4	
t_{21}	D_{15}–D_0 Read Data setup time	9		ns	7.4	
t_{22}	D_{15}–D_0 Read Data hold time	6		ns	7.4	
t_{23}	HOLD setup time	26		ns	7.4	
t_{24}	HOLD hold time	5		ns	7.4	
t_{25}	RESET setup time	13		ns	7.7	
t_{26}	RESET hold time	4		ns	7.7	

Functional operating range: Vcc = 5V ± 10%; T_{CASE} = 0°C to 85°C

Table 7.4. A.C. Characteristics at 16 MHz (Continued)

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{27}	NMI, INTR setup time	16		ns	7.4	(Note 2)
t_{28}	NMI, INTR hold time	16		ns	7.4	(Note 2)
t_{29}	PEREQ, ERROR#, BUSY# setup time	16		ns	7.4	(Note 2)
t_{30}	PEREQ, ERROR#, BUSY# hold time	5		ns	7.4	(Note 2)

NOTES:
1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2: These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3: These are not tested. They are guaranteed by design characterization.
4: Tested with C_L set at 50 pf and derated to support the indicated distributed capacitive load. See figure 7.8 for the capacitive derating curve.

A.C. TEST LOADS

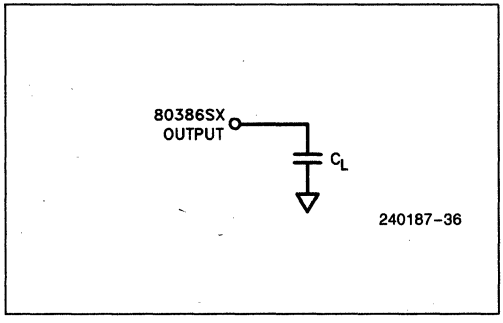


Figure 7.2. A.C. Test Loads

A.C. TIMING WAVEFORMS

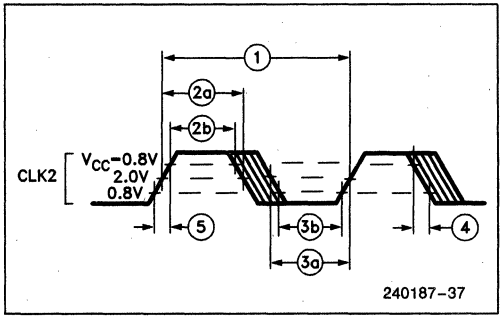


Figure 7.3. CLK2 Waveform

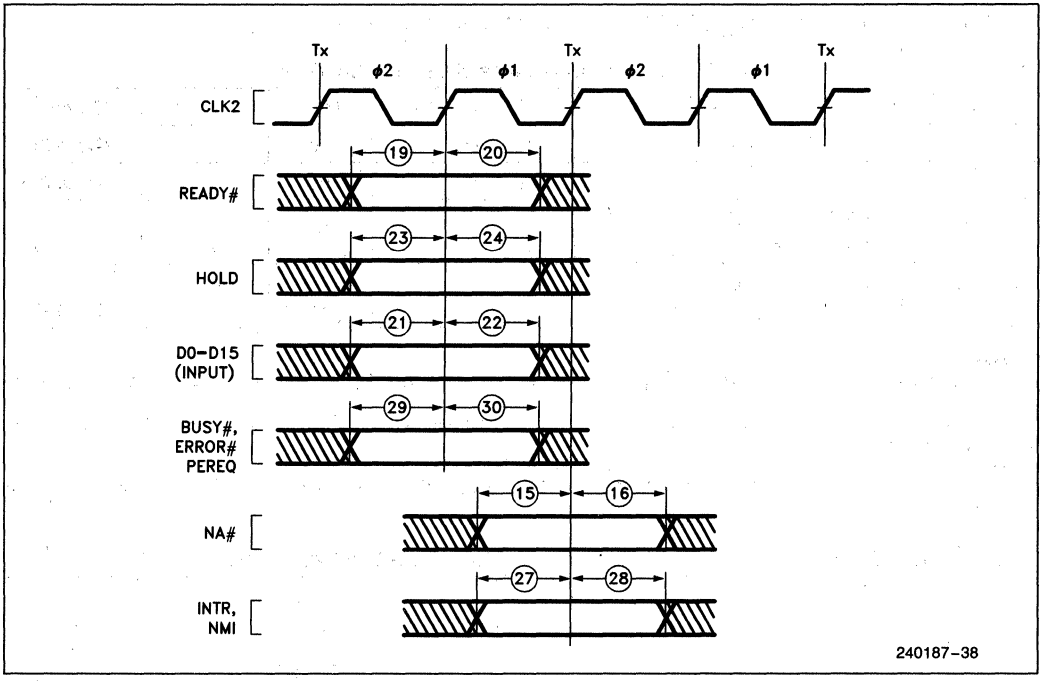


Figure 7.4. A.C. Timing Waveforms—Input Setup and Hold Timing

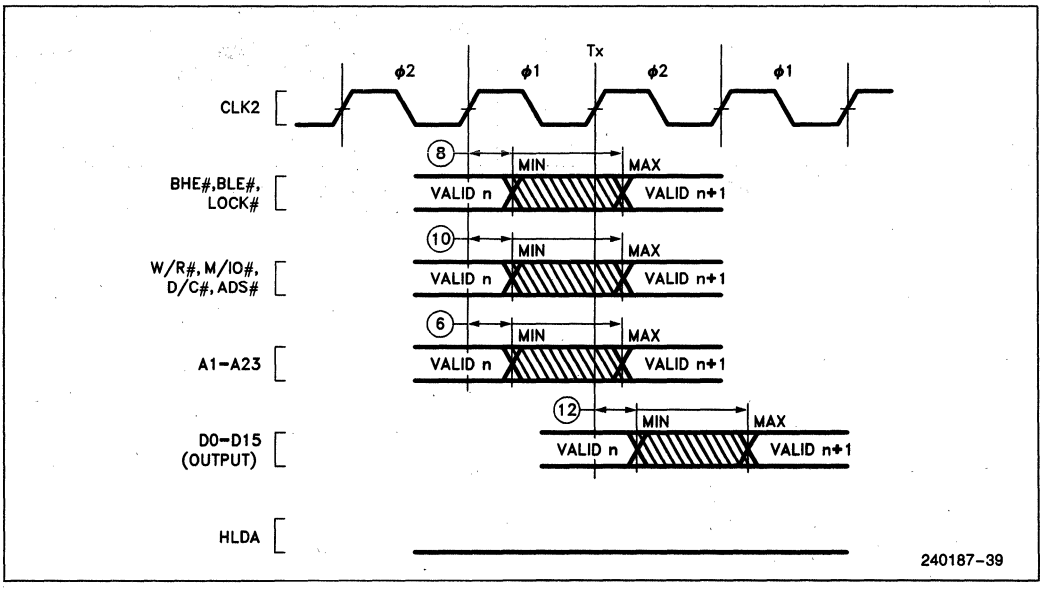


Figure 7.5. A.C. Timing Waveforms—Output Valid Delay Timing

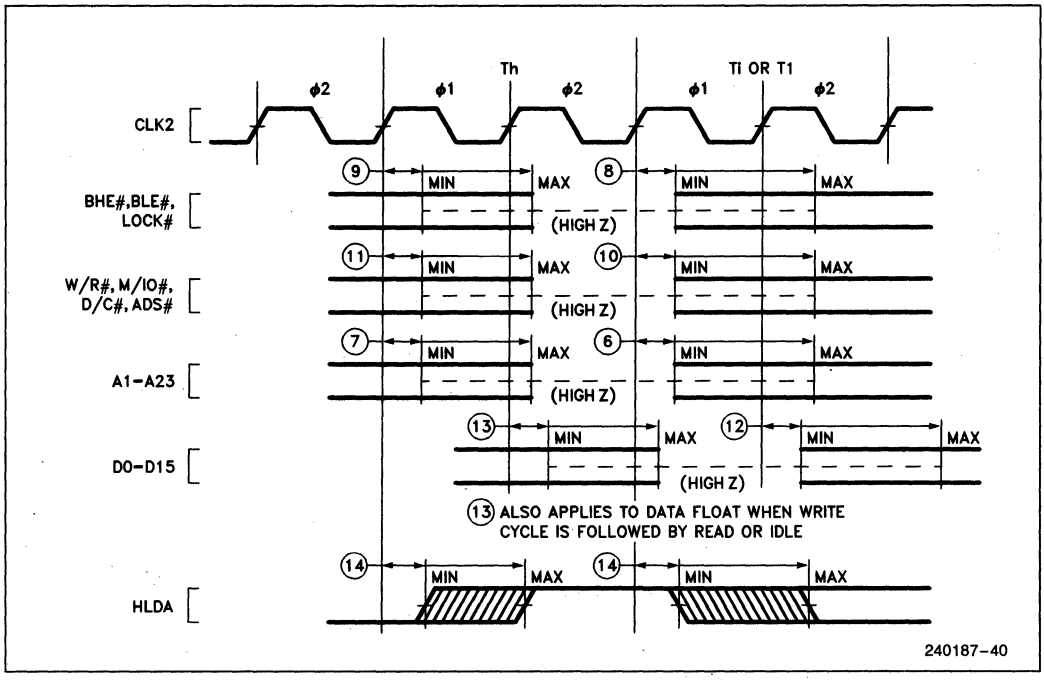


Figure 7.6. A.C. Timing Waveforms—Output Float Delay and HLDA Valid Delay Timing

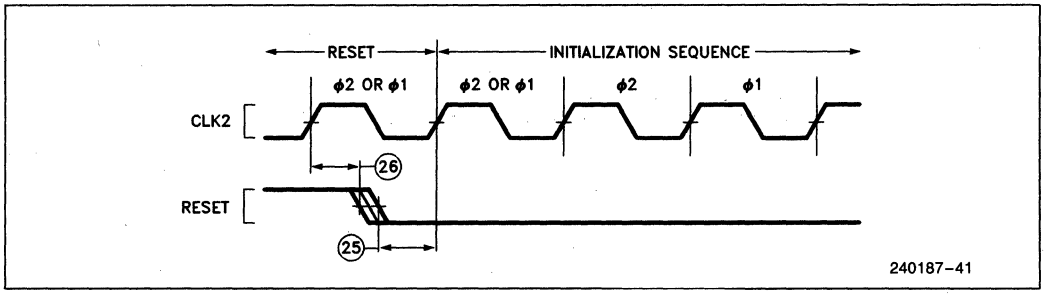


Figure 7.7. A.C. Timing Waveforms—RESET Setup and Hold Timing and Internal Phase

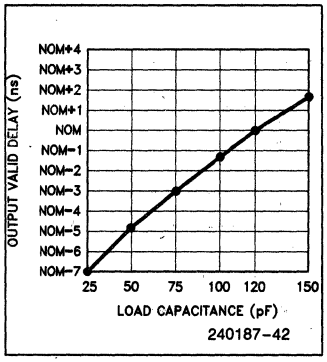


Figure 7.8. Capacitive Derating Curve

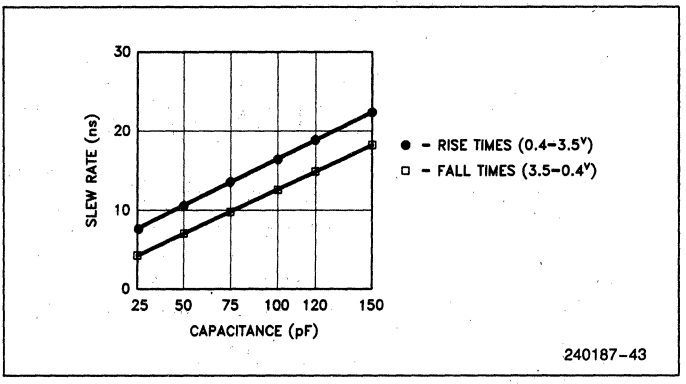


Figure 7.9. CMOS Level Slew Rates for Output Buffers

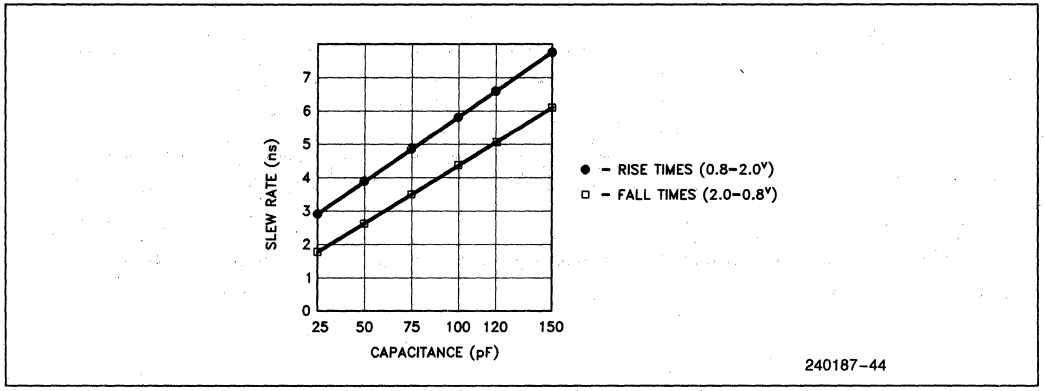


Figure 7.10. TTL Level Slew Rates for Output Buffers

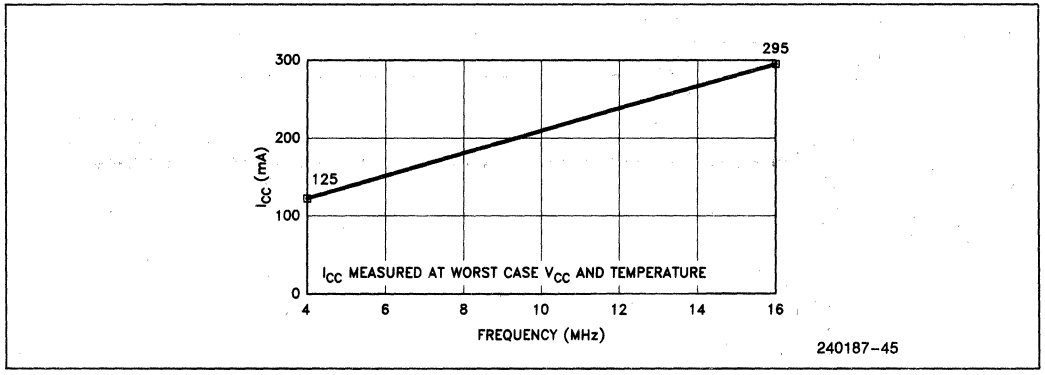


Figure 7.11. Typical Icc vs Frequency

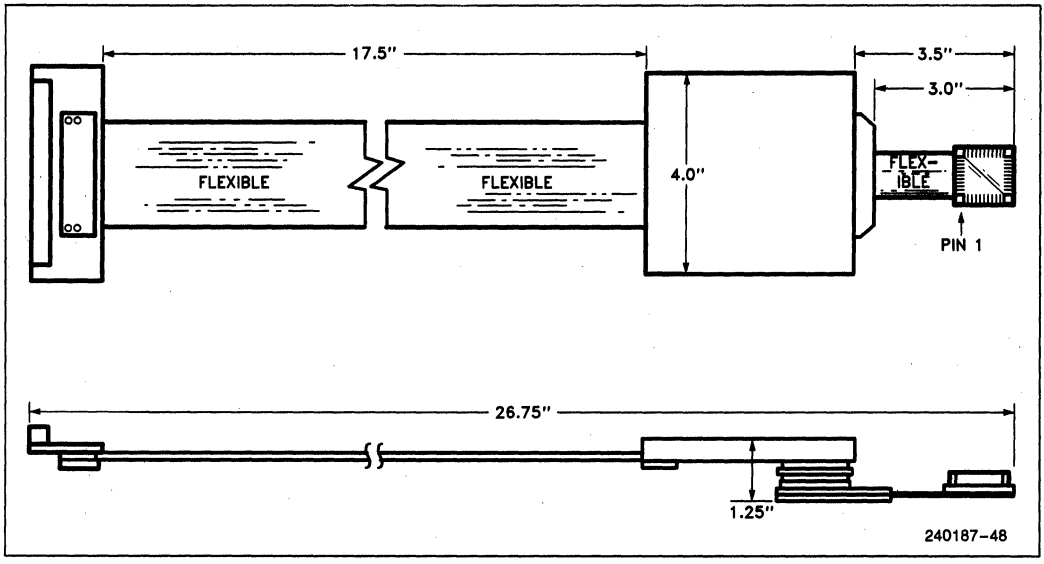


Figure 7.12. Preliminary ICE™-386SX Emulator User Cable with PQFP Adapter

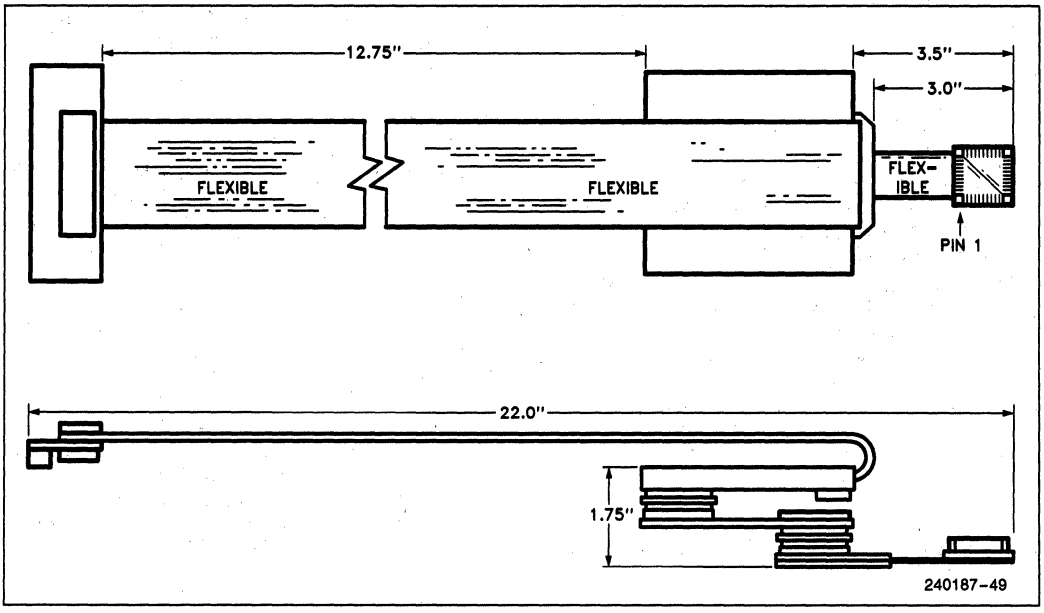


Figure 7.13. Preliminary ICE™-386SX Emulator User Cable with OIB and PQFP Adapter

7.5 Designing for ICE™-386SX Emulator (Advanced Data)

The 386SX CPU's in-circuit emulator product is the ICE™-386SX emulator. Use of the emulator requires the target system to provide a socket that is compatible with the ICE-386SX emulator. The ICE-386SX offers a 100-pin fine pitch flat-pack probe for emulating user systems. The 100-pin fine pitch flat-pack probe requires a socket, called the 100-pin PQFP, which is available from 3M text-tool (part number 2-0100-07243-000). The ICE-386SX emulator probe attaches to the target system via an adapter which replaces the 386SX CPU component in the target system. Because of the high operating frequency of 386SX CPU systems and of the ICE-386SX emulator, there is no buffering between the 386SX CPU emulation processor in the ICE-386SX emulator probe and the target system. A direct result of the non-buffered interconnect is that the ICE-386SX emulator shares the address and data bus with the user's system, and the RESET signal is intercepted by the ICE emulator hardware. In order for the ICE-386SX emulator to be functional in the user's system without the Optional Isolation Board (OIB) the designer must be aware of the following conditions:

1. The bus controller must only enable data transceivers onto the data bus during valid read cycles of the 386SX CPU, other local devices or other bus masters.
2. Before another bus master drives the local processor address bus, the other master must gain control of the address bus by asserting HOLD and receiving the HLDA response.
3. The emulation processor receives the RESET signal 2 or 4 CLK2 cycles later than an 386SX CPU would, and responds to RESET later. Correct phase of the response is guaranteed.

In addition to the above considerations, the ICE-386SX emulator processor module has several electrical and mechanical characteristics that should be taken into consideration when designing the 386SX CPU system.

Capacitive Loading: ICE-386SX adds up to 27 pF to each 386SX CPU signal.

Drive Requirements: ICE-386SX adds one FAST TTL load on the CLK2, control, address, and data lines. These loads are within the processor module and are driven by the 386SX CPU emulation processor, which has standard drive and loading capability listed in Tables 7.3 and 7.4.

Power Requirements: For noise immunity and CMOS latch-up protection the ICE-386SX emulator processor module is powered by the user system. The circuitry on the processor module draws up to 1.4A including the maximum 386SX CPU I_{CC} from the user 386SX CPU socket.

386SX CPU Location and Orientation: The ICE-386SX emulator processor module may require lateral clearance. Figure 7.12 shows the clearance requirements of the iMP adapter. The optional isolation board (OIB), which provides extra electrical buffering and has the same lateral clearance requirements as Figure 7.12, adds an additional 0.5 inches to the vertical clearance requirement. This is illustrated in Figure 7.13.

Optional Isolation Board (OIB) and the CLK2 speed reduction: Due to the unbuffered probe design, the ICE-386SX emulator is susceptible to errors on the user's bus. The OIB allows the ICE-386SX emulator to function in user systems with faults (shorted signals, etc.). After electrical verification the OIB may be removed. When the OIB is installed, the user system must have a maximum CLK2 frequency of 20 MHz.

8.0 DIFFERENCES BETWEEN THE 386SX™ CPU AND THE 386™ CPU

The following are the major differences between the 386SX CPU and the 386 CPU:

1. The 386SX CPU generates byte selects on BHE# and BLE# (like the 8086 and 80286) to distinguish the upper and lower bytes on its 16-bit data bus. The 386 CPU uses four byte selects, BE0#-BE3#, to distinguish between the different bytes on its 32-bit bus.
2. The 386SX CPU has no bus sizing option. The 386 CPU can select between either a 32-bit bus or a 16-bit bus by use of the BS16# input. The 386SX CPU has a 16-bit bus size.
3. The NA# pin operation in the 386SX CPU is identical to that of the NA# pin on the 386 CPU with one exception: the 386 CPU NA# pin cannot be activated on 16-bit bus cycles (where BS16# is LOW in the 386 CPU case), whereas NA# can be activated on any 386SX CPU bus cycle.
4. The contents of all 386SX CPU registers at reset are identical to the contents of the 386 CPU registers at reset, except the DX register. The DX register contains a component-stepping identifier at reset, i.e.

in 386 CPU, after reset	DH =	3 indicates 386 CPU
	DL =	revision number;
in 386SX CPU, after reset	DH =	23H indicates 386SX CPU
	DL =	revision number.

5. The 386 CPU uses A_{31} and M/IO# as selects for the numerics coprocessor. The 386SX CPU uses A_{23} and M/IO# as selects.
6. The 386 CPU prefetch unit fetches code in four-byte units. The 386SX CPU prefetch unit reads two bytes as one unit (like the 80286). In BS16 mode, the 386 CPU takes two consecutive bus cycles to complete a prefetch request. If there is a data read or write request after the prefetch starts, the 386 CPU will fetch all four bytes before addressing the new request.
7. Both 386 CPU and 386SX CPU have the same logical address space. The only difference is that the 386 CPU has a 32-bit physical address space and the 386SX CPU has a 24-bit physical address space. The 386SX CPU has a physical memory address space of up to 16 megabytes instead of the 4 gigabytes available to the 386 CPU. Therefore, in 386SX CPU systems, the operating system must be aware of this physical memory limit and should allocate memory for applications programs within this limit. If a 386 CPU system uses only the lower 16 megabytes of physical address, then there will be no extra effort required to migrate 386 CPU software to the 386SX CPU. Any application which uses more than 16 megabytes of memory can run on the 386SX CPU if the operating system utilizes the 386SX CPU's paging mechanism. In spite of this difference in physical address space, the 386SX CPU and 386 CPU can run the same operating systems and applications within their respective physical memory constraints.

9.0 INSTRUCTION SET

This section describes the instruction set. Table 9.1 lists all instructions along with instruction encoding diagrams and clock counts. Further details of the instruction encoding are then provided in the following sections, which completely describe the encoding structure and the definition of all fields occurring within instructions.

9.1 386SX™ CPU Instruction Encoding and Clock Count Summary

To calculate elapsed time for an instruction, multiply the instruction clock count, as listed in Table 9.1 below, by the processor clock period (e.g. 62.5 ns for an 386SX Microprocessor operating at 16 MHz). The actual clock count of an 386SX Microprocessor program will average 5% more than the calculated clock count due to instruction sequences which execute faster than they can be fetched from memory.

Instruction Clock Count Assumptions

1. The instruction has been prefetched, decoded, and is ready for execution.
2. Bus cycles do not require wait states.
3. There are no local bus HOLD requests delaying processor access to the bus.
4. No exceptions are detected during instruction execution.
5. If an effective address is calculated, it does not use two general register components. One register, scaling and displacement can be used within the clock counts shown. However, if the effective address calculation uses two general register components, add 1 clock to the clock count shown.

Instruction Clock Count Notation

1. If two clock counts are given, the smaller refers to a register operand and the larger refers to a memory operand.
2. n = number of times repeated.
3. m = number of components in the next instruction executed, where the entire displacement (if any) counts as one component, the entire immediate data (if any) counts as one component, and all other bytes of the instruction and prefix(es) each count as one component.

Misaligned or 32-Bit Operand Accesses

— If instructions accesses a misaligned 16-bit operand or 32-bit operand on even address add:
  2* clocks for read or write
  4** clocks for read and write
— If instructions accesses a 32-bit operand on odd address add:
  4* clocks for read or write
  8** clocks for read and write

Wait States

Wait states add 1 clock per wait state to instruction execution for each data access.

Table 9-1. Instruction Set Clock Count Summary

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
GENERAL DATA TRANSFER					
MOV = Move:					
Register to Register/Memory	1000100w mod reg r/m	2/2	2/2*	b	h
Register/Memory to Register	1000101w mod reg r/m	2/4	2/4*	b	h
Immediate to Register/Memory	1100011w mod 000 r/m immediate data	2/2	2/2*	b	h
Immediate to Register (short form)	1011 w reg immediate data	2	2		
Memory to Accumulator (short form)	1010000w full displacement	4*	4*	b	h
Accumulator to Memory (short form)	1010001w full displacement	2*	2*	b	h
Register Memory to Segment Register	10001110 mod sreg3 r/m	2/5	22/23	b	h, i, j
Segment Register to Register/Memory	10001100 mod sreg3 r/m	2/2	2/2	b	h
MOVSX = Move With Sign Extension					
Register From Register/Memory	00001111 1011111w mod reg r/m	3/6*	3/6*	b	h
MOVZX = Move With Zero Extension					
Register From Register/Memory	00001111 1011011w mod reg r/m	3/6*	3/6*	b	h
PUSH = Push:					
Register/Memory	11111111 mod 110 r/m	5/7*	7/9*	b	h
Register (short form)	01010 reg	2	4	b	h
Segment Register (ES, CS, SS or DS) (short form)	000 sreg2 110	2	4	b	h
Segment Register (ES, CS, SS, DS, FS or GS)	00001111 10 sreg3 000	2	4	b	h
Immediate	011010s0 immediate data	2	4	b	h
PUSHA = Push All	01100000	18	34	b	h
POP = Pop					
Register/Memory	10001111 mod 000 r/m	5/7	7/9	b	h
Register (short form)	01011 reg	6	6	b	h
Segment Register (ES, CS, SS or DS) (short form)	000 sreg 2 111	7	25	b	h, i, j
Segment Register (ES, CS, SS or DS), FS or GS	00001111 10 sreg 3 001	7	25	b	h, i, j
POPA = Pop All	01100001	24	40	b	h
XCHG = Exchange					
Register/Memory With Register	1000011w mod reg r/m	3/5**	3/5**	b, f	f, h
Register With Accumulator (short form)	10010 reg	3	3		
IN = Input from:		**Clk Count Virtual 8086 Mode**			
Fixed Port	1110010w port number	†26	12*	6*/26*	s/t,m
Variable Port	1110110w	†27	13*	7*/27*	s/t,m
OUT = Output to:					
Fixed Port	1110011w port number	†24	10*	4*/24*	s/t,m
Variable Port	1110111w	†25	11*	5*/25*	s/t,m
LEA = Load EA to Register	10001101 mod reg r/m		2	2	

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT		NOTES	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
SEGMENT CONTROL					
LDS = Load Pointer to DS	11000101 | mod reg r/m	7*	26*/28*	b	h, i, j
LES = Load Pointer to ES	11000100 | mod reg r/m	7*	26*/28*	b	h, i, j
LFS = Load Pointer to FS	00001111 | 10110100 | mod reg r/m	7*	29*/31*	b	h, i, j
LGS = Load Pointer to GS	00001111 | 10110101 | mod reg r/m	7*	26*/28*	b	h, i, j
LSS = Load Pointer to SS	00001111 | 10110010 | mod reg r/m	7*	26*/28*	b	h, i, j
FLAG CONTROL					
CLC = Clear Carry Flag	11111000	2	2		
CLD = Clear Direction Flag	11111100	2	2		
CLI = Clear Interrupt Enable Flag	11111010	8	8		m
CLTS = Clear Task Switched Flag	00001111 | 00000110	5	5	c	l
CMC = Complement Carry Flag	11110101	2	2		
LAHF = Load AH into Flag	10011111	2	2		
POPF = Pop Flags	10011101	5	5	b	h, n
PUSHF = Push Flags	10011100	4	4	b	h
SAHF = Store AH into Flags	10011110	3	3		
STC = Set Carry Flag	11111001	2	2		
STD = Set Direction Flag	11111101				
STI = Set Interrupt Enable Flag	11111011	8	8		m
ARITHMETIC					
ADD = Add					
Register to Register	000000dw | mod reg r/m	2	2		
Register to Memory	0000000w | mod reg r/m	7**	7**	b	h
Memory to Register	0000001w | mod reg r/m	6*	6*	b	h
Immediate to Register/Memory	100000sw | mod000 r/m | immediate data	2/7**	2/7**	b	h
Immediate to Accumulator (short form)	0000010w | immediate data	2	2		
ADC = Add With Carry					
Register to Register	000100dw | mod reg r/m	2	2		
Register to Memory	0001000w | mod reg r/m	7**	7**	b	h
Memory to Register	0001001w | mod reg r/m	6*	6*	b	h
Immediate to Register/Memory	100000sw | mod010 r/m | immediate data	2/7**	2/7**	b	h
Immediate to Accumulator (short form)	0001010w | immediate data	2	2		
INC = Increment					
Register/Memory	1111111w | mod000 r/m	2/6**	2/6**	b	h
Register (short form)	01000 reg	2	2		
SUB = Subtract					
Register from Register	001010dw | mod reg r/m	2	2		

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
ARITHMETIC (Continued)					
Register from Memory	0010100w | mod reg r/m	7**	7**	b	h
Memory from Register	0010101w | mod reg r/m	6*	6*	b	h
Immediate from Register/Memory	100000sw | mod 101 r/m | immediate data	2/7**	2/7**	b	h
Immediate from Accumulator (short form)	0010110w | immediate data	2	2		
SBB = Subtract with Borrow					
Register from Register	000110dw | mod reg r/m	2	2		
Register from Memory	0001100w | mod reg r/m	7**	7**	b	h
Memory from Register	0001101w | mod reg r/m	6*	6*	b	h
Immediate from Register/Memory	100000sw | mod 011 r/m | immediate data	2/7**	2/7**	b	h
Immediate from Accumulator (short form)	0001110w | immediate data	2	2		
DEC = Decrement					
Register/Memory	1111111w | reg 001 r/m	2/6	2/6	b	h
Register (short form)	01001 reg	2	2		
CMP = Compare					
Register with Register	001110dw | mod reg r/m	2	2		
Memory with Register	0011100w | mod reg r/m	5*	5*	b	h
Register with Memory	0011101w | mod reg r/m	6*	6*	b	h
Immediate with Register/Memory	100000sw | mod 111 r/m | immediate data	2/5*	2/5*	b	h
Immediate with Accumulator (short form)	0011110w | immediate data	2	2		
NEG = Change Sign	1111011w | mod 011 r/m	2/6*	2/6*	b	h
AAA = ASCII Adjust for Add	00110111	4	4		
AAS = ASCII Adjust for Subtract	00111111	4	4		
DAA = Decimal Adjust for Add	00100111	4	4		
DAS = Decimal Adjust for Subtract	00101111	4	4		
MUL = Multiply (unsigned)					
Accumulator with Register/Memory	1111011w | mod 100 r/m				
Multiplier-Byte		12-17/15-20*	12-17/15-20*	b, d	d, h
-Word		12-25/15-28*	12-25/15-28*	b, d	d, h
-Doubleword		12-41/17-46*	12-41/17-46*	b, d	d, h
IMUL = Integer Multiply (signed)					
Accumulator with Register/Memory	1111011w | mod 101 r/m				
Multiplier-Byte		12-17/15-20*	12-17/15-20*	b, d	d, h
-Word		12-25/15-28*	12-25/15-28*	b, d	d, h
-Doubleword		12-41/17-46*	12-41/17-46*	b, d	d, h
Register with Register/Memory	00001111 | 10101111 | mod reg r/m				
Multiplier-Byte		12-17/15-20*	12-17/15-20*	b, d	d, h
-Word		12-25/15-28*	12-25/15-28*	b, d	d, h
-Doubleword		12-41/17-46*	12-41/17-46*	b, d	d, h
Register/Memory with Immediate to Register	011010s1 | mod reg r/m | immediate data				
-Word		13-26	13-26/14-27	b, d	d, h
-Doubleword		13-42	13-42/16-45	b, d	d, h

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
ARITHMETIC (Continued)					
DIV = Divide (Unsigned)					
Accumulator by Register/Memory	1111011w mod110 r/m				
Divisor—Byte		14/17	14/17	b,e	e,h
—Word		22/25	22/25	b,e	e,h
—Doubleword		38/43	38/43	b,e	e,h
IDIV = Integer Divide (Signed)					
Accumulator By Register/Memory	1111011w mod111 r/m				
Divisor—Byte		19/22	19/22	b,e	e,h
—Word		27/30	27/30	b,e	e,h
—Doubleword		43/48	43/48	b,e	e,h
AAD = ASCII Adjust for Divide	11010101 00001010	19	19		
AAM = ASCII Adjust for Multiply	11010100 00001010	17	17		
CBW = Convert Byte to Word	10011000	3	3		
CWD = Convert Word to Double Word	10011001	2	2		
LOGIC					
Shift Rotate Instructions					
Not Through Carry (**ROL, ROR, SAL, SAR, SHL,** and **SHR**)					
Register/Memory by 1	1101000w mod TTT r/m	3/7**	3/7**	b	h
Register/Memory by CL	1101001w mod TTT r/m	3/7*	3/7*	b	h
Register/Memory by Immediate Count	1100000w mod TTT r/m immed 8-bit data	3/7*	3/7*	b	h
Through Carry (**RCL** and **RCR**)					
Register/Memory by 1	1101000w mod TTT r/m	9/10*	9/10*	b	h
Register/Memory by CL	1101001w mod TTT r/m	9/10*	9/10*	b	h
Register/Memory by Immediate Count	1100000w mod TTT r/m immed 8-bit data	9/10*	9/10*	b	h
	TTT Instruction: 000 ROL; 001 ROR; 010 RCL; 011 RCR; 100 SHL/SAL; 101 SHR; 111 SAR				
SHLD = Shift Left Double					
Register/Memory by Immediate	00001111 10100100 mod reg r/m immed 8-bit data	3/7**	3/7**		
Register/Memory by CL	00001111 10100101 mod reg r/m	3/7**	3/7**		
SHRD = Shift Right Double					
Register/Memory by Immediate	00001111 10101100 mod reg r/m immed 8-bit data	3/7**	3/7**		
Register/Memory by CL	00001111 10101101 mod reg r/m	3/7**	3/7**		
AND = And					
Register to Register	001000dw mod reg r/m	2	2		

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	Clk Count Virtual 8086 Mode	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
LOGIC (Continued)						
Register to Memory	0010000w mod reg r/m		7**	7**	b	h
Memory to Register	0010001w mod reg r/m		6*	6*	b	h
Immediate to Register/Memory	1000000w mod 100 r/m immediate data		2/7*	2/7**	b	h
Immediate to Accumulator (Short Form)	0010010w immediate data		2	2		
TEST = And Function to Flags, No Result						
Register/Memory and Register	1000010w mod reg r/m		2/5*	2/5*	b	h
Immediate Data and Register/Memory	1111011w mod 000 r/m immediate data		2/5*	2/5*	b	h
Immediate Data and Accumulator (Short Form)	1010100w immediate data		2	2		
OR = Or						
Register to Register	000010dw mod reg r/m		2	2		
Register to Memory	0000100w mod reg r/m		7**	7**	b	h
Memory to Register	0000101w mod reg r/m		6*	6*	b	h
Immediate to Register/Memory	1000000w mod 001 r/m immediate data		2/7**	2/7**	b	h
Immediate to Accumulator (Short Form)	0000110w immediate data		2	2		
XOR = Exclusive Or						
Register to Register	001100dw mod reg r/m		2	2		
Register to Memory	0011000w mod reg r/m		7**	7**	b	h
Memory to Register	0011001w mod reg r/m		6*	6*	b	h
Immediate to Register/Memory	1000000w mod 110 r/m immediate data		2/7**	2/7**	b	h
Immediate to Accumulator (Short Form)	0011010w immediate data		2	2		
NOT = Invert Register/Memory	1111011w mod 010 r/m		2/6**	2/6**	b	h
STRING MANIPULATION						
CMPS = Compare Byte Word	1010011w		10*	10*	b	h
INS = Input Byte/Word from DX Port	0110110w	†29	15	9*/29**	b	s/t, h, m
LODS = Load Byte/Word to AL/AX/EAX	1010110w		5	5*	b	h
MOVS = Move Byte Word	1010010w		7	7**	b	h
OUTS = Output Byte/Word to DX Port	0110111w	†28	14	8*/28*	b	s/t, h, m
SCAS = Scan Byte Word	1010111w		7*	7*	b	h
STOS = Store Byte/Word from AL/AX/EX	1010101w		4*	4*	b	h
XLAT = Translate String	11010111		5*	5*		h
REPEATED STRING MANIPULATION Repeated by Count in CX or ECX						
REPE CMPS = Compare String (Find Non-Match)	11110011 1010011w		5 + 9n**	5 + 9n**	b	h

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
REPEATED STRING MANIPULATION (Continued)					
REPNE CMPS = Compare String					
(Find Match)	11110010 1010011w — Clk Count Virtual 8086 Mode	5+9n**	5+9n**	b	h
REP INS = Input String	11110010 0110110w — †	13+6n*	7+6n*/ 27+6n*	b	s/t, h, m
REP LODS = Load String	11110010 1010110w	5+6n*	5+6n*	b	h
REP MOVS = Move String	11110010 1010010w	7+4n*	7+4n**	b	h
REP OUTS = Output String	11110010 0110111w — ‡	12+5n*	6+5n*/ 26+5n*	b	s/t, h, m
REPE SCAS = Scan String					
(Find Non-AL/AX/EAX)	11110011 1010111w	5+8n*	5+8n*	b	h
REPNE SCAS = Scan String					
(Find AL/AX/EAX)	11110010 1010111w	5+8n*	5+8n*	b	h
REP STOS = Store String	11110010 1010101w	5+5n*	5+5n*	b	h
BIT MANIPULATION					
BSF = Scan Bit Forward	00001111 10111100 mod reg r/m	10+3n*	10+3n**	b	h
BSR = Scan Bit Reverse	00001111 10111101 mod reg r/m	10+3n*	10+3n**	b	h
BT = Test Bit					
Register/Memory, Immediate	00001111 10111010 mod 100 r/m immed 8-bit data	3/6*	3/6*	b	h
Register/Memory, Register	00001111 10100011 mod reg r/m	3/12*	3/12*	b	h
BTC = Test Bit and Complement					
Register/Memory, Immediate	00001111 10111010 mod 111 r/m immed 8-bit data	6/8*	6/8*	b	h
Register/Memory, Register	00001111 10111011 mod reg r/m	6/13*	6/13*	b	h
BTR = Test Bit and Reset					
Register/Memory, Immediate	00001111 10111010 mod 110 r/m immed 8-bit data	6/8*	6/8*	b	h
Register/Memory, Register	00001111 10110011 mod reg r/m	6/13*	6/13*	b	h
BTS = Test Bit and Set					
Register/Memory, Immediate	00001111 10111010 mod 101 r/m immed 8-bit data	6/8*	6/8*	b	h
Register/Memory, Register	00001111 10101011 mod reg r/m	6/13*	6/13*	b	h
CONTROL TRANSFER					
CALL = Call					
Direct Within Segment	11101000 full displacement	7+m*	9+m*	b	r
Register/Memory					
Indirect Within Segment	11111111 mod 010 r/m	7+m*/10+m*	9+m/ 12+m*	b	h, r
Direct Intersegment	10011010 unsigned full offset, selector	17+m*	42+m*	b	j,k,r

NOTE:
† Clock count shown applies if I/O permission allows I/O to the port in virtual 8086 mode. If I/O bit map denies permission exception 13 fault occurs; refer to clock counts for INT 3 instruction.

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONTROL TRANSFER (Continued)					
Protected Mode Only (Direct Intersegment)					
Via Call Gate to Same Privilege Level			64+m		h,j,k,r
Via Call Gate to Different Privilege Level, (No Parameters)			98+m		h,j,k,r
Via Call Gate to Different Privilege Level, (x Parameters)			106+8x+m		h,j,k,r
From 286 Task to 286 TSS			285		h,j,k,r
From 286 Task to 386SX™ CPU TSS			310		h,j,k,r
From 286 Task to Virtual 8086 Task (386SX™ CPU TSS)			229		h,j,k,r
From 386SX™ CPU Task to 286 TSS			285		h,j,k,r
From 386SX™ CPU Task to 386SX™ CPU TSS			392		h,j,k,r
From 386SX™ CPU Task to Virtual 8086 Task (386SX™ CPU TSS)			309		h,j,k,r
Indirect Intersegment	11111111 mod011 r/m	30+m	46+m	b	h,j,k,r
Protected Mode Only (Indirect Intersegment)					
Via Call Gate to Same Privilege Level			68+m		h,j,k,r
Via Call Gate to Different Privilege Level, (No Parameters)			102+m		h,j,k,r
Via Call Gate to Different Privilege Level, (x Parameters)			110+8x+m		h,j,k,r
From 286 Task to 286 TSS					h,j,k,r
From 286 Task to 386SX™ CPU TSS					h,j,k,r
From 286 Task to Virtual 8086 Task (386SX™ CPU TSS)					h,j,k,r
From 386SX™ CPU Task to 286 TSS					h,j,k,r
From 386SX™ CPU Task to 386SX™ CPU TSS			399		h,j,k,r
From 386SX™ CPU Task to Virtual 8086 Task (386SX™ CPU TSS)					h,j,k,r
JMP = Unconditional Jump					
Short	11101011 8-bit displacement	7+m	7+m		r
Direct within Segment	11101001 full displacement	7+m	7+m		r
Register/Memory Indirect within Segment	11111111 mod100 r/m	9+m/14+m	9+m/14+m	b	h,r
Direct Intersegment	11101010 unsigned full offset, selector	16+m	31+m		j,k,r
Protected Mode Only (Direct Intersegment)					
Via Call Gate to Same Privilege Level			53+m		h,j,k,r
From 286 Task to 286 TSS					h,j,k,r
From 286 Task to 386SX™ CPU TSS					h,j,k,r
From 286 Task to Virtual 8086 Task (386SX™ CPU TSS)					h,j,k,r
From 386SX™ CPU Task to 286 TSS					h,j,k,r
From 386SX™ CPU Task to 386SX™ CPU TSS					h,j,k,r
From 386SX™ CPU Task to Virtual 8086 Task (386SX™ CPU TSS)			395		h,j,k,r
Indirect Intersegment	11111111 mod101 r/m	17+m	31+m	b	h,j,k,r
Protected Mode Only (Indirect Intersegment)					
Via Call Gate to Same Privilege Level			49+m		h,j,k,r
From 286 Task to 286 TSS					h,j,k,r
From 286 Task to 386SX™ CPU TSS					h,j,k,r
From 286 Task to Virtual 8086 Task (386SX™ CPU TSS)					h,j,k,r
From 386SX™ CPU Task to 286 TSS					h,j,k,r
From 386SX™ CPU Task to 386SX™ CPU TSS			328		h,j,k,r
From 386SX™ CPU Task to Virtual 8086 Task (386SX™ CPU TSS)					h,j,k,r

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONTROL TRANSFER (Continued)					
RET = Return from CALL:					
Within Segment	11000011		12+m	b	g, h, r
Within Segment Adding Immediate to SP	11000010 | 16-bit displ		12+m	b	g, h, r
Intersegment	11001011		36+m	b	g, h, j, k, r
Intersegment Adding Immediate to SP	11001010 | 16-bit displ		36+m	b	g, h, j, k, r
Protected Mode Only (RET):					
to Different Privilege Level					
Intersegment			72		h, j, k, r
Intersegment Adding Immediate to SP			72		h, j, k, r
CONDITIONAL JUMPS					
NOTE: Times Are Jump "Taken or Not Taken"					
JO = Jump on Overflow					
8-Bit Displacement	01110000 | 8-bit displ		7+m or 3	7+m or 3	r
Full Displacement	00001111 | 10000000 | full displacement		7+m or 3	7+m or 3	r
JNO = Jump on Not Overflow					
8-Bit Displacement	01110001 | 8-bit displ		7+m or 3	7+m or 3	r
Full Displacement	00001111 | 10000001 | full displacement		7+m or 3	7+m or 3	r
JB/JNAE = Jump on Below/Not Above or Equal					
8-Bit Displacement	01110010 | 8-bit displ		7+m or 3	7+m or 3	r
Full Displacement	00001111 | 10000010 | full displacement		7+m or 3	7+m or 3	r
JNB/JAE = Jump on Not Below/Above or Equal					
8-Bit Displacement	01110011 | 8-bit displ		7+m or 3	7+m or 3	r
Full Displacement	00001111 | 10000011 | full displacement		7+m or 3	7+m or 3	r
JE/JZ = Jump on Equal/Zero					
8-Bit Displacement	01110100 | 8-bit displ		7+m or 3	7+m or 3	r
Full Displacement	00001111 | 10000100 | full displacement		7+m or 3	7+m or 3	r
JNE/JNZ = Jump on Not Equal/Not Zero					
8-Bit Displacement	01110101 | 8-bit displ		7+m or 3	7+m or 3	r
Full Displacement	00001111 | 10000101 | full displacement		7+m or 3	7+m or 3	r
JBE/JNA = Jump on Below or Equal/Not Above					
8-Bit Displacement	01110110 | 8-bit displ		7+m or 3	7+m or 3	r
Full Displacement	00001111 | 10000110 | full displacement		7+m or 3	7+m or 3	r
JNBE/JA = Jump on Not Below or Equal/Above					
8-Bit Displacement	01110111 | 8-bit displ		7+m or 3	7+m or 3	r
Full Displacement	00001111 | 10000111 | full displacement		7+m or 3	7+m or 3	r
JS = Jump on Sign					
8-Bit Displacement	01111000 | 8-bit displ		7+m or 3	7+m or 3	r
Full Displacement	00001111 | 10001000 | full displacement		7+m or 3	7+m or 3	r

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT			CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONDITIONAL JUMPS (Continued)							
JNS = Jump on Not Sign							
8-Bit Displacement	01111001	8-bit displ		7+m or 3	7+m or 3		r
Full Displacement	00001111	10001001	full displacement	7+m or 3	7+m or 3		r
JP/JPE = Jump on Parity/Parity Even							
8-Bit Displacement	01111010	8-bit displ		7+m or 3	7+m or 3		r
Full Displacement	00001111	10001010	full displacement	7+m or 3	7+m or 3		r
JNP/JPO = Jump on Not Parity/Parity Odd							
8-Bit Displacement	01111011	8-bit displ		7+m or 3	7+m or 3		r
Full Displacement	00001111	10001011	full displacement	7+m or 3	7+m or 3		r
JL/JNGE = Jump on Less/Not Greater or Equal							
8-Bit Displacement	01111100	8-bit displ		7+m or 3	7+m or 3		r
Full Displacement	00001111	10001100	full displacement	7+m or 3	7+m or 3		r
JNL/JGE = Jump on Not Less/Greater or Equal							
8-Bit Displacement	01111101	8-bit displ		7+m or 3	7+m or 3		r
Full Displacement	00001111	10001101	full displacement	7+m or 3	7+m or 3		r
JLE/JNG = Jump on Less or Equal/Not Greater							
8-Bit Displacement	01111110	8-bit displ		7+m or 3	7+m or 3		r
Full Displacement	00001111	10001110	full displacement	7+m or 3	7+m or 3		r
JNLE/JG = Jump on Not Less or Equal/Greater							
8-Bit Displacement	01111111	8-bit displ		7+m or 3	7+m or 3		r
Full Displacement	00001111	10001111	full displacement	7+m or 3	7+m or 3		r
JCXZ = Jump on CX Zero	11100011	8-bit displ		9+m or 5	9+m or 5		r
JECXZ = Jump on ECX Zero	11100011	8-bit displ		9+m or 5	9+m or 5		r
(Address Size Prefix Differentiates JCXZ from JECXZ)							
LOOP = Loop CX Times	11100010	8-bit displ		11+m	11+m		r
LOOPZ/LOOPE = Loop with Zero/Equal	11100001	8-bit displ		11+m	11+m		r
LOOPNZ/LOOPNE = Loop While Not Zero	11100000	8-bit displ		11+m	11+m		r
CONDITIONAL BYTE SET NOTE: Times Are Register/Memory							
SETO = Set Byte on Overflow							
To Register/Memory	00001111	10010000	mod 000 r/m	4/5*	4/5*		h
SETNO = Set Byte on Not Overflow							
To Register/Memory	00001111	10010001	mod 000 r/m	4/5*	4/5*		h
SETB/SETNAE = Set Byte on Below/Not Above or Equal							
To Register/Memory	00001111	10010010	mod 000 r/m	4/5*	4/5*		h

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONDITIONAL BYTE SET (Continued)					
SETNB = Set Byte on Not Below/Above or Equal					
To Register/Memory	00001111 | 10010011 | mod 000 r/m	4/5*	4/5*		h
SETE/SETZ = Set Byte on Equal/Zero					
To Register/Memory	00001111 | 10010100 | mod 000 r/m	4/5*	4/5*		h
SETNE/SETNZ = Set Byte on Not Equal/Not Zero					
To Register/Memory	00001111 | 10010101 | mod 000 r/m	4/5*	4/5*		h
SETBE/SETNA = Set Byte on Below or Equal/Not Above					
To Register/Memory	00001111 | 10010110 | mod 000 r/m	4/5*	4/5*		h
SETNBE/SETA = Set Byte on Not Below or Equal/Above					
To Register/Memory	00001111 | 10010111 | mod 000 r/m	4/5*	4/5*		h
SETS = Set Byte on Sign					
To Register/Memory	00001111 | 10011000 | mod 000 r/m	4/5*	4/5*		h
SETNS = Set Byte on Not Sign					
To Register/Memory	00001111 | 10011001 | mod 000 r/m	4/5*	4/5*		h
SETP/SETPE = Set Byte on Parity/Parity Even					
To Register/Memory	00001111 | 10011010 | mod 000 r/m	4/5*	4/5*		h
SETNP/SETPO = Set Byte on Not Parity/Parity Odd					
To Register/Memory	00001111 | 10011011 | mod 000 r/m	4/5*	4/5*		h
SETL/SETNGE = Set Byte on Less/Not Greater or Equal					
To Register/Memory	00001111 | 10011100 | mod 000 r/m	4/5*	4/5*		h
SETNL/SETGE = Set Byte on Not Less/Greater or Equal					
To Register/Memory	00001111 | 01111101 | mod 000 r/m	4/5*	4/5*		h
SETLE/SETNG = Set Byte on Less or Equal/Not Greater					
To Register/Memory	00001111 | 10011110 | mod 000 r/m	4/5*	4/5*		h
SETNLE/SETG = Set Byte on Not Less or Equal/Greater					
To Register/Memory	00001111 | 10011111 | mod 000 r/m	4/5*	4/5*		h
ENTER = Enter Procedure	11001000 | 16-bit displacement, 8-bit level				
L = 0		10	10	b	h
L = 1		14	14	b	h
L > 1		17 + 8(n − 1)	17 + 8(n − 1)	b	h
LEAVE = Leave Procedure	11001001	4	4	b	h

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
INTERRUPT INSTRUCTIONS					
INT = Interrupt:					
Type Specified	11001101 | type	37		b	
Type 3	11001100	33		b	
INTO = Interrupt 4 if Overflow Flag Set	11001110				
If OF = 1		35		b, e	
If OF = 0		3	3	b, e	
Bound = Interrupt 5 if Detect Value Out of Range	01100010 | mod reg r/m				
If Out of Range		44		b, e	e, g, h, j, k, r
If In Range		10	10	b, e	e, g, h, j, k, r
Protected Mode Only (INT)					
INT: Type Specified					
Via Interrupt or Trap Gate					
Via Interrupt or Trap Gate					
to Same Privilege Level			71		g, j, k, r
to Different Privilege Level			111		g, j, k, r
From 286 Task to 286 TSS via Task Gate			438		g, j, k, r
From 286 Task to 386SX™ CPU TSS via Task Gate			465		g, j, k, r
From 286 Task to virt 8086 md via Task Gate			382		g, j, k, r
From 386SX™ CPU Task to 286 TSS via Task Gate			440		g, j, k, r
From 386SX™ CPU Task to 386SX™ CPU TSS via Task Gate			467		g, j, k, r
From 386SX™ CPU Task to virt 8086 md via Task Gate			384		g, j, k, r
From virt 8086 md to 286 TSS via Task Gate			445		g, j, k, r
From virt 8086 md to 386SX™ CPU TSS via Task Gate			472		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			275		
INT: TYPE 3					
Via Interrupt or Trap Gate					
to Same Privilege Level			71		g, j, k, r
Via Interrupt or Trap Gate					
to Different Privilege Level			111		g, j, k, r
From 286 Task to 286 TSS via Task Gate			382		g, j, k, r
From 286 Task to 386SX™ CPU TSS via Task Gate			409		g, j, k, r
From 286 Task to Virt 8086 md via Task Gate			326		g, j, k, r
From 386SX™ CPU Task to 286 TSS via Task Gate			384		g, j, k, r
From 386SX™ CPU Task to 386SX™ CPU TSS via Task Gate			411		g, j, k, r
From 386SX™ CPU Task to Virt 8086 md via Task Gate			328		g, j, k, r
From virt 8086 md to 286 TSS via Task Gate			389		g, j, k, r
From virt 8086 md to 386SX™ CPU TSS via Task Gate			416		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			223		
INTO:					
Via Interrupt or Trap Grate					
to Same Privilege Level			71		g, j, k, r
Via Interrupt or Trap Gate					
to Different Privilege Level			111		g, j, k, r
From 286 Task to 286 TSS via Task Gate			384		g, j, k, r
From 286 Task to 386SX™ CPU TSS via Task Gate			411		g, j, k, r
From 286 Task to virt 8086 md via Task Gate			328		g, j, k, r
From 386SX™ CPU Task to 286 TSS via Task Gate			386		g, j, k, r
From 386SX™ CPU Task to 386SX™ CPU TSS via Task Gate			413		g, j, k, r
From 386SX™ CPU Task to virt 8086 md via Task Gate			329		g, j, k, r
From virt 8086 md to 286 TSS via Task Gate			391		g, j, k, r
From virt 8086 md to 386SX™ CPU TSS via Task Gate			418		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			223		

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
INTERRUPT INSTRUCTIONS (Continued)					
BOUND:					
Via Interrupt or Trap Gate to Same Privilege Level			71		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			111		g, j, k, r
From 286 Task to 286 TSS via Task Gate			358		g, j, k, r
From 286 Task to 386SX™ CPU TSS via Task Gate			398		g, j, k, r
From 268 Task to virt 8086 Mode via Task Gate			335		g, j, k, r
From 386SX™ CPU Task to 286 TSS via Task Gate			368		g, j, k, r
From 386SX™ CPU Task to 386SX™ CPU TSS via Task Gate			398		g, j, k, r
From 386SX™ CPU Task to virt 8086 Mode via Task Gate			347		g, j, k, r,
From virt 8086 Mode to 286 TSS via Task Gate			368		g, j, k, r
From virt 8086 Mode to 386SX™ CPU TSS via Task Gate			398		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			223		
INTERRUPT RETURN					
IRET = Interrupt Return	11001111		24		g, h, j, k, r
Protected Mode Only (IRET)					
To the Same Privilege Level (within task)			42		g, h, j, k, r
To Different Privilege Level (within task)			86		g, h, j, k, r
From 286 Task to 286 TSS			285		h, j, k, r
From 286 Task to 386SX™ CPU TSS			318		h, j, k, r
From 286 Task to Virtual 8086 Task			267		h, j, k, r
From 286 Task to Virtual 8086 Mode (within task)			113		
From 386SX™ CPU Task to 286 TSS			324		h, j, k, r
From 386SX™ CPU Task to 386SX™ CPU TSS			328		h, j, k, r
From 386SX™ CPU Task to Virtual 8086 Task			377		h, j, k, r
From 386SX™ CPU Task to Virtual 8086 Mode (within task)			113		
PROCESSOR CONTROL					
HLT = HALT	11110100	5	5		I
MOV = Move to and From Control/Debug/Test Registers					
CR0/CR2/CR3 from register	00001111 00100010 11 eee reg	10/4/5	10/4/5		I
Register From CR0–3	00001111 00100000 11 eee reg	6	6		I
DR0–3 From Register	00001111 00100011 11 eee reg	22	22		I
DR6–7 From Register	00001111 00100011 11 eee reg	16	16		I
Register from DR6–7	00001111 00100001 11 eee reg	14	14		I
Register from DR0–3	00001111 00100001 11 eee reg	22	22		I
TR6–7 from Register	00001111 00100110 11 eee reg	12	12		I
Register from TR6–7	00001111 00100100 11 eee reg	12	12		I
NOP = No Operation	10010000	3	3		
WAIT = Wait until BUSY# pin is negated	10011011	6	6		

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
PROCESSOR EXTENSION INSTRUCTIONS					
Processor Extension Escape	11011TTT | mod LLL r/m TTT and LLL bits are opcode information for coprocessor.	See 387SX data sheet for clock counts			h
PREFIX BYTES					
Address Size Prefix	01100111	0	0		
LOCK = Bus Lock Prefix	11110000	0	0		m
Operand Size Prefix	01100110	0	0		
Segment Override Prefix					
CS:	00101110	0	0		
DS:	00111110	0	0		
ES:	00100110	0	0		
FS:	01100100	0	0		
GS:	01100101	0	0		
SS:	00110110	0	0		
PROTECTION CONTROL					
ARPL = Adjust Requested Privilege Level					
From Register/Memory	01100011 | mod reg r/m	N/A	20/21**	a	h
LAR = Load Access Rights					
From Register/Memory	00001111 | 00000010 | mod reg r/m	N/A	15/16*	a	g, h, j, p
LGDT = Load Global Descriptor					
Table Register	00001111 | 00000001 | mod 010 r/m	11*	11*	b, c	h, l
LIDT = Load Interrupt Descriptor					
Table Register	00001111 | 00000001 | mod 011 r/m	11*	11*	b, c	h, l
LLDT = Load Local Descriptor					
Table Register to Register/Memory	00001111 | 00000000 | mod 010 r/m	N/A	20/24*	a	g, h, j, l
LMSW = Load Machine Status Word					
From Register/Memory	00001111 | 00000001 | mod 110 r/m	10/13	10/13*	b, c	h, l
LSL = Load Segment Limit					
From Register/Memory	00001111 | 00000011 | mod reg r/m				
Byte-Granular Limit		N/A	20/21*	a	g, h, j, p
Page-Granular Limit		N/A	25/26*	a	g, h, j, p
LTR = Load Task Register					
From Register/Memory	00001111 | 00000000 | mod 001 r/m	N/A	23/27*	a	g, h, j, l
SGDT = Store Global Descriptor					
Table Register	00001111 | 00000001 | mod 000 r/m	9*	9*	b, c	h
SIDT = Store Interrupt Descriptor					
Table Register	00001111 | 00000001 | mod 001 r/m	9*	9*	b, c	h
SLDT = Store Local Descriptor Table Register					
To Register/Memory	00001111 | 00000000 | mod 000 r/m	N/A	2/2*	a	h

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
PROTECTION CONTROL (Continued)					
SMSW = **Store Machine Status Word**	00001111 | 00000001 | mod 1 0 0 r/m	2/2*	2/2*	b, c	h, l
STR = **Store Task Register** To Register/Memory	00001111 | 00000000 | mod 0 0 1 r/m	N/A	2/2*	a	h
VERR = **Verify Read Access** Register/Memory	00001111 | 00000000 | mod 1 0 0 r/m	N/A	10/11*	a	g, h, j, p
VERW = **Verify Write Access**	00001111 | 00000000 | mod 1 0 1 r/m	N/A	15/16*	a	g, h, j, p

INSTRUCTION NOTES FOR TABLE 9-1

Notes a through c apply to Real Address Mode only:

a. This is a Protected Mode instruction. Attempted execution in Real Mode will result in exception 6 (invalid opcode).

b. Exception 13 fault (general protection) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum CS, DS, ES, FS or GS limit, FFFFH. Exception 12 fault (stack segment limit violation or not present) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum SS limit.

c. This instruction may be executed in Real Mode. In Real Mode, its purpose is primarily to initialize the CPU for Protected Mode.

Notes d through g apply to Real Address Mode and Protected Virtual Address Mode:

d. The 386SX CPU uses an early-out multiply algorithm. The actual number of clocks depends on the position of the most significant bit in the operand (multiplier).

Clock counts given are minimum to maximum. To calculate actual clocks use the following formula:

Actual Clock = if $m < > 0$ then max ($[\log_2 |m|]$, 3) + b clocks:
if $m = 0$ then 3+b clocks

In this formula, m is the multiplier, and

b = 9 for register to register,
b = 12 for memory to register,
b = 10 for register with immediate to register,
b = 11 for memory with immediate to register.

e. An exception may occur, depending on the value of the operand.

f. LOCK# is automatically asserted, regardless of the presence or absence of the LOCK# prefix.

g. LOCK# is asserted during descriptor table accesses.

Notes h through r apply to Protected Virtual Address Mode only:

h. Exception 13 fault (general protection violation) will occur if the memory operand in CS, DS, ES, FS or GS cannot be used due to either a segment limit violation or access rights violation. If a stack limit is violated, an exception 12 (stack segment limit violation or not present) occurs.

i. For segment load operations, the CPL, RPL, and DPL must agree with the privilege rules to avoid an exception 13 fault (general protection violation). The segment's descriptor must indicate "present" or exception 11 (CS, DS, ES, FS, GS not present). If the SS register is loaded and a stack segment not present is detected, an exception 12 (stack segment limit violation or not present) occurs.

j. All segment descriptor accesses in the GDT or LDT made by this instruction will automatically assert LOCK# to maintain descriptor integrity in multiprocessor systems.

k. JMP, CALL, INT, RET and IRET instructions referring to another code segment will cause an exception 13 (general protection violation) if an applicable privilege rule is violated.

l. An exception 13 fault occurs if CPL is greater than 0 (0 is the most privileged level).

m. An exception 13 fault occurs if CPL is greater than IOPL.

n. The IF bit of the flag register is not updated if CPL is greater than IOPL. The IOPL and VM fields of the flag register are updated only if CPL = 0.

o. The PE bit of the MSW (CR0) cannot be reset by this instruction. Use MOV into CR0 if desiring to reset the PE bit.

p. Any violation of privilege rules as applied to the selector operand does not cause a protection exception; rather, the zero flag is cleared.

q. If the coprocessor's memory operand violates a segment limit or segment access rights, an exception 13 fault (general protection exception) will occur before the ESC instruction is executed. An exception 12 fault (stack segment limit violation or not present) will occur if the stack limit is violated by the operand's starting address.

r. The destination of a JMP, CALL, INT, RET or IRET must be in the defined limit of a code segment or an exception 13 fault (general protection violation) will occur.

s/t. The instruction will execute in s clocks if CPL $\leq$ IOPL. If CPL > IOPL, the instruction will take t clocks.

9.2 INSTRUCTION ENCODING

9.2.1 Overview

All instruction encodings are subsets of the general instruction format shown in Figure 8-1. Instructions consist of one or two primary opcode bytes, possibly an address specifier consisting of the "mod r/m" byte and "scaled index" byte, a displacement if required, and an immediate data field if required.

Within the primary opcode or opcodes, smaller encoding fields may be defined. These fields vary according to the class of operation. The fields define such information as direction of the operation, size of the displacements, register encoding, or sign extension.

Almost all instructions referring to an operand in memory have an addressing mode byte following the primary opcode byte(s). This byte, the mod r/m byte, specifies the address mode to be used. Certain encodings of the mod r/m byte indicate a second addressing byte, the scale-index-base byte, follows the mod r/m byte to fully specify the addressing mode.

Addressing modes can include a displacement immediately following the mod r/m byte, or scaled index byte. If a displacement is present, the possible sizes are 8, 16 or 32 bits.

If the instruction specifies an immediate operand, the immediate operand follows any displacement bytes. The immediate operand, if specified, is always the last field of the instruction.

Figure 9-1 illustrates several of the fields that can appear in an instruction, such as the mod field and the r/m field, but the Figure does not show all fields. Several smaller fields also appear in certain instructions, sometimes within the opcode bytes themselves. Table 9-2 is a complete list of all fields appearing in the instruction set. Further ahead, following Table 9-2, are detailed tables for each field.

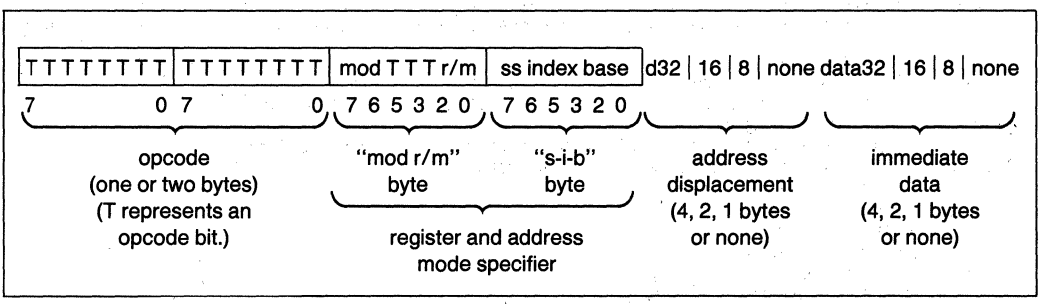


Figure 9-1. General Instruction Format

Table 9-2. Fields within Instructions

Field Name	Description	Number of Bits
w	Specifies if Data is Byte or Full Size (Full Size is either 16 or 32 Bits	1
d	Specifies Direction of Data Operation	1
s	Specifies if an Immediate Data Field Must be Sign-Extended	1
reg	General Register Specifier	3
mod r/m	Address Mode Specifier (Effective Address can be a General Register)	2 for mod; 3 for r/m
ss	Scale Factor for Scaled Index Address Mode	2
index	General Register to be used as Index Register	3
base	General Register to be used as Base Register	3
sreg2	Segment Register Specifier for CS, SS, DS, ES	2
sreg3	Segment Register Specifier for CS, SS, DS, ES, FS, GS	3
tttn	For Conditional Instructions, Specifies a Condition Asserted or a Condition Negated	4

Note: Table 9-1 shows encoding of individual instructions.

9.2.2 32-Bit Extensions of the Instruction Set

With the 386SX CPU, the 8086/80186/80286 instruction set is extended in two orthogonal directions: 32-bit forms of all 16-bit instructions are added to support the 32-bit data types, and 32-bit addressing modes are made available for all instructions referencing memory. This orthogonal instruction set extension is accomplished having a Default (D) bit in the code segment descriptor, and by having 2 prefixes to the instruction set.

Whether the instruction defaults to operations of 16 bits or 32 bits depends on the setting of the D bit in the code segment descriptor, which gives the default length (either 32 bits or 16 bits) for both operands and effective addresses when executing that code segment. In the Real Address Mode or Virtual 8086 Mode, no code segment descriptors are used, but a D value of 0 is assumed internally by the 386SX CPU when operating in those modes (for 16-bit default sizes compatible with the 8086/80186/80286).

Two prefixes, the Operand Size Prefix and the Effective Address Size Prefix, allow overriding individually the Default selection of operand size and effective address size. These prefixes may precede any opcode bytes and affect only the instruction they precede. If necessary, one or both of the prefixes may be placed before the opcode bytes. The presence of the Operand Size Prefix and the Effective Address Prefix will toggle the operand size or the effective address size, respectively, to the value "opposite" from the Default setting. For example, if the default operand size is for 32-bit data operations, then presence of the Operand Size Prefix toggles the instruction to 16-bit data operation. As another example, if the default effective address size is 16 bits, presence of the Effective Address Size prefix toggles the instruction to use 32-bit effective address computations.

These 32-bit extensions are available in all modes, including the Real Address Mode or the Virtual 8086 Mode. In these modes the default is always 16 bits, so prefixes are needed to specify 32-bit operands or addresses. For instructions with more than one prefix, the order of prefixes is unimportant.

Unless specified otherwise, instructions with 8-bit and 16-bit operands do not affect the contents of the high-order bits of the extended registers.

9.2.3 Encoding of Instruction Fields

Within the instruction are several fields indicating register selection, addressing mode and so on. The exact encodings of these fields are defined immediately ahead.

9.2.3.1 ENCODING OF OPERAND LENGTH (w) FIELD

For any given instruction performing a data operation, the instruction is executing as a 32-bit operation or a 16-bit operation. Within the constraints of the operation size, the w field encodes the operand size as either one byte or the full operation size, as shown in the table below.

w Field	Operand Size During 16-Bit Data Operations	Operand Size During 32-Bit Data Operations
0	8 Bits	8 Bits
1	16 Bits	32 Bits

9.2.3.2 ENCODING OF THE GENERAL REGISTER (reg) FIELD

The general register is specified by the reg field, which may appear in the primary opcode bytes, or as the reg field of the "mod r/m" byte, or as the r/m field of the "mod r/m" byte.

Encoding of reg Field When w Field is not Present in Instruction

reg Field	Register Selected During 16-Bit Data Operations	Register Selected During 32-Bit Data Operations
000	AX	EAX
001	CX	ECX
010	DX	EDX
011	BX	EBX
100	SP	ESP
101	BP	EBP
101	SI	ESI
101	DI	EDI

Encoding of reg Field When w Field is Present in Instruction

Register Specified by reg Field During 16-Bit Data Operations:		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Register Specified by reg Field During 32-Bit Data Operations		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	EAX
001	CL	ECX
010	DL	EDX
011	BL	EBX
100	AH	ESP
101	CH	EBP
110	DH	ESI
111	BH	EDI

9.2.3.3 ENCODING OF THE SEGMENT REGISTER (sreg) FIELD

The sreg field in certain instructions is a 2-bit field allowing one of the four 80286 segment registers to be specified. The sreg field in other instructions is a 3-bit field, allowing the 386SX CPU FS and GS segment registers to be specified.

2-Bit sreg2 Field

2-Bit sreg2 Field	Segment Register Selected
00	ES
01	CS
10	SS
11	DS

3-Bit sreg3 Field

3-Bit sreg3 Field	Segment Register Selected
000	ES
001	CS
010	SS
011	DS
100	FS
101	GS
110	do not use
111	do not use

9.2.3.4 ENCODING OF ADDRESS MODE

Except for special instructions, such as PUSH or POP, where the addressing mode is pre-determined, the addressing mode for the current instruction is specified by addressing bytes following the primary opcode. The primary addressing byte is the "mod r/m" byte, and a second byte of addressing information, the "s-i-b" (scale-index-base) byte, can be specified.

The s-i-b byte (scale-index-base byte) is specified when using 32-bit addressing mode and the "mod r/m" byte has r/m = 100 and mod = 00, 01 or 10. When the sib byte is present, the 32-bit addressing mode is a function of the mod, ss, index, and base fields.

The primary addressing byte, the "mod r/m" byte, also contains three bits (shown as TTT in Figure 8-1) sometimes used as an extension of the primary opcode. The three bits, however, may also be used as a register field (reg).

When calculating an effective address, either 16-bit addressing or 32-bit addressing is used. 16-bit addressing uses 16-bit address components to calculate the effective address while 32-bit addressing uses 32-bit address components to calculate the effective address. When 16-bit addressing is used, the "mod r/m" byte is interpreted as a 16-bit addressing mode specifier. When 32-bit addressing is used, the "mod r/m" byte is interpreted as a 32-bit addressing mode specifier.

Tables on the following three pages define all encodings of all 16-bit addressing modes and 32-bit addressing modes.

Encoding of 16-bit Address Mode with "mod r/m" Byte

mod r/m	Effective Address
00 000	DS:[BX + SI]
00 001	DS:[BX + DI]
00 010	SS:[BP + SI]
00 011	SS:[BP + DI]
00 100	DS:[SI]
00 101	DS:[DI]
00 110	DS:d16
00 111	DS:[BX]
01 000	DS:[BX + SI + d8]
01 001	DS:[BX + DI + d8]
01 010	SS:[BP + SI + d8]
01 011	SS:[BP + DI + d8]
01 100	DS:[SI + d8]
01 101	DS:[DI + d8]
01 110	SS:[BP + d8]
01 111	DS:[BX + d8]

mod r/m	Effective Address
10 000	DS:[BX + SI + d16]
10 001	DS:[BX + DI + d16]
10 010	SS:[BP + SI + d16]
10 011	SS:[BP + DI + d16]
10 100	DS:[SI + d16]
10 101	DS:[DI + d16]
10 110	SS:[BP + d16]
10 111	DS:[BX + d16]
11 000	register—see below
11 001	register—see below
11 010	register—see below
11 011	register—see below
11 100	register—see below
11 101	register—see below
11 110	register—see below
11 111	register—see below

Register Specified by r/m During 16-Bit Data Operations		
mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Register Specified by r/m During 32-Bit Data Operations		
mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Encoding of 32-bit Address Mode with "mod r/m" byte (no "s-i-b" byte present):

mod r/m	Effective Address
00 000	DS:[EAX]
00 001	DS:[ECX]
00 010	DS:[EDX]
00 011	DS:[EBX]
00 100	s-i-b is present
00 101	DS:d32
00 110	DS:[ESI]
00 111	DS:[EDI]
01 000	DS:[EAX + d8]
01 001	DS:[ECX + d8]
01 010	DS:[EDX + d8]
01 011	DS:[EBX + d8]
01 100	s-i-b is present
01 101	SS:[EBP + d8]
01 110	DS:[ESI + d8]
01 111	DS:[EDI + d8]

mod r/m	Effective Address
10 000	DS:[EAX + d32]
10 001	DS:[ECX + d32]
10 010	DS:[EDX + d32]
10 011	DS:[EBX + d32]
10 100	s-i-b is present
10 101	SS:[EBP + d32]
10 110	DS:[ESI + d32]
10 111	DS:[EDI + d32]
11 000	register—see below
11 001	register—see below
11 010	register—see below
11 011	register—see below
11 100	register—see below
11 101	register—see below
11 110	register—see below
11 111	register—see below

Register Specified by reg or r/m during 16-Bit Data Operations:		
mod r/m	function of w field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Register Specified by reg or r/m during 32-Bit Data Operations:		
mod r/m	function of w field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Encoding of 32-bit Address Mode ("mod r/m" byte and "s-i-b" byte present):

mod base	Effective Address
00 000	DS:[EAX + (scaled index)]
00 001	DS:[ECX + (scaled index)]
00 010	DS:[EDX + (scaled index)]
00 011	DS:[EBX + (scaled index)]
00 100	SS:[ESP + (scaled index)]
00 101	DS:[d32 + (scaled index)]
00 110	DS:[ESI + (scaled index)]
00 111	DS:[EDI + (scaled index)]
01 000	DS:[EAX + (scaled index) + d8]
01 001	DS:[ECX + (scaled index) + d8]
01 010	DS:[EDX + (scaled index) + d8]
01 011	DS:[EBX + (scaled index) + d8]
01 100	SS:[ESP + (scaled index) + d8]
01 101	SS:[EBP + (scaled index) + d8]
01 110	DS:[ESI + (scaled index) + d8]
01 111	DS:[EDI + (scaled index) + d8]
10 000	DS:[EAX + (scaled index) + d32]
10 001	DS:[ECX + (scaled index) + d32]
10 010	DS:[EDX + (scaled index) + d32]
10 011	DS:[EBX + (scaled index) + d32]
10 100	SS:[ESP + (scaled index) + d32]
10 101	SS:[EBP + (scaled index) + d32]
10 110	DS:[ESI + (scaled index) + d32]
10 111	DS:[EDI + (scaled index) + d32]

NOTE:
Mod field in "mod r/m" byte; ss, index, base fields in "s-i-b" byte.

ss	Scale Factor
00	x1
01	x2
10	x4
11	x8

index	Index Register
000	EAX
001	ECX
010	EDX
011	EBX
100	no index reg**
101	EBP
110	ESI
111	EDI

****IMPORTANT NOTE:**
When index field is 100, indicating "no index register," then ss field MUST equal 00. If index is 100 and ss does not equal 00, the effective address is undefined.

9.2.3.5 ENCODING OF OPERATION DIRECTION (d) FIELD

In many two-operand instructions the d field is present to indicate which operand is considered the source and which is the destination.

d	Direction of Operation
0	Register/Memory <- - Register "reg" Field Indicates Source Operand; "mod r/m" or "mod ss index base" Indicates Destination Operand
1	Register <- - Register/Memory "reg" Field Indicates Destination Operand; "mod r/m" or "mod ss index base" Indicates Source Operand

9.2.3.6 ENCODING OF SIGN-EXTEND (s) FIELD

The s field occurs primarily to instructions with immediate data fields. The s field has an effect only if the size of the immediate data is 8 bits and is being placed in a 16-bit or 32-bit destination.

s	Effect on Immediate Data8	Effect on Immediate Data 16|32
0	None	None
1	Sign-Extend Data8 to Fill 16-Bit or 32-Bit Destination	None

9.2.3.7 ENCODING OF CONDITIONAL TEST (tttn) FIELD

For the conditional instructions (conditional jumps and set on condition), tttn is encoded with n indicating to use the condition (n=0) or its negation (n=1), and tt giving the condition to test.

Mnemonic	Condition	tttn
O	Overflow	0000
NO	No Overflow	0001
B/NAE	Below/Not Above or Equal	0010
NB/AE	Not Below/Above or Equal	0011
E/Z	Equal/Zero	0100
NE/NZ	Not Equal/Not Zero	0101
BE/NA	Below or Equal/Not Above	0110
NBE/A	Not Below or Equal/Above	0111
S	Sign	1000
NS	Not Sign	1001
P/PE	Parity/Parity Even	1010
NP/PO	Not Parity/Parity Odd	1011
L/NGE	Less Than/Not Greater or Equal	1100
NL/GE	Not Less Than/Greater or Equal	1101
LE/NG	Less Than or Equal/Greater Than	1110
NLE/G	Not Less or Equal/Greater Than	1111

9.2.3.8 ENCODING OF CONTROL OR DEBUG OR TEST REGISTER (eee) FIELD

For the loading and storing of the Control, Debug and Test registers.

When Interpreted as Control Register Field

eee Code	Reg Name
000	CR0
010	CR2
011	CR3
Do not use any other encoding	

When Interpreted as Debug Register Field

eee Code	Reg Name
000	DR0
001	DR1
010	DR2
011	DR3
110	DR6
111	DR7
Do not use any other encoding	

When Interpreted as Test Register Field

eee Code	Reg Name
110	TR6
111	TR7
Do not use any other encoding	

DATA SHEET REVISION REVIEW

The following list represents key differences between this and the -001 version of the 386SX™ microprocessor data sheet. Please review this summary carefully.

The sections significantly revised since version -001 are:

Front Page	The Microarchitecture diagram was added.
Section 2.0	Figure 2.1 was updated to show the 16-bit registers SI, DI, BP and SP.
Section 2.1	Figure 2.2 was updated to show the correct bit polarity for bit 4 in the CR0 register.
Section 2.1	Table 2.1 was updated to include additional information on the EFLAGs register.
Section 2.7	In the subsection Maskable Interrupt a paragraph was added to describe the effect of interrupt gates on the IF EFLAGs bit.
Section 4.2	Figures 4.4 and 4.5 were updated to show the AVL bit field.
Section 4.5	The last sentence in the first paragraph of subsection **PROTECTION AND I/O PERMISSION BIT MAP** was deleted. This was an incorrect statement.
Section 5.1	In the Subsection **ADDRESS BUS (BHE#, BLE#, A_{23}-A_1),** the last sentence in the first paragraph was updated to reflect the numerics operand addresses as 8000FCH and 8000FEH. Because the 386SX CPU sometimes does a double word I/O access a second access, to 8000FEH, can be seen.
Section 5.1	The Subsection **Hold Latencies** was updated to describe how 32-bit and unaligned accesses are internally locked but do not assert the LOCK# signal.
Section 5.4	Figure 5.14 was modified to show the correct states of A1 and BLE# during a Halt Indication Cycle.
Section 7.5	This entire section was updated to reflect the new ICE-386SX emulator.
Section 9.2	The section **INSTRUCTION ENCODING** was appended to the data sheet.

80387SX
80-BIT NUMERIC PROCESSOR EXTENSION

- **High Performance 80-Bit Internal Architecture**
- **Two to Three Times 8087/80287 Performance at Equivalent Clock Speed**
- **Implements ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic**
- **Fully compatible with 80387. Implements all 80387 architectural enhancements over 8087 and 80287.**
- **Upward Object-Code Compatible from 8087 and 80287**
- **Interfaces with 386SX™ Microprocessor**
- **Expands 386SX™ CPU Data Types to Include 32-, 64-, 80-Bit Floating Point, 32-, 64-Bit Integers and 18-Digit BCD Operands**
- **Directly Extends 386SX™ CPU Instruction Set to Trigonometric, Logarithmic, Exponential, and Arithmetic Instructions for All Data Types**
- **Full-Range Transcendental Operations for SINE, COSINE, TANGENT, ARCTANGENT, and LOGARITHM.**
- **Built-In Exception Handling**
- **Operates Independently of Real, Protected, and Virtual-8086 Modes of the 386SX™ Microprocessor**
- **Eight 80-Bit Numeric Registers, Usable as Individually Addressable General Registers or as a Register Stack**
- **Available in a 68-pin PLCC Package (see Packaging Specs: Order #231369)**

The Intel 80387SX is a high-performance numerics processor extension that extends the architecture of the 386SX™ Microprocessor with floating point, extended integer, and BCD data types. A computing system that includes the 80387SX fully conforms to the IEEE Floating Point Standard. Using a numerics oriented architecture, the 80387SX adds over seventy mnemonics to the instruction set of the 386SX Microprocessor, making a complete solution for high-performance numerics processing. The 80387SX is implemented with 1.5 micron, high-speed CHMOS III technology. The 80387SX is upward object-code compatible from the 80287 and 8087 numerics coprocessors and completely object-code compatible with the 80387 numerics coprocessor.

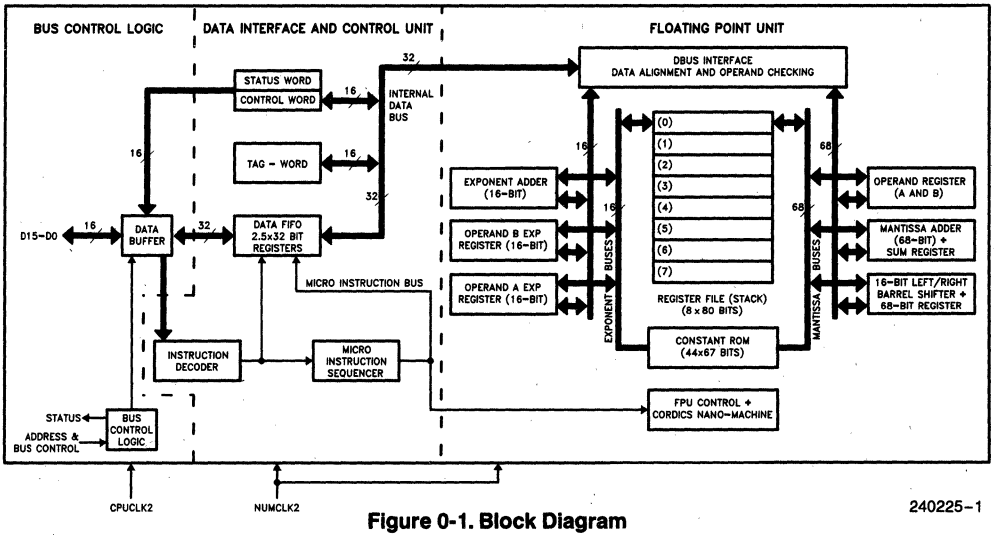


Figure 0-1. Block Diagram

November 1988
Order Number: 240225-002

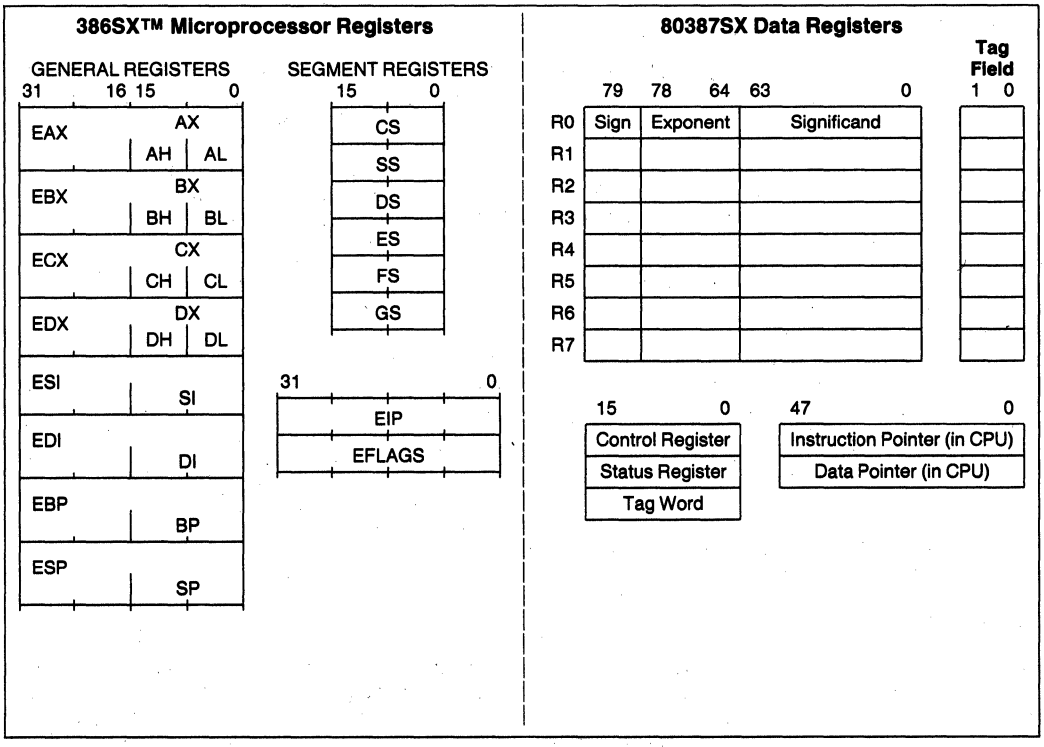


Figure 1-1. 386SX™ Microprocessor and 80387SX Coprocessor Register Set

1.0 FUNCTIONAL DESCRIPTION

The 80387SX Numerics Processor Extension (NPX) provides arithmetic instructions for a variety of numeric data types. It also executes numerous built-in transcendental functions (e.g. tangent, sine, cosine, and log functions). The 80387SX effectively extends the register and instruction set of its CPU for existing data types and adds several new data types as well. Figure 1-1 shows the model of registers visible to the 386SX™ Microprocessor and 80387 Coprocessor applications programs. Essentially, the 80387SX can be treated as an additional resource or an extension to the 386SX Microprocessor. The 386SX Microprocessor together with an 80387SX NPX can be used as a single unified system, the 386SX Microprocessor and 80387 Coprocessor.

The 80387SX Numerics Processor Extension works the same whether the CPU is executing in real-address mode, protected mode, or virtual-8086 mode. All references to memory for numerics data or status information are performed by the CPU, and therefore obey the memory-management and protection rules of the CPU mode currently in effect. The 80387SX Numerics Processor Extension merely operates on instructions and values passed to it by the CPU and therefore is not sensitive to the processing mode of the CPU.

In real-address mode and virtual-8086 mode, the 386SX Microprocessor and 80387 Coprocessor is completely upward compatible with software for the 8086/8087 and 80286/80287 real-address mode systems.

In protected mode, the 386SX Microprocessor and 80387 Coprocessor is completely upward compatible with software for the 80286/80287 protected mode system.

In all modes, the 386SX Microprocessor and 80387 Coprocessor is completely compatible with software for the 80836/80387 system.

The only differences of operation that may appear when 8086/8087 programs are ported to the protected-mode 386SX Microprocessor and 80387 Coprocessor system (*not* using virtual-8086 mode) is in the format of operands for the administrative instructions FLDENV, FSTENV, FRSTOR, and FSAVE. These instruction are normally used only by exception handlers and operating systems, not by applications programs.

2.0 PROGRAMMING INTERFACE

The 80387SX NPX adds to an 386SX Microprocessor system additional data types, registers, instructions, and interrupts specifically designed to facilitate high-speed numerics processing. To use the 80387SX NPX requires no special programming tools, because all new instructions and data types are directly supported by the assembler and compilers for high-level languages. All 386 Microprocessor development tools that support 80387 programs can also be used to develop software for the 386SX Microprocessor and 80387 Coprocessor. All 8086/8088 development tools that support the 8087 can also be used to develop software for the 386SX Microprocessor and 80387 Coprocessor in real-address mode or virtual-8086 mode. All 80286 development tools that support the 80287 can also be used to develop software for the 386SX Microprocessor and 80387 Coprocessor.

The 80387SX NPX supports all 80387 instructions. The 386SX Microprocessor and 80387 Coprocessor supports all the same programs and gives the same results as an 386 Microprocessor and 80387 Coprocessor.

All communication between the CPU and the NPX is transparent to applications software. The CPU automatically controls the NPX whenever a numerics instruction is executed. All physical memory and virtual memory of the CPU are available for storage of the instructions and operands of programs that use the NPX. All memory addressing modes, including use of displacement, base register, index register, and scaling, are available for addressing numerics operands.

Section 7 at the end of this data sheet lists by class the instructions that the 80387SX NPX adds to the instruction set of an 386SX Microprocessor system.

2.1 Data Types

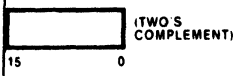
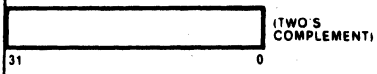
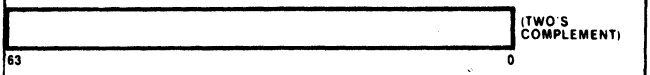
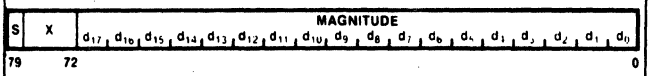
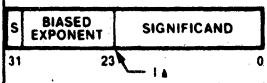
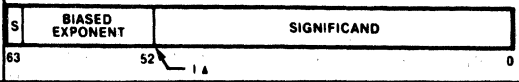
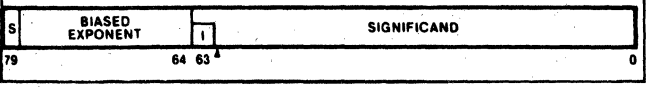
Table 2-1 lists the seven data types that the NPX supports and presents the format for each type. Operands are stored in memory with the least significant digit at the lowest memory address. Programs retrieve these values by generating the lowest address. For maximum system performance, all operands should start at physical-memory addresses that correspond to the word size of the CPU; operands may begin at any other addresses, but will require extra memory cycles to access the entire operand.

Internally, the NPX holds all numbers in the extended-precision real format. Instructions that load operands from memory automatically convert operands represented in memory as 16-, 32-, or 64-bit integers, 32- or 64-bit floating-point numbers, or 18-digit packed BCD numbers into extended-precision real format. Instructions that store operands in memory perform the inverse type conversion.

2.2 Numeric Operands

A typical NPX instruction accepts one or two operands and produces one (or sometimes two) results. In two-operand instructions, one operand is the contents of an NPX register, while the other may be a memory location. The operands of some instructions are predefined; for example, FSQRT always takes the square root of the number in the top stack element.

Table 2-1. 80387SX Data Type Representation in Memory

Data Formats	Range	Precision	Most Significant Byte = HIGHEST ADDRESSED BYTE
Word Integer	$\pm 10^4$	16 Bits	
Short Integer	$\pm 10^9$	32 Bits	
Long Integer	$\pm 10^{18}$	64 Bits	
Packed BCD	$\pm 10^{18}$	18 Digits	
Single Precision	$\pm 10^{\pm 38}$	24 Bits	
Double Precision	$\pm 10^{\pm 308}$	53 Bits	
Extended Precision	$\pm 10^{\pm 4932}$	64 Bits	

240225-2

NOTES:
(1) S = Sign bit (0 = positive, 1 = negative)
(2) d_n = Decimal digit (two per byte)
(3) X = Bits have no significance; 80387 ignores when loading, zeros when storing
(4) ▲ = Position of implicit binary point
(5) I = Integer bit of significand; stored in temporary real, implicit in single and double precision
(6) Exponent Bias (normalized values):
Single: 127 (7FH)
Double: 1023 (3FFH)
Extended REal: 16383 (3FFFH)
(7) Packed BCD: $(-1)^S (D_{17}..D_0)$
(8) Real: $(-1)^S (2^{E-BIAS}) (F_0 F_1...)$

2.3 Register Set

Figure 1-1 shows the 80387SX register set. When an NPX is present in a system, programmers may use these registers in addition to the registers normally available on the CPU.

2.3.1 DATA REGISTERS

80387SX computations use the 80387SX's data registers. These eight 80-bit registers provide the equivalent capacity of 20 32-bit registers. Each of the eight data registers in the NPX is 80 bits wide and is divided into "fields" corresponding to the NPX's extended-precision real data type.

The NPX register set can be accessed either as a stack, with instructions operating on the top one or two stack elements, or as individually addressable registers. The TOP field in the status word identifies the current top-of-stack register. A "push" operation decrements TOP by one and loads a value into the new top register. A "pop" operation stores the value from the current top register and then increments TOP by one. The NPX register stack grows "down" toward lower-addressed registers.

Instructions may address the data registers either implicitly or explicitly. Many instructions operate on the register at the TOP of the stack. These instructions implicitly address the register at which TOP points. Other instructions allow the programmer to explicitly specify which register to use. This explicit register addressing is also relative to TOP.

2.3.2 TAG WORD

The tag word marks the content of each numeric data register, as Figure 2-1 shows. Each two-bit tag represents one of the eight data registers. The principal function of the tag word is to optimize the NPX's performance and stack handling by making it possible to distinguish between empty and nonempty register locations. It also enables exception handlers to identify special values (e.g. NaNs or denormals) in the contents of a stack location without the need to perform complex decoding of the actual data.

2.3.3 STATUS WORD

The 16-bit status word (in the status register) shown in Figure 2-2 reflects the overall state of the NPX. It may be read and inspected by programs.

Bit 15, the B-bit (busy bit) is included for 8087 compatibility only. It always has the same value as the ES bit (bit 7 of the status word); it does **not** indicate the status of the BUSY# output of NPX.

Bits 13–11 (TOP) point to the NPX register that is the current top-of-stack.

The four numeric condition code bits (C_3–C_0) are similar to the flags in a CPU; instructions that perform arithmetic operations update these bits to reflect the outcome. The effects of these instructions on the condition code are summarized in Tables 2-2 through 2-5.

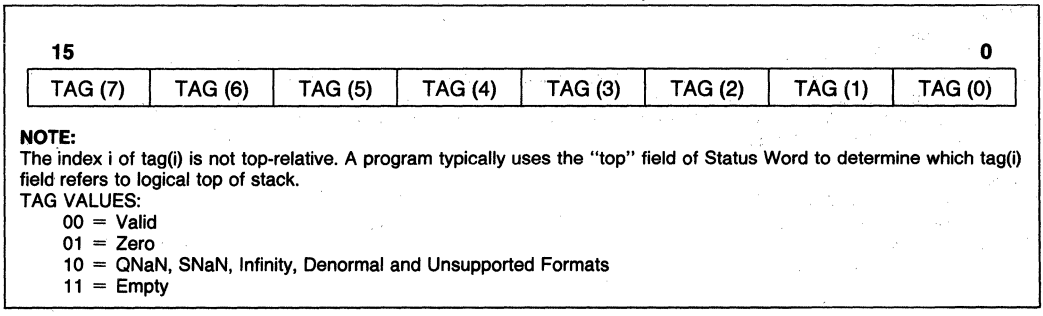
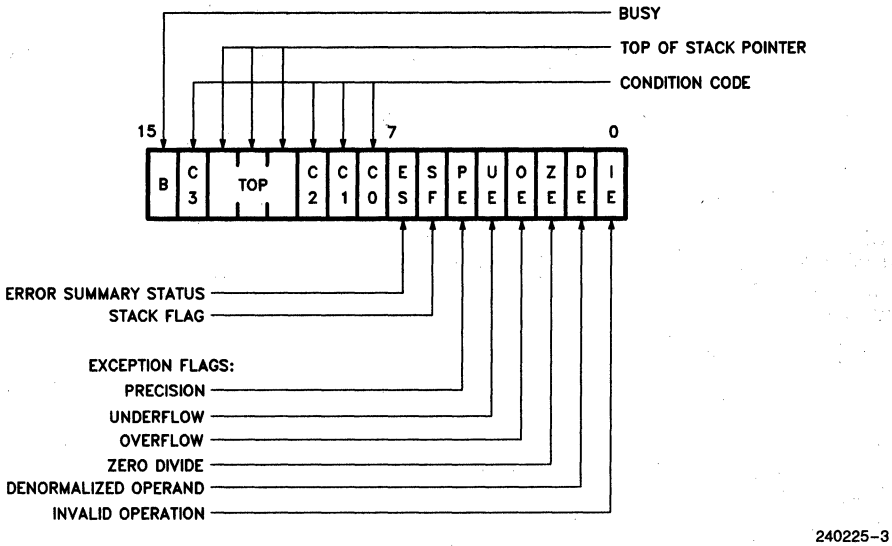


Figure 2-1. Tag Word



ES is set if any unmasked exception bit is set; cleared otherwise. See Table 2-2 for interpretation of condition code.
TOP values:
000 = Register 0 is Top of Stack
001 = Register 1 is Top of Stack
.
.
.
111 = Register 7 is Top of Stack
For definitions of exceptions, refer to the section entitled "Exception Handling"

Figure 2-2. Status Word

Bit 7 is the error summary (ES) status bit. This bit is set if any unmasked exception bit is set; it is clear otherwise. If this bit is set, the ERROR# signal is asserted.

Bit 6 is the stack flag (SF). This bit is used to distinguish invalid operations due to stack overflow or underflow from other kinds of invalid operations. When SF is set, bit 9 (C_1) distinguishes between stack overflow ($C_1 = 1$) and underflow ($C_1 = 0$).

Figure 2-2 shows the six exception flags in bits 5–0 of the status word. Bits 5–0 are set to indicate that the NPX has detected an exception while executing an instruction. A later section entitled "Exception Handling" explains how they are set and used.

Note that when a new value is loaded into the status word by the FLDENV or FRSTOR instruction, the value of ES (bit 7) and its reflection in the B-bit (bit 15) are not derived from the values loaded from memory but rather are dependent upon the values of the exception flags (bits 5–0) in the status word and their corresponding masks in the control word. If ES is set in such a case, the ERROR# output of the NPX is activated immediately.

Table 2-2. Condition Code Interpretation

Instruction	C0 (S)	C3 (Z)	C1 (A)	C2 (C)
FPREM, FPREM1 (see Table 2.3)	Three least significant bits of quotient Q2 Q0		Q1 or O/U#	Reduction 0 = complete 1 = incomplete
FCOM, FCOMP, FCOMPP, FTST, FUCOM, FUCOMP, FUCOMPP, FICOM, FICOMP	Result of comparison (see Table 2.4)		Zero or O/U#	Operand is not comparable (Table 2.4)
FXAM	Operand class (see Table 2.5)		Sign or O/U#	Operand class (Table 2.5)
FCHS, FABS, FXCH, FINCTOP, FDECTOP, Constant loads, FXTRACT, FLD, FILD, FBLD, FSTP (ext real)	UNDEFINED		Zero or O/U#	UNDEFINED
FIST, FBSTP, FRNDINT, FST, FSTP, FADD, FMUL, FDIV, FDIVR, FSUB, FSUBR, FSCALE, FSQRT, FPATAN, F2XM1, FYL2X, FYL2XP1	UNDEFINED		Roundup or O/U#	UNDEFINED
FPTAN, FSIN FCOS, FSINCOS	UNDEFINED		Roundup or O/U#, undefined if C2 = 1	Reduction 0 = complete 1 = incomplete
FLDENV, FRSTOR	Each bit loaded from memory			
FLDCW, FSTENV, FSTCW, FSTSW, FCLEX, FINIT, FSAVE	UNDEFINED			

O/U#	When both IE and SF bits of status word are set, indicating a stack exception, this bit distinguishes between stack overflow (C1 = 1) and underflow (C1 = 0).
Reduction	If FPREM or FPREM1 produces a remainder that is less than the modulus, reduction is complete. When reduction is incomplete the value at the top of the stack is a partial remainder, which can be used as input to further reduction. For FPTAN, FSIN, FCOS,and FSINCOS, the reudction bit is set if the oeprand at the top of the stack is too large. In this case the original operand remains at the top of the stack.
Roundup	When the PE bit of the status word is set, this bit indicates whether the last rounding in the instruction was upward.
UNDEFINED	Do not rely on finding any specific value in these bits.

Table 2-3. Condition Code Interpretation after FPREM and FPREM1 Instructions

Condition Code				Interpretation after FPREM and FPREM1	
C2	C3	C1	C0		
1	X	X	X	Incomplete Reduction: further interation required for complete reduction	
	Q1	Q0	Q2	Q MOD8	
0	0	0	0	0	Complete Reduction: C0, C3, C1 contain three least significant bits of quotient
	0	1	0	1	
	1	0	0	2	
	1	1	0	3	
	0	0	1	4	
	0	1	1	5	
	1	0	1	6	
	1	1	1	7	

Table 2-4. Condition Code Resulting from Comparison

Order	C3	C2	C0
TOP > Operand	0	0	0
TOP < Operand	0	0	1
TOP = Operand	1	0	0
Unordered	1	1	1

Table 2.5. Condition Code Defining Operand Class

C3	C2	C1	C0	Value at TOP
0	0	0	0	+ Unsupported
0	0	0	1	+ NaN
0	0	1	0	− Unsupported
0	0	1	1	− NaN
0	1	0	0	+ Normal
0	1	0	1	+ Infinity
0	1	1	0	− Normal
0	1	1	1	− Infinity
1	0	0	0	+ 0
1	0	0	1	+ Empty
1	0	1	0	− 0
1	0	1	1	− Empty
1	1	0	0	+ Denormal
1	1	1	0	− Denormal

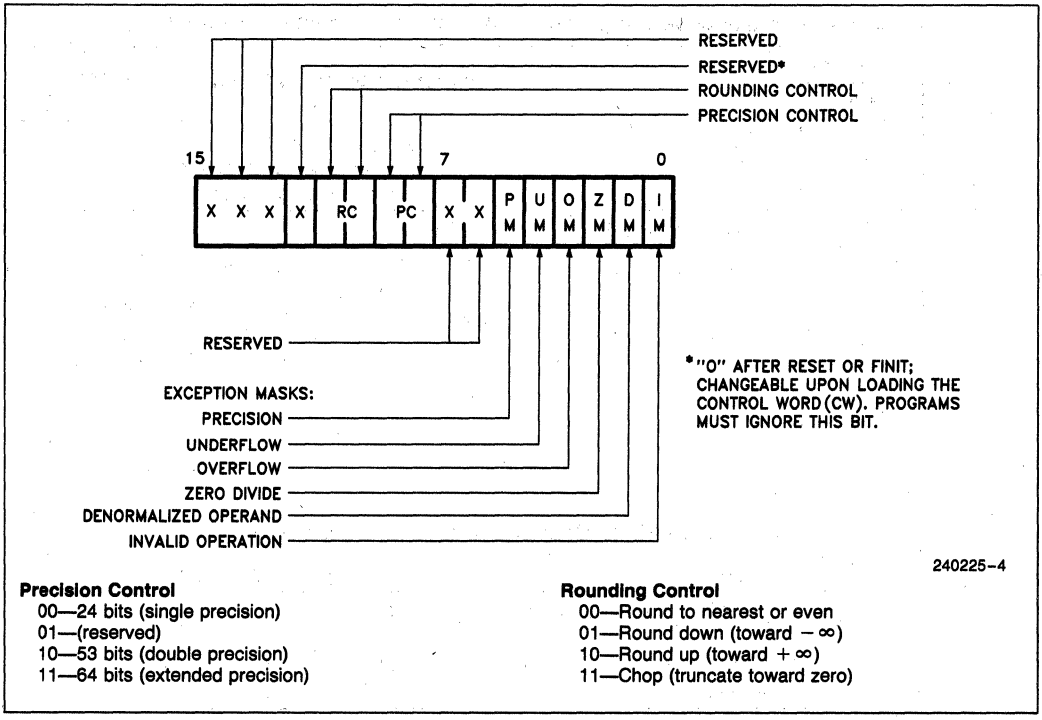


Figure 2-3. Control Word

2.3.4 CONTROL WORD

The NPX provides several processing options that are selected by loading a control word from memory into the control register. Figure 2-3 shows the format and encoding of fields in the control word.

The low-order byte of this control word configures exception masking. Bits 5–0 of the control word contain individual masks for each of the six exceptions that the NPX recognizes.

The high-order byte of the control word configures the NPX operating mode, including precision, rounding, and infinity control.

- The "infinity control bit" (bit 12) is not meaningful to the 80387SX NPX, and programs must ignore its value. To maintain compatibility with the 8087 and 80287, this bit can be programmed; however, regardless of its value, the 80387SX NPX always treats infinity in the affine sense ($-\infty < +\infty$). This bit is initialized to zero both after a hardware reset and after the FINIT instruction.
- The rounding control (RC) bits (bits 11–10) provide for directed rounding and true chop, as well as the unbiased round to nearest even mode specified in the IEEE standard. Rounding control affects only those instructions that perform rounding at the end of the operation (and thus can generate a precision exception); namely, FST, FSTP, FIST, all arithmetic instructions (except FPREM, FPREM1, FXTRACT, FABS, and FCHS), and all transcendental instructions.
- The precision control (PC) bits (bits 9–8) can be used to set the NPX internal operating precision of the significand at less than the default of 64 bits (extended precision). This can be useful in providing compatibility with early generation arithmetic processors of smaller precision. PC affects only the instructions ADD, SUB, DIV, MUL, and SQRT. For all other instructions, either the precision is determined by the opcode or extended precision is used.

2.3.5 INSTRUCTION AND DATA POINTERS

Because the NPX operates in parallel with the CPU, any exceptions detected by the NPX may be reported after the CPU has executed the ESC instruction which caused it. To allow identification of the failing numeric instruction, the 386SX Microprocessor and 80387 Coprocessor contains registers that aid in diagnosis. These registers supply the address of the failing instruction and the address of its numeric memory operand (if appropriate).

The instruction and data pointers are provided for user-written exception handlers. These registers are actually located in the CPU, but appear to be located in the NPX because they are accessed by the ESC instructions FLDENV, FSTENV, FSAVE, and FRSTOR. Whenever the CPU executes a new ESC instruction, it saves the address of the instruction (including any prefixes that may be present), the address of the operand (if present), and the opcode.

The instruction and data pointers appear in one of four formats depending on the operating mode of the CPU (protected mode or real-address mode) and depending on the operand-size attribute in effect (32-bit operand or 16-bit operand). (See Figures 2-4, 2-5, 2-6, and 2-7.) The ESC instructions FLDENV, FSTENV, FSAVE, and FRSTOR are used to transfer these values between the registers and memory. Note that the value of the data pointer is *undefined* if the prior ESC instruction did not have a memory operand.

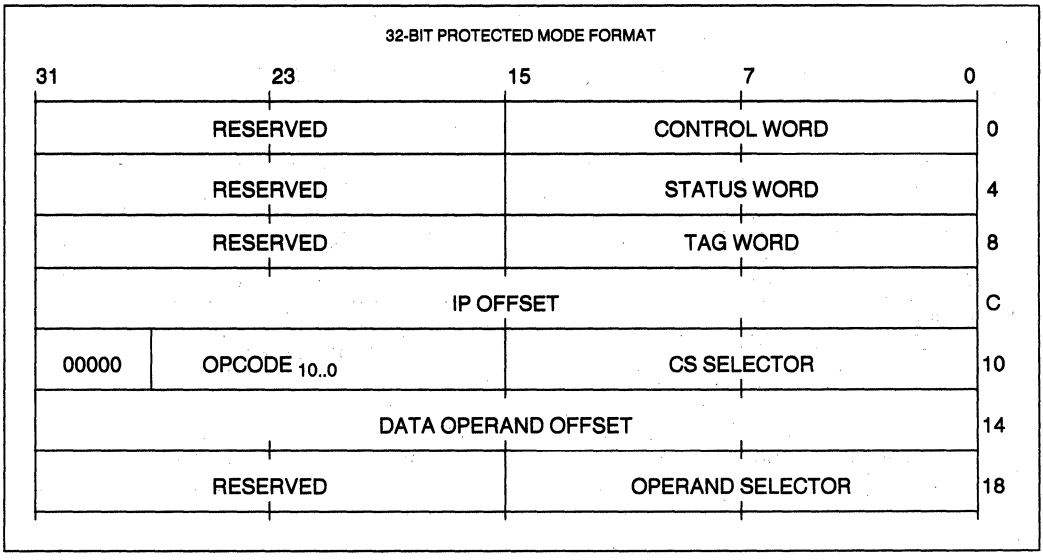


Figure 2-4.Instruction and Data Pointer Image in Memory, 32-bit Protected-Mode Format

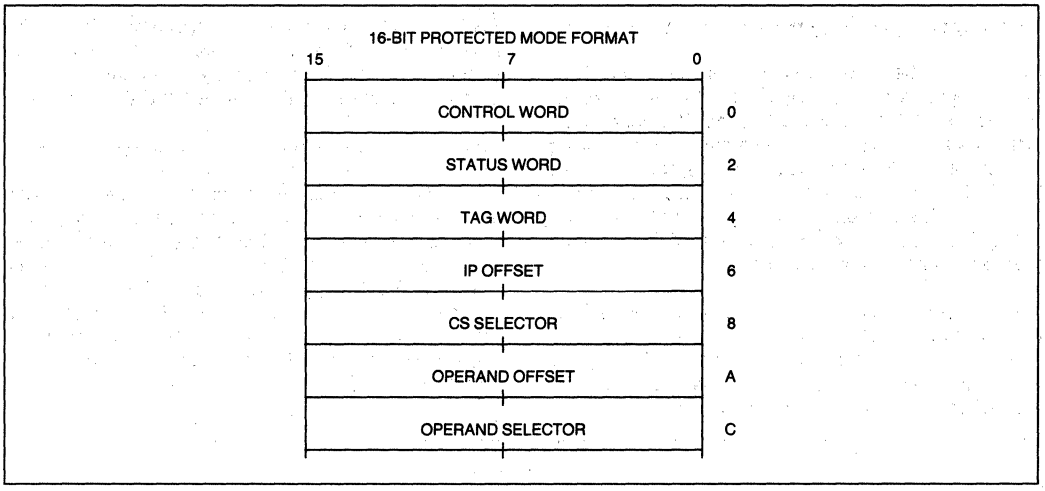


Figure 2-5. Instruction and Data Pointer Image in Memory, 16-bit Protected-Mode Format

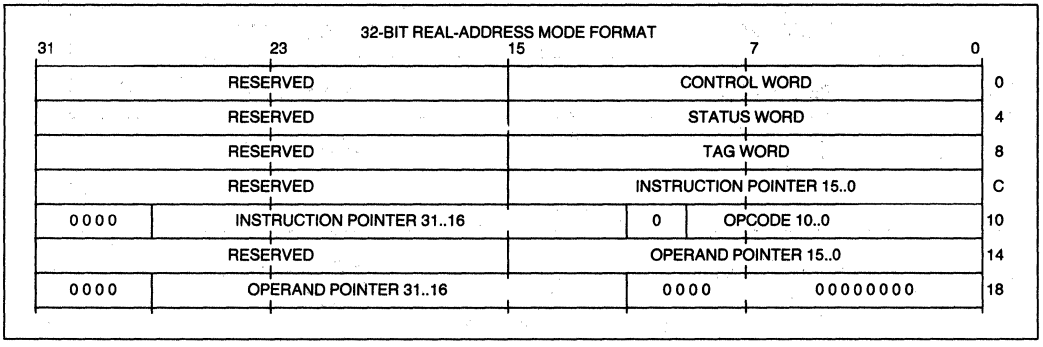


Figure 2-6. Instruction and Data Pointer Image in Memory, 32-bit Real-Mode Format

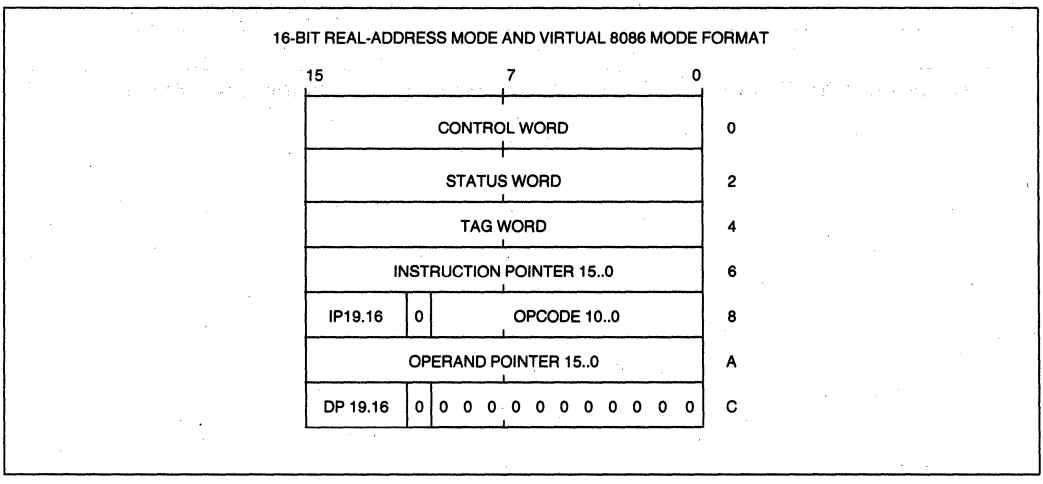


Figure 2-7. Instruction and Data Pointer Image in Memory, 16-bit Real-Mode Format

Table 2-6. CPU Interrupt Vectors Reserved for NPX

Interrupt Number	Cause of Interrupt
7	An ESC instruction was encountered when EM or TS of CPU control register zero (CR0) was set. EM = 1 indicates that software emulation of the instruction is required. When TS is set, either an ESC or WAIT instruction causes interrupt 7. This indicates that the current NPX context may not belong to the current task.
9	In a protected-mode system, an operand of a coprocessor instruction wrapped around an addressing limit (0FFFFH for expand-up segments, zero for expand-down segments) and spanned inaccessible addresses[a]. The failing numerics instruction is not restartable. The address of the failing numerics instruction and data operand may be lost; an FSTENV does not return reliable addresses. The segment overrun exception should be handled by executing an FNINIT instruction (i.e. an FINIT without a preceding WAIT). The exception can be avoided by never allowing numerics operands to cross the end of a segment.
13	In a protected-mode system, the first word of a numeric operand is not entirely within the limit of its segment. The return address pushed onto the stack of the exception handler points at the ESC instruction that caused the exception, including any prefixes. The NPX has not executed this instruction; the instruction pointer and data pointer register refer to a previous, correctly executed instruction.
16	The previous numerics instruction caused an unmasked exception. The address of the faulty instruction and the address of its operand are stored in the instruction pointer and data pointer registers. Only ESC and WAIT instructions can cause this interrupt. The CPU return address pushed onto the stack of the exception handler points to a WAIT or ESC instruction (including prefixes). This instruction can be restarted after clearing the exception condition in the NPX. FNINIT, FNCLEX, FNSTSW, FNSTENV, and FNSAVE cannot cause this interrupt.

a. An operand may wrap around an addressing limit when the segment limit is near an addressing limit and the operand is near the largest valid address in the segment. Because of the wrap-around, the beginning and ending addresses of such an operand will be at opposite ends of the segment. There are two ways that such an operand may also span inaccessible addresses: 1) if the segment limit is not equal to the addressing limit (e.g. addressing limit is FFFFH and segment limit is FFFDH) the operand will span addresses that are not within the segment (e.g. an 8-byte operand that starts at valid offset FFFCH will span addresses FFFC–FFFFH and 0000-0003H; however addresses FFFEH and FFFFH are not valid, because they exceed the limit); 2) if the operand begins and ends in present and accessible segments but intermediate bytes of the operand fall in a not-present page or in a segment or page to which the procedure does not have access rights.

2.4 Interrupt Description

CPU interrupts are used to report exceptional conditions while executing numeric programs in either real or protected mode. Table 2-6 shows these interrupts and their functions.

2.5 Exception Handling

The NPX detects six different exception conditions that can occur during instruction execution. Table 2-7 lists the exception conditions in order of precedence, showing for each the cause and the default action taken by the NPX if the exception is masked by its corresponding mask bit in the control word.

Any exception that is not masked by the control word sets the corresponding exception flag of the status word, sets the ES bit of the status word, and asserts the ERROR# signal. When the CPU attempts to execute another ESC instruction or WAIT, exception 16 occurs. The exception condition must be resolved via an interrupt service routine. The return address pushed onto the CPU stack upon entry to the service routine does not necessarily point to the failing instruction nor to the following instruction. The CPU saves the address of the floating-point instruction that caused the exception and the address of any memory operand required by that instruction.

2.6 Initialization

After FNINIT or RESET, the control word contains the value 037FH (all exceptions masked, precision control 64 bits, rounding to nearest) the same values as in an 80287 after RESET. For compatibility with the 8087 and 80287, the bit that used to indicate infinity control (bit 12) is set to zero; however, regardless of its setting, infinity is treated in the affine sense. After FNINIT or RESET, the status word is initialized as follows:

- All exceptions are set to zero.
- Stack TOP is zero, so that after the first push the stack top will be register seven (111B).
- The condition code C_3–C_0 is **undefined**.
- The B-bit is zero.

Table 2-7. Exceptions

Exception	Cause	Default Action (if exception is masked)
Invalid Operation	Operation on a signalling NaN, unsupported format, indeterminate for (0-∞, 0/0, (+∞) + (−∞), etc.), or stack overflow/underflow (SF is also set)	Result is a quiet NaN, integer indefinite, or BCD indefinte
Denormalized Operand	At least one of the operands is denormalized, i.e., it has the smallest exponent but a nonzero significand.	Normal processing continues
Zero Divisor	The divisor is zero while the dividend is a noninfinite, nonzero number	Result is ∞.
Overflow	The result is too large in magnitude to fit in the specified format	Result is largest finite value or ∞
Underflow	The true result is nonzero but too small to be represented in the specified format, and, if underflow exception is masked, denormalization causes the loss of accuracy.	Result is denormalized or zero
Inexact Result (Precision	The true result is not exactly representable in the specified format (e.g. 1/3); the result is rounded according to the rounding mode.	Normal processing continues

The tag word contains FFFFH (all stack locations are empty).

The 386SX Microprocessor and 80387 Coprocessor initialization software must execute an FNINIT instruction (i.e an FINIT without a preceding WAIT) after RESET. The FNINIT is not strictly required for the 80287 software, but Intel recommends its use to help ensure upward compatibility with other processors. After a hardware RESET, the ERROR# output is asserted to indicate that an 80387SX is present. To accomplish this, the IE and ES bits of the status word are set, and the IM bit in the control word is cleared. After FNINIT, the status word and the control word have the same values as in an 80287 after RESET.

2.7 8087 and 80287 Compatibility

This section summarizes the differences between the 80387SX and the 80287. Any migration from the 8087 directly to the 80387SX must also take into account the differences between the 8087 and the 80287 as listed in Appendix A.

Many changes have been designed into the 80387SX to directly support the IEEE standard in hardware. These changes result in increased performance by eliminating the need for software that supports the standard.

2.7.1 GENERAL DIFFERENCES

The 80387SX supports only affine closure for infinity arithmetic, not projective closure.

Operands for FSCALE and FPATAN are no longer restricted in range (except for ±∞); F2XM1 and FPTAN accept a wider range of operands.

Rounding control is in effect for FLD *constant*.

Software cannot change entries of the tag word to values (other than empty) that differ from actual register contents.

After reset, FINIT, and incomplete FPREM, the 80387SX resets to zero the condition code bits C_3–C_0 of the status word.

In conformance with the IEEE standard, the 80387SX does not support the special data formats pseudozero, pseudo-NaN, pseudoinfinity, and unnormal.

The denormal exception has a different purpose on the 80387SX. A system that uses the denormal-exception handler solely to normalize the denormal operands, would better mask the denormal exception on the 80387SX. The 80387SX automatically normalizes denormal operands when the denormal exception is masked.

2.7.2 EXCEPTIONS

A number of differences exist due to changes in the IEEE standard and to functional improvements to the architecture of the 80387SX:

1. When the overflow or underflow exception is masked, the 80387SX differs from the 80287 in rounding when overflow or underflow occurs. The 80387SX produces results that are consistent with the rounding mode.
2. When the underflow exception is masked, the 80387SX sets its underflow flag only if there is also a loss of accuracy during denormalization.
3. Fewer invalid-operation exceptions due to denormal operands, because the instructions FSQRT, FDIV, FPREM, and conversions to BCD or to integer normalize denormal operands before proceeding.
4. The FSQRT, FBSTP, and FPREM instructions may cause underflow, because they support denormal operands.
5. The denormal exception can occur during the transcendental instructions and the FXTRACT instruction.
6. The denormal exception no longer takes precedence over all other exceptions.
7. When the denormal exception is masked, the 80387SX automatically normalizes denormal operands. The 8087/80287 performs unnormal arithmetic, which might produce an unnormal result.
8. When the operand is zero, the FXTRACT instruction reports a zero-divide exception and leaves $-\infty$ in ST(1).
9. The status word has a new bit (SF) that signals when invalid-operation exceptions are due to stack underflow or overflow.
10. FLD *extended precision* no longer reports denormal exceptions, because the instruction is not numeric.
11. FLD *single/double precision* when the operand is denormal converts the number to extended precision and signals the denormalized operand exception. When loading a signalling NaN, FLD *single/double precision* signals an invalid-operand exception.
12. The 80387SX only generates quiet NaNs (as on the 80287); however, the 80387SX distinguishes between quiet NaNs and signaling NaNs. Signaling NaNs trigger exceptions when they are used as operands; quiet NaNs do not (except for FCOM, FIST, and FBSTP which also raise IE for quiet NaNs).
13. When stack overflow occurs during FPTAN and overflow is masked, both ST(0) and ST(1) contain quiet NaNs. The 80287/8087 leaves the original operand in ST(1) intact.
14. When the scaling factor is $\pm\infty$, the FSCALE (ST(0), ST(1)) instruction behaves as follows (ST(0) and ST(1) contain the scaled and scaling operands respectively):
    - FSCALE(0,∞) generates the invalid operation exception.
    - FSCALE(finite, $-\infty$) generates zero with the same sign as the scaled operand.
    - FSCALE(finite, $+\infty$) generates ∞ with the same sign as the scaled operand.

    The 8087/80287 returns zero in the first case and raises the invalid-operation exception in the other cases.
15. The 80387SX returns signed infinity/zero as the unmasked response to massive overflow/underflow. The 8087 and 80287 support a limited range for the scaling factor; within this range either massive overflow/underflow do not occur or undefined results are produced.

3.0 HARDWARE INTERFACE

In the following description of hardware interface, the # symbol at the end of a signal name indicates that the active or asserted state occurs when the signal is at a low voltage. When no # is present after the signal name, the signal is asserted when at the high voltage level.

3.1 Signal Description

In the following signal descriptions, the 80387SX pins are grouped by function as shown by Table 3-1. Table 3-1 lists every pin by its identifier, gives a brief description of its function, and lists some of its characteristics (Refer to Figure 5-1 and Table 5-1 for pin configuration).

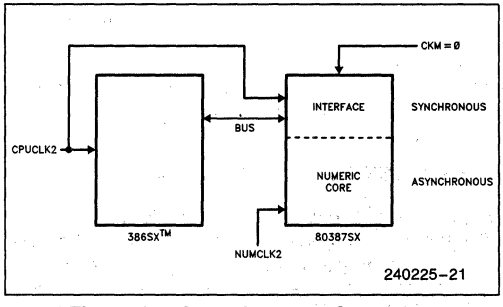


Figure 3.1. Asynchronous Operation

Table 3-1. Pin Summary

Pin Name	Function	Active State	Input/ Output	Referenced To...
Execution Control				
CPUCLK2	386SX™ Microprocessor CLocK 2		I	
NUMCLK2	80387SX CLocK 2		I	
CKM	80387SX ClocKing Mode		I	
RESETIN	System reset	High	I	CPUCLK2
NPX Handshake				
PEREQ	Processor Extension REQuest	High	O	STEN/CPUCLK2
BUSY#	Busy status	Low	O	STEN/CPUCLK2
ERROR#	Error status	Low	O	STEN/NUMCLK2
Bus Interface				
D15–D0	Data pins	High	I/O	CPUCLK2
W/R#	Write/Read bus cycle	Hi/Lo	I	CPUCLK2
ADS#	ADdress Strobe	Low	I	CPUCLK2
READY#	Bus ready input	Low	I	CPUCLK2
READYO#	Ready output	Low	O	STEN/CPUCLK2
Chip/Port Select				
STEN	STatus ENable	High	I	CPUCLK2
NPS1#	NPX select #1	Low	I	CPUCLK2
NPS2	NPX select #2	High	I	CPUCLK2
CMD0#	CoMmanD	Low	I	CPUCLK2
Power and Ground				
V_{CC}	System power			
V_{SS}	System ground			

All output signals are tristate; they leave floating state only when STEN is active. The output buffers of the bidirectional data pins D15–D0 are also tristate; they leave floating state only during cycles when the NPX is selected (i.e. when STEN, NPS1#, and NPS2 are all active).

3.1.1 386SX™ CPU CLOCK 2 (CPUCLK2)

This input uses the CLK2 signal of the CPU to time the bus control logic. Several other NPX signals are referenced to the rising edge of this signal. When CKM = 1 (synchronous mode) this pin also clocks the data interface and control unit and the floating-point unit of the NPX. This pin requires MOS-level input. The signal on this pin is divided by two to produce the internal clock signal CLK.

3.1.2 80387SX CLOCK 2 (NUMCLK2)

When CKM = 0 (asynchronous mode) this pin provides the clock for the data interface and control unit and the floating-point unit of the NPX. In this case, the ratio of the frequency of NUMCLK2 to the frequency of CPUCLK2 must lie within the range 10:16 to 14:10. When CKM = 1 (synchronous mode) signals on this pin are ignored; CPUCLK2 is used instead for the data interface and control unit and the floating-point unit. This pin requires MOS-level input.

3.1.3 CLOCKING MODE (CKM)

This pin is a strapping option. When it is strapped to V_{CC} (HIGH), the NPX operates in synchronous mode; when strapped to V_{SS} (LOW), the NPX operates in asynchronous mode. These modes relate to clocking of the data interface and control unit and the floating-point unit only; the bus control logic always operates synchronously with respect to the CPU.

3.1.4 SYSTEM RESET (RESETIN)

A LOW to HIGH transition on this pin causes the NPX to terminate its present activity and to enter a dormant state. RESETIN must remain active (HIGH) for at least 40 NUMCLK2 periods.

The HIGH to LOW transitions of RESETIN must be synchronous with CPUCLK2, so that the phase of the internal clock of the bus control logic (which is the CPUCLK2 divided by two) is the same as the phase of the internal clock of the CPU. After RESETIN goes LOW, at least 50 NUMCLK2 periods must pass before the first NPX instruction is written into the NPX. This pin should be connected to the CPU RESET pin. Table 3-1 shows the status of the output pins during the reset sequence. After a reset, all output pins return to their inactive states.

Table 3-2. Output Pin Status during Reset

Pin Value	Pin Name
HIGH	READYO#, BUSY#
LOW	PEREQ, ERROR#
Tri-State OFF	D15–D0

3.1.5 PROCESSOR EXTENSION REQUEST (PEREQ)

When active, this pin signals to the CPU that the NPX is ready for data transfer to/from its data FIFO. When all data is written to or read from the data FIFO, PEREQ is deactivated. This signal always goes inactive before BUSY# goes inactive. This signal is referenced to CPUCLK2. It should be connected to the CPU PEREQ input.

3.1.6 BUSY STATUS (BUSY#)

When active, this pin signals to the CPU that the NPX is currently executing an instruction. This signal is referenced to CPUCLK2. It should be connected to the CPU BUSY# pin.

3.1.7 ERROR STATUS (ERROR#)

This pin reflects the ES bit of the status register. When active, it indicates that an unmasked exception has occurred. This signal can be changed to inactive state only by the following instructions (without a preceding WAIT): FNINIT, FNCLEX, FNSTENV, FNSAVE, FLDCW, FLDENV, and FRSTOR. This pin is referenced to CPUCLK2. It should be connected to the ERROR# pin of the CPU.

3.1.8 DATA PINS (D15–D0)

These bidirectional pins are used to transfer data and opcodes between the CPU and NPX. They are normally connected directly to the corresponding CPU data pins. HIGH state indicates a value of one. D0 is the least significant data bit. Timings are referenced to CPUCLK2.

3.1.9 WRITE/READ BUS CYCLE (W/R#)

This signal indicates to the NPX whether the CPU bus cycle in progress is a read or a write cycle. This pin should be connected directly to the CPU's W/R# pin. HIGH indicates a write cycle; LOW a read cycle. This input is ignored if any of the signals STEN, NPS1#, or NPS2 is inactive. Setup and hold times are referenced to CPUCLK2.

3.1.10 ADDRESS STROBE (ADS#)

This input, in conjunction with the READY# input, indicates when the NPX bus-control logic may sample W/R# and the chip-select signals. Setup and hold times are referenced to CPUCLK2. This pin should be connected to the ADS# pin of the CPU.

3.1.11 BUS READY INPUT (READY#)

This input indicates to the NPX when a CPU bus cycle is to be terminated. It is used by the bus-control logic to trace bus activities. Bus cycles can be extended indefinitely until terminated by READY#. This input should be connected to the same signal that drives the CPU's READY# input. Setup and hold times are referenced to CPUCLK2.

3.1.12 READY OUTPUT (READYO#)

This pin is activated at such a time that write cycles are terminated after two clocks and read cycles after three clocks. In configurations where no extra wait states are required, this pin must directly or indirectly drive the READY# input of the CPU. Refer to the section entitled "Bus Operation" for details. This pin is activated only during bus cycles that select the NPX. This signal is referenced to CPUCLK2.

3.1.13 STATUS ENABLE (STEN)

This pin serves as a chip select for the NPX. When inactive, this pin forces, BUSY#, PEREQ#, ERROR#, and READYO# outputs into floating state. D15–D0 are normally floating; they leave floating state only if STEN is active and additional conditions are met. STEN also causes the chip to recognize its other chip-select inputs. STEN makes it easier to do on-board testing (using the overdrive method) of other chips in systems containing the NPX. STEN should be pulled up with a resistor so that it can be pulled down when testing. In boards that do not use on-board testing. STEN should be connected to V_{CC}. Setup and hold times are relative to CPUCLK2. Note that STEN must maintain the same setup and hold times as NPS1#, NPS2, and CMD0# (i.e. if STEN changes state during an NPX bus cycle, it must change state during the same CLK period as the NPS1#, NPS2, and CMD0# signals).

3.1.14 NPX SELECT 1 (NPS1#)

When active (along with STEN and NPS2) in the first period of a CPU bus cycle, this signal indicates that the purpose of the bus cycle is to communicate with the NPX. This pin should be connected directly to the M/IO# pin of the CPU, so that the NPX is selected only when the CPU performs I/O cycles. Setup and hold times are referenced to CPUCLK2.

3.1.15 NPX SELECT 2 (NPS2)

When active (along with STEN and NPS1#) in the first period of a CPU bus cycle, this signal indicates that the purpose of the bus cycle is to communicate with the NPX. This pin should be connected directly to the A23 pin of the CPU, so that the NPX is selected only when the CPU issues one of the I/O addresses reserved for the NPX (8000F8H, 8000FCH or 8000FEH which is treated as 8000FCH by the 80387SX). Setup and hold times are referenced to CPUCLK2.

3.1.16 COMMAND (CMD0#)

During a write cycle, this signal indicates whether an opcode (CMD0# active) or data (CMD0# inactive) is being sent to the NPX. During a read cycle, it indicates whether the control or status register (CMD0# active) or a data register (CMD0# inactive) is being read. CMD0# should be connected directly to the A2 output of the CPU. Setup and hold times are referenced to CPUCLK2.

3.1.17 SYSTEM POWER (V_{CC})

System power provides the +5V DC supply input. All V_{CC} pins should be tied together on the circuit board and local decoupling capacitors should be used between V_{CC} and V_{SS}.

3.1.18 SYSTEM GROUND (V_{SS})

All V_{SS} pins should be tied together on the circuit board and local decoupling capacitors should be used between V_{CC} and V_{SS}.

3.2 System Configuration

The 80387SX is designed to interface with the 386SX Microprocessor as shown by Figure 3-1. A dedicated communication protocol makes possible high-speed transfer of opcodes and operands between the CPU and NPX. The 80387SX is designed so that no additional components are required for interface with the CPU. Most control pins of the NPX are connected directly to pins of the CPU.

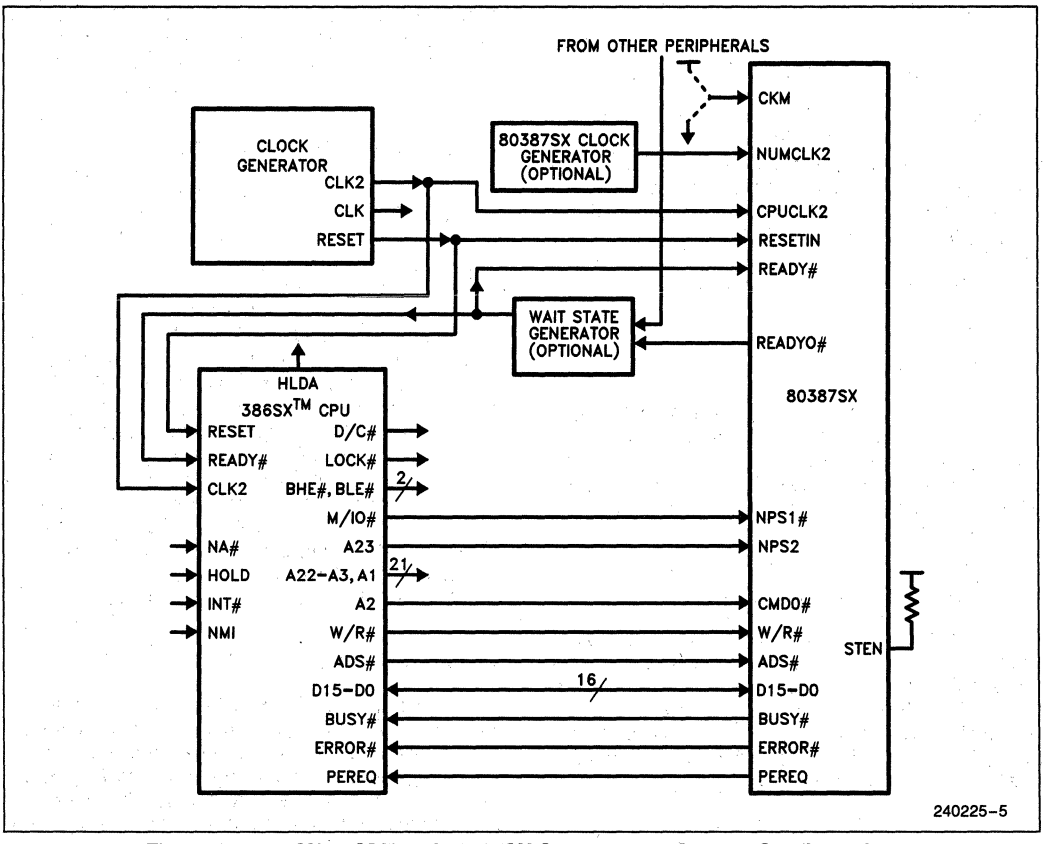


Figure 3-1. 386SX™ CPU and 80387SX Coprocessor System Configuration

The interface between the 80387SX and the CPU has these characteristics:

- The NPX shares the local bus of the 386SX Microprocessor.
- The CPU and NPX share the same reset signals. They may also share the same clock input; however, for greatest performance, an external oscillator may be needed.
- The corresponding BUSY#, ERROR#, and PEREQ pins are connected together.
- The NPX NPS1# and NPS2 inputs are connected to the latched CPU M/IO# and A23 outputs respectively. For coprocessor cycles, M/IO# is always LOW and A23 always HIGH.
- The NPX input CMD0 is connected to the latched A_2 output. The 386SX Microprocessor generates address 8000F8H when writing a command and address 8000FCH or 8000FEH (treated as 8000FCH by the 80387SX) when writing or reading data. It does not generate any other addresses during NPX bus cycles.

3.3 Processor Architecture

As shown by the block diagram on the front page, the 80387SX NPX is internally divided into three sections: the bus control logic (BCL), the data interface and control unit, and the floating point unit (FPU). The FPU (with the support of the control unit which contains the sequencer and other support units) executes all numerics instructions. The data interface and control unit is responsible for the data flow to and from the FPU and the control registers, for receiving the instructions, decoding them, and sequencing the microinstructions, and for handling some of the administrative instructions. The BCL is responsible for CPU bus tracking and interface. The BCL is the only unit in the NPX that must run synchronously with the CPU; the rest of the NPX can run asynchronously with respect to the CPU.

3.3.1 BUS CONTROL LOGIC

The BCL communicates solely with the CPU using I/O bus cycles. The BCL appears to the CPU as a special peripheral device. It is special in two respects: the CPU initiates I/O automatically when it encounters ESC instructions, and the CPU uses reserved I/O addresses to communicate with the BCL. The BCL does not communicate directly with memory. The CPU performs all memory access, transferring input operands from memory to the NPX and transferring outputs from the NPX to memory.

3.3.2 DATA INTERFACE AND CONTROL UNIT

The data interface and control unit latches the data and, subject to BCL control, directs the data to the FIFO or the instruction decoder. The instruction decoder decodes the ESC instructions sent to it by the CPU and generates controls that direct the data flow in the FIFO. It also triggers the microinstruction sequencer that controls execution of each instruction. If the ESC instruction is FINIT, FCLEX, FSTSW, FSTSW AX, FSTCW, FSETPM, or FRSTPM, the control executes it independently of the FPU and the sequencer. The data interface and control unit is the one that generates the BUSY#, PEREQ, and ERROR# signals that synchronize NPX activities with the CPU.

3.3.3 FLOATING-POINT UNIT

The FPU executes all instructions that involve the register stack, including arithmetic, logical, transcendental, constant, and data transfer instructions. The data path in the FPU is 84 bits wide (68 significant bits, 15 exponent bits, and a sign bit) which allows internal operand transfers to be performed at very high speeds.

3.4 Bus Cycles

The pins STEN, NPS1#, NPS2, CMD0, and W/R# identify bus cycles for the NPX. Table 3-3 defines the types of NPX bus cycles.

3.4.1 80387SX ADDRESSING

The NPS1#, NPS2, and CMD0 signals allow the NPX to identify which bus cycles are intended for the NPX. The NPX responds to I/O cycles when the I/O address is 8000F8H, 8000FCH or 8000FEH (treated

Table 3-3. Bus Cycle Definition

STEN	NPS1#	NPS2	CMD0#	W/R#	Bus Cycle Type
0	x	x	x	x	80387SX not selected and all outputs in floating state
1	1	x	x	x	80387SX not selected
1	x	0	x	x	80387SX not selected
1	0	1	0	0	CW or SW read from 80387SX
1	0	1	0	1	Opcode write to 80387SX
1	0	1	1	0	Data read from 80387SX
1	0	1	1	1	Data write to 80387SX

as 8000FCH by the 80387SX). The NPX responds to I/O cycles when bit 23 of the I/O address is set. In other words, the NPX acts as an I/O device in a reserved I/O address space.

Because A23 is used to select the 80387SX Numerics Processor Extension for data transfers, it is not possible for a program running on the CPU to address the NPX with an I/O instruction. Only ESC instructions cause the CPU to communicate with the NPX.

3.4.2 CPU/NPX SYNCHRONIZATION

The pins BUSY#, PEREQ, and ERROR# are used for various aspects of synchronization between the CPU and the NPX.

BUSY# is used to synchronize instruction transfer from the CPU to the NPX. When the NPX recognizes an ESC instruction, it asserts BUSY#. For most ESC instructions, the CPU waits for the NPX to deassert BUSY# before sending the new opcode.

The NPX uses the PEREQ pin of the CPU to signal that the NPX is ready for data transfer to or from its data FIFO. The NPX does not directly access memory; rather, the CPU provides memory access services for the NPX. (For this reason, memory access on behalf of the NPX always obeys the protection rules applicable to the current CPU mode.) Once the CPU initiates an NPX instruction that has operands, the CPU waits for PEREQ signals that indicate when the NPX is ready for operand transfer. Once all operands have been transferred (or if the instruction has no operands) the CPU continues program execution while the NPX executes the ESC instruction.

In 8086/8087 systems, WAIT instructions may be required to achieve synchronization of both commands and operands. In the 386SX Microprocessor and 80387 Coprocessor systems, however, WAIT instructions are required only for operand synchronization; namely, after NPX stores to memory (except FSTSW and FSTCW) or load from memory. (In 80286/80287 systems, WAIT is required before FLDENV and FRSTOR; with the 386SX Microprocessor and 80387 Coprocessor, WAIT is not required in these cases.) Used this way, WAIT ensures that the value has already been written or read by the NPX before the CPU reads or changes the value.

Once it has started to execute a numerics instruction and has transferred the operands from the CPU, the NPX can process the instruction in parallel with and independent of the host CPU. When the NPX detects an exception, it asserts the ERROR# signal, which causes a CPU interrupt.

3.4.3 SYNCHRONOUS OR ASYNCHRONOUS MODES

The internal logic of the 80387SX (the FPU) can operate either directly from the CPU clock (synchronous mode) or from a separate clock (asynchronous mode). The two configurations are distinguished by the CKM pin. In either case, the bus control logic (BCL) of the 80387SX is synchronized with the CPU clock. Use of asynchronous mode allows the CPU and the FPU section of the NPX to run at different speeds. In this case, the ratio of the frequency of NUMCLK2 to the frequency of CPUCLK2 must lie within the range 10:16 to 14:10. Use of synchronous mode eliminates one clock generator from the board design.

3.4.4 AUTOMATIC BUS CYCLE TERMINATION

In configurations where no extra wait states are required, READYO# can drive the CPU's READY# input. If this pin is used, it should be connected to the logic that ORs all READY outputs from peripherals on the CPU bus. READYO# is asserted by the NPX only during I/O cycles that select the NPX. Refer to Section 4.0 "Bus Operation" for details.

4.0 BUS OPERATION

With respect to bus interface, the 80387SX is fully synchronous with the CPU. Both operate at the same rate, because each generates its internal CLK signal by dividing CPUCLK2 by two. Furthermore, both internal CLK signals are in phase, because they are synchronized by the same RESETIN signal.

A bus cycle for the NPX starts when the CPU activates ADS# and drives new values on the address and cycle-definition lines. The NPX examines the address and cycle-definition lines in the same CLK period during which ADS# is activated. This CLK period is considered the first CLK of the bus cycle. During this first CLK period, the 80387SX also examines the R/W# input signal to determine whether the cycle is a read or a write cycle and examines the CMD0 input to determine whether an opcode, operand, or control/status register transfer is to occur.

The 80387SX supports both pipelined (i.e. overlapped) and nonpipelined bus cycles. A nonpipelined cycle is one for which the CPU asserts ADS# when no other NPX bus cycle is in progress. A pipelined bus cycle is one for which the CPU asserts ADS# and provides valid next-address and control signals before the prior NPX cycle terminates. The CPU may do this as early as the second CLK period after asserting ADS# for the prior cycle. Pipelining increas-

es the availability of the bus by at least one CLK period. The 80387SX supports pipelined bus cycles in order to optimize address pipelining by the CPU for memory cycles.

Bus operation is described in terms of an abstract state machine. Figure 4-1 illustrates the states and state transitions for NPX bus cycles:

- T_I is the idle state. This is the state of the bus logic after RESET, the state to which bus logic returns after every nonpipelined bus cycle, and the state to which bus logic returns after a series of pipelined cycles.
- T_{RS} is the READY#-sensitive state. Different types of bus cycles may require a minimum of one or two successive T_{RS} states. The bus logic remains in T_{RS} state until READY# is sensed, at which point the bus cycle terminates. Any number of wait states may be implemented by delaying READY#, thereby causing additional successive T_{RS} states.
- T_P is the first state for every pipelined bus cycle. This state is not used by nonpipelined cycles.

Note that the bus logic tracks bus state regardless of the values on the chip/port select pins.

The READYO# output of the NPX indicates when an NPX bus cycle may be terminated if no extra wait states are required. For all write cycles (except those for the instructions FLDENV and FRSTOR), READYO# is always asserted during the first T_{RS} state, regardless of the number of wait states. For all read cycles and write cycles for FLDENV and FRSTOR, READYO# is always asserted in the second T_{RS} state, regardless of the number of wait states. These rules apply to both pipelined and nonpipelined cycles. Systems designers may use READYO# in one of the following ways:

1. Connect it (directly or through logic that ORs READY# signals from other devices) to the READY# inputs of the CPU and NPX.
2. Use it as one input to a wait-state generator.

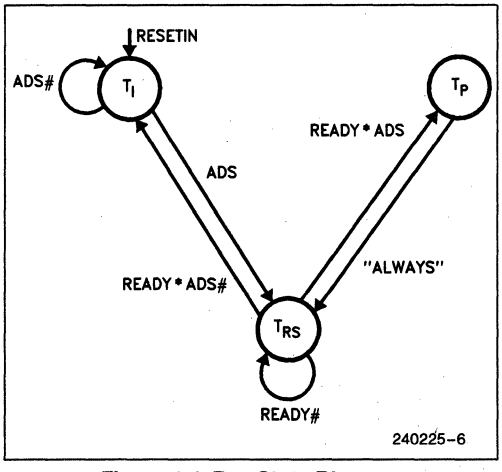


Figure 4-1. Bus State Diagram

The following sections illustrate different types of 80387SX bus cycles. Because different instructions have different amounts of overhead before, between, and after operand transfer cycles, it is not possible to represent in a few diagrams all of the combinations of successive operand transfer cycles. The following bus-cycle diagrams show memory cycles between NPX operand-transfer cycles. Note however that, during FRSTOR, some consecutive accesses to the NPX do not have intervening memory accesses. For the timing relationship between operand transfer cycles and opcode write or other overhead activities, see the figure "Other Parameters" in section 6.

4.1 Nonpipelined Bus Cycles

Figure 4-2 illustrates bus activity for consecutive nonpipelined bus cycles.

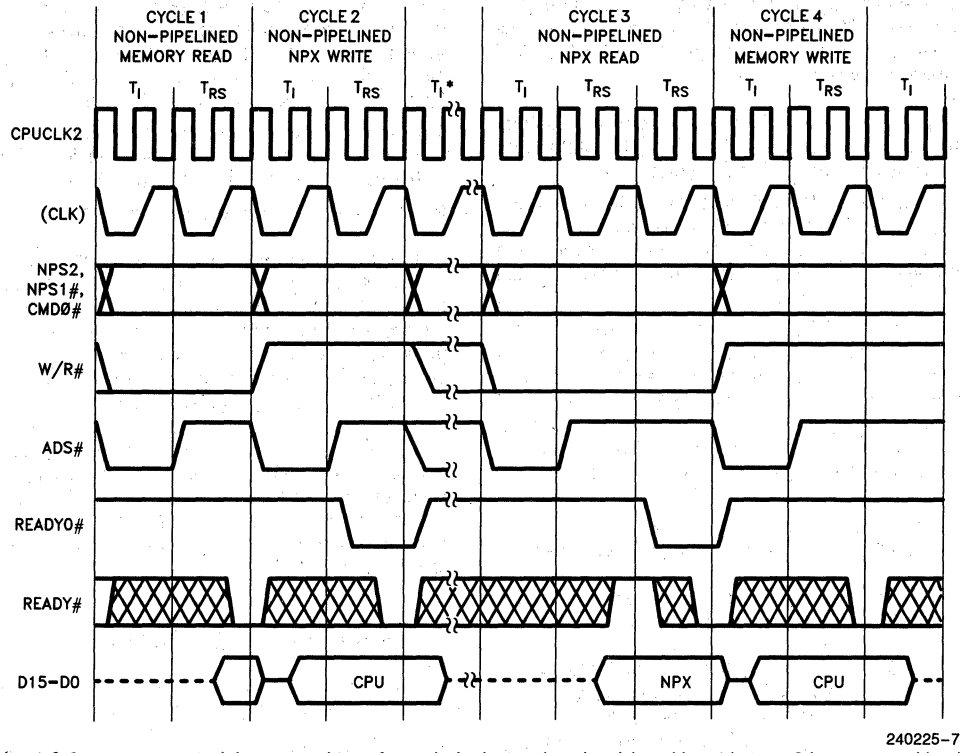
At the second clock of the bus cycle, the NPX enters the T_{RS} state. During this state, it samples the READY# input and stays in this state as long as READY# is inactive.

4.1.1 WRITE CYCLE

In write cycles, the NPX drives the READYO# signal for one CLK period during the second CLK period of the cycle (i.e. the first T_{RS} state); therefore, the fastest write cycle takes two CLK periods (see cycle 2 of Figure 4-2). For the instructions FLDENV and FRSTOR, however, the 80387SX forces a wait state by delaying the activation of READYO# to the second T_{RS} state (not shown in Figure 4-2).

The NPX samples the D15–D0 inputs into data latches at the falling edge of CLK as long as it stays in T_{RS} state.

When READY# is asserted, the NPX returns to the idle state. Simultaneously with the NPX's entering the idle state, the CPU may assert ADS# again, signaling the beginning of yet another cycle.



Cycles 1 & 2 represent part of the operand transfer cycle for instructions involving either 4-byte or 8-byte operand loads.
Cycles 3 & 4 represent part of the operand transfer cycle for a store operation.
*Cycles 1 & 2 could repeat here or T_I states for various non-operand transfer cycles and overhead.

Figure 4-2. Nonpipelined Read and Write Cycles

4.1.2 READ CYCLE

At the rising edge of CLK in the second CLK period of the cycle (i.e. the first T_{RS} state), the NPX starts to drive the D15–D0 outputs and continues to drive them as long as it stays in T_{RS} state.

At least one wait state must be inserted to ensure that the CPU latches the correct data. Because the NPX starts driving the data bus only at the rising edge of CLK in the second clock period of the bus cycle, not enough time is left for the data signals to propagate and be latched by the CPU before the next falling edge of CLK. Therefore, the 80387SX does not drive the READYO# signal until the third CLK period of the cycle. Thus, if the READYO# output drives the CPU's READY# input, one wait state is automatically inserted.

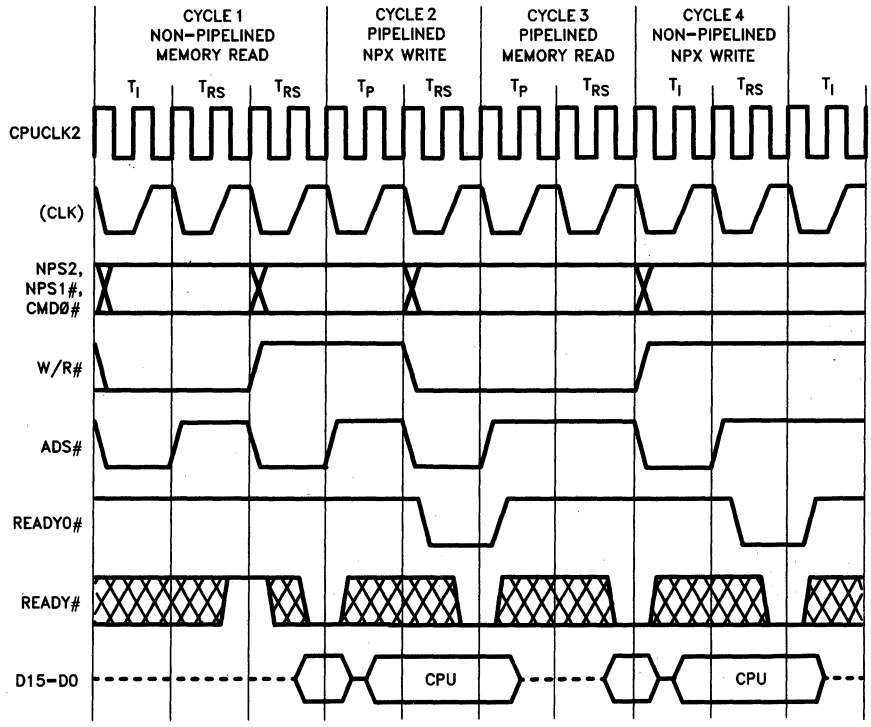
Because one wait state is required for NPX reads, the minimum length of an NPX read cycle is three CLK periods, as cycle 3 of Figure 4-2 shows.

When READY# is asserted, the NPX returns to the idle state. Simultaneously with the NPX's entering the idle state, the CPU may assert ADS# again, signaling the beginning of yet another cycle. The transition from T_{RS} state to idle state causes the NPX to put the tristate D15–D0 outputs into the floating state, allowing another device to drive the data bus.

4.2 Pipelined Bus Cycles

Because all the activities of the 80387SX bus interface occur either during the T_{RS} state or during the transitions to or from that state, the only difference between a pipelined and a nonpipelined cycle is the manner of changing from one state to another. The exact activities during each state are detailed in the previous section "Nonpipelined Bus Cycles".

When the CPU asserts ADS# before the end of a bus cycle, both ADS# and READY# are active dur-



Cycle 1–Cycle 4 represent the operand transfer cycle for an instruction involving a transfer of two 32-bit loads in total. The opcode write cycles and other overhead are not shown.
Note that the next cycle will be a pipelined cycle if both READY# and ADS# are sampled active at the end of a T_{RS} state of the current cycle.

Figure 4-3. Fastest Transitions to and from Pipelined Cycles

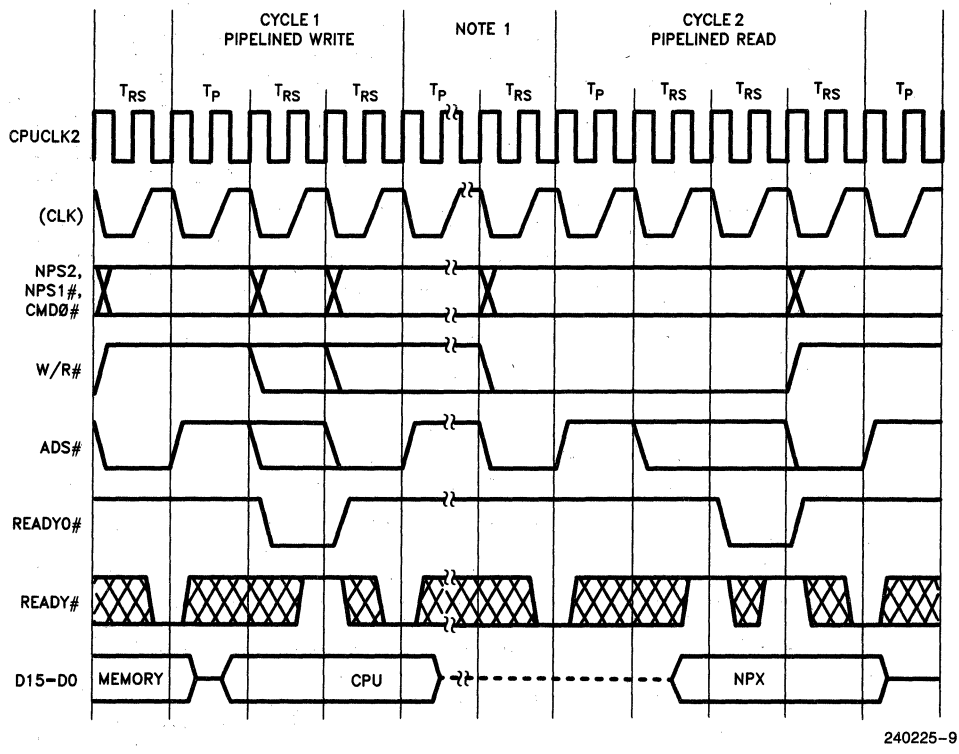
ing a T_{RS} state. This condition causes the NPX to change to a different state named T_P. One clock period after a T_P state, the NPX always returns to T_{RS} state. In consecutive pipelined cycles, the NPX bus logic uses only the T_{RS} and T_P states.

Figure 4-3 shows the fastest transitions into and out of the pipelined bus cycles. Cycle 1 in the figure represents a nonpipelined cycle. (Nonpipelined write cycles with only one T_{RS} state (i.e. no wait states) are always followed by another nonpipelined cycle, because READY# is asserted before the earliest possible assertion of ADS# for the next cycle.)

Figure 4-4 shows pipelined write and read cycles with one additional T_{RS} state beyond the minimum required. To delay the assertion of READY# requires external logic.

4.3 Bus Cycles of Mixed Type

When the NPX bus logic is in the T_{RS} state, it distinguishes between nonpipelined and pipelined cycles according to the behavior of ADS# and READY#. In a nonpipelined cycle, only READY# is activated, and the transition is from T_{RS} state to idle state. In a



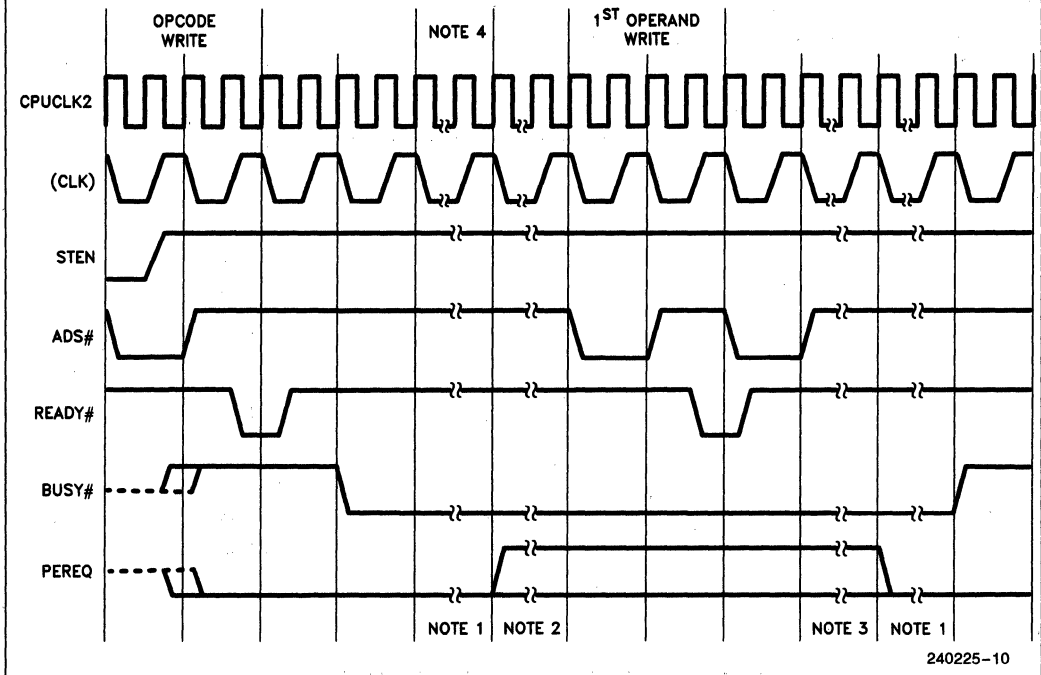
NOTE:
1. Cycles between operand write to the NPX and storing result.

Figure 4-4. Pipelined Cycles with Wait States

pipelined cycle, both READY# and ADS# are active, and the transition is first from T_{RS} state to T_P state, then, after one clock period, back to T_{RS} state.

4.4 BUSY# and PEREQ Timing Relationship

Figure 4-5 shows the activation of BUSY# at the beginning of instruction execution and its deactivation upon completion of the instruction. PEREQ is activated within this interval. If ERROR# (not shown in the figure) is ever asserted, it would be asserted at least six CPUCLK2 periods after the deactivation of PEREQ and would be deasserted at least six CPUCLK2 periods before the deactivation of BUSY#. Figure 4-5 also shows that STEN is activated at the beginning of an NPX bus cycle.

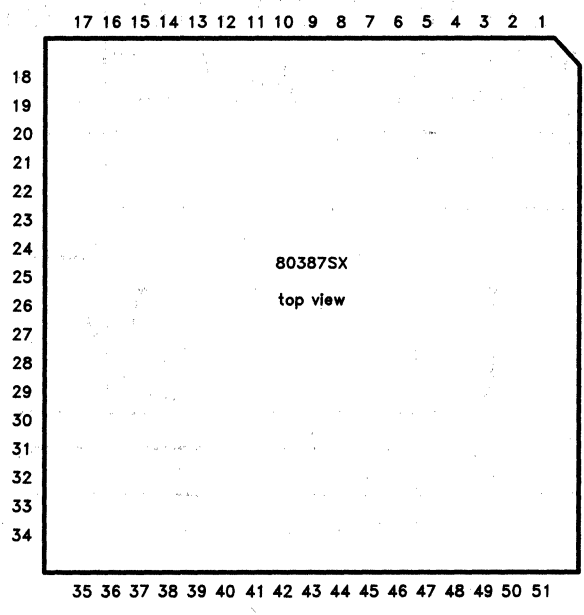


NOTES:
1. Instruction dependent.
2. PEREQ is an asynchronous input to the 386™ Microprocessor; it may not be asserted (instruction dependent).
3. More operand transfers.
4. Memory read (operand) cycle is not shown.

Figure 4-5. STEN, BUSY#, and PEREQ Timing Relationships

5.0 MECHANICAL DATA

Figure 5-1 shows the locations of pins on the chip package. Table 5-1 helps to locate pin identifiers in Figure 5-1.



The term "top view" means "as viewed when mounted in a printed-circuit board".

Figure 5-1. PLCC Pin Configuration

Table 5-1. Pin Cross-Reference

1 — n.c.	18 — n.c.	35 — ERROR#	52 — n.c.
2 — D07	19 — D00	36 — BUSY#	53 — NUMCLK2
3 — D06	20 — D01	37 — V_{CC}	54 — CPUCLK2
4 — V_{CC}	21 — V_{SS}	38 — V_{SS}	55 — V_{SS}
5 — V_{SS}	22 — V_{CC}	39 — V_{CC}	56 — PEREQ
6 — D05	23 — D02	40 — STEN	57 — READYO#
7 — D04	24 ,— D08	41 — W/R#	58 — V_{CC}
8 — D03	25 — V_{SS}	42 — V_{SS}	59 — CKM
9 — V_{CC}	26 — V_{CC}	43 — V_{CC}	60 — V_{SS}
10 — n.c.	27 — V_{SS}	44 — NPS1#	61 — V_{SS}
11 — D15	28 — D09	45 — NPS2	62 — V_{CC}
12 — D14	29 — D10	46 — V_{CC}	63 — V_{SS}
13 — V_{CC}	30 — D11	47 — ADS#	64 — V_{CC}
14 — V_{SS}	31 — V_{CC}	48 — CMD0#	65 — n.c.
15 — D13	32 — V_{SS}	49 — READY#	66 — V_{SS}
16 — D12	33 — V_{CC}	50 — V_{CC}	67 — n.c.
17 — n.c.	34 — V_{SS}	51 — RESETIN	68 — n.c.

n.c.—The corresponding pins of the 80387SX are left unconnected.

6.0 ELECTRICAL DATA

6.1 Absolute Maximum Ratings

NOTE:
Stresses above those listed may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Case temperature T_C under bias .....0°C to 85°C
Storage temperature .....−65°C to +150°C
Voltage on any pin with respect to ground −0.5 to V_{CC}+0.5V
Power dissipation .....1.5 Watt

6.2 D.C. Characteristics

Table 6-1. D.C. Specifications T_C = 0° to 85°C, V_{CC} = 5V ± 10%

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input LO Voltage	−0.3	+0.8	V	See note 1
V_{IH}	Input HI Voltage	2.0	V_{CC}+0.3	V	See note 1
V_{CL}	CPUCLK2 and NUMCLK2 Input LO Voltage	−0.3	+0.8	V	
V_{CH}	CPUCLK2 and NUMCLK2 Input HI Voltage	V_{CC}−0.8	V_{CC}+0.3	V	
V_{OL}	Output LO Voltage		0.45	V	See note 2
V_{OH}	Output HI Voltage	2.4		V	See note 3
V_{OH}	Output HI Voltage	V_{CC}−0.8		V	See note 4
I_{CC}	Power Supply Current NUMCLK = 32 MHz(5)		250	mA	I_{CC} typ. = 150 mA
I_{LI}	Input Leakage Current		±15	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{LO}	I/O Leakage Current		±15	μA	0.45V ≤ V_O ≤ V_{CC}
C_{IN}	Input Capacitance		10	pF	fc = 1MHz
C_O	I/O or Output Capacitance		12	pF	fc = 1MHz
C_{CLK}	Clock Capacitance		20	pF	fc = 1MHz

NOTES:
1. This parameter is for all inputs, excluding the clock inputs.
2. This parameter is measured at I_{OL} as follows:
   data = 4.0mA
   READY0#, ERROR#, BUSY#, PEREQ = 2.5mA
3. This parameter is measured at I_{OH} as follows:
   data = 1.0mA
   READY0#, ERROR#, BUSY#, PEREQ = 0.6mA
4. This parameter is measured at I_{OH} as follows:
   data = 0.2mA
   READY0#, ERROR#, BUSY#, PEREQ = 0.12mA
5. I_{CC} is measured at steady state, maximum capacitive loading on the outputs, and worst-case D.C. level at the inputs; CPUCLK2 at the same frequency as NUMCLK2.

6.3 A.C. Characteristics

Table 6-2a. Combinations of Bus Interface and Execution Speeds

Functional Block	80387SX-16
Bus Interface Unit (MHz)	16
Execution Unit (MHz)	16

Table 6-2b. Timing Requirements of Execution Unit T_C = 0° to 85° C, V_{CC} = 5V ± 10%

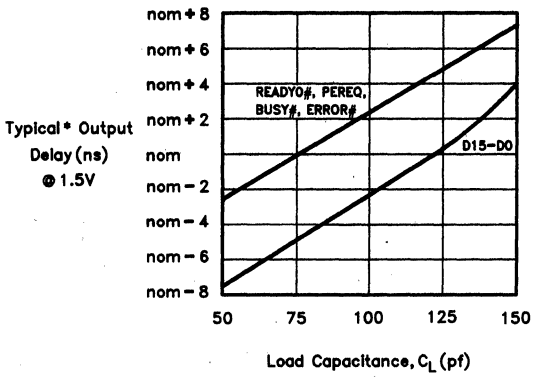
Pin	Symbol	Parameter	16 MHz		Test Conditions	Refer to Figure
			Min (ns)	Max (ns)		
NUMCLK2	T1	Period	31.25	125	2.0V	6.2
NUMCLK2	t2b	High Time	5		V_{CC} − 0.8V	
NUMCLK2	t3b	Low Time	7		0.8V	
NUMCLK2	t4	Fall Time		8	From V_{CC} − 0.8 to 0.8V	(Note 1)
NUMCLK2	t5	Rise Time		8	From 0.8 to V_{CC} − 0.8V	

Note:
1. If not used (CKM = 1), tie LOW.

Table 6-2c. Timing Requirements of Bus Interface Unit T_C = 0° to 85° C, V_{CC} = 5V ± 10%

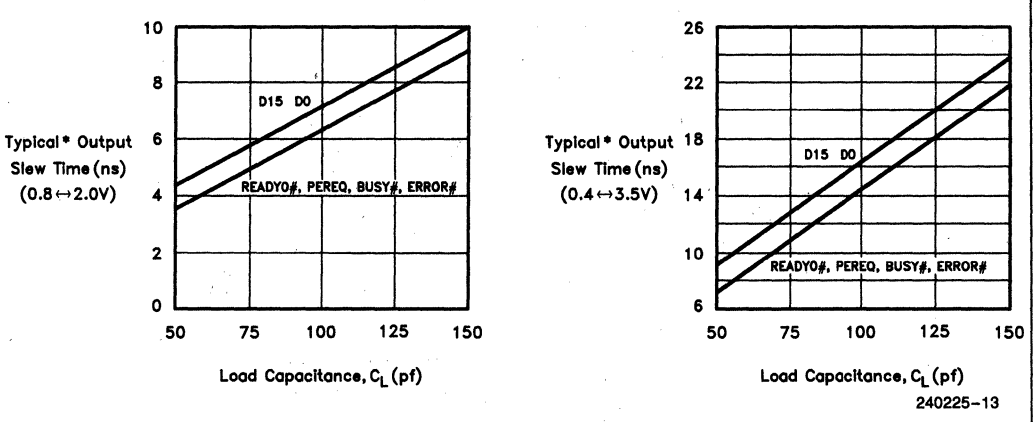
Pin	Symbol	Parameter	16 MHz (1.5V)		Test Conditions	Refer to Figure
			Min (ns)	Max (ns)		
CPUCLK2	t1	Period	31.25	125	2.0V	6.2
CPUCLK2	t2b	High Time	5		V_{CC} − 0.8V	
CPUCLK2	t3b	Low Time	7		0.8V	
CPUCLK2	t4	Fall Time		8	From V_{CC} − 0.8 to 0.8V	
CPUCLK2	t5	Rise Time		8	From 0.8 to V_{CC} − 0.8V	
CPUCLK2/ NUMCLK2		Ratio	10/16	16/10		
READYO#	t7	Out Delay	4	34	C_L = 75pf	6.3
READYO#	t7	Out Delay	4	31	C_L = 25pf**	
PEREQ	t7	Out Delay	5	34	C_L = 75pf	
BUSY#	t7	Out Delay	5	34	C_L = 75pf	
ERROR#	t7	Out Delay	5	34	C_L = 75pf	
D15–D0	T10	Out Delay	1	34	C_L = 120pf	6.4
D15–D0	t10	Setup Time	11			
D15–D0	t11	Hold Time	11			
D15–D0	t12*	Float Time	6	33	C_L = 120pf	
PEREQ	t13*	Float Time	1	60	C_L = 75pf	6.6
BUSY#	t13*	Float Time	1	60	C_L = 75 pf	
ERROR#	t13*	Float Time	1	60	C_L = 75 pf	
READY0#	t13*	Float Time	1	60	C_L = 75pf	
ADS#	t14	Setup Time	26			6.4
ADS#	t15	Hold Time	5			
W/R#	t14	Setup Time	26			
W/R#	t15	Hold Time	5			
READY#	t16	Setup Time	19			6.4
READY#	t17	Hold Time	4			
CMD0#	t16	Setup Time	21			
CMD0#	T17	Hold Time	2			
NPS1#, NPS2	t16	Setup Time	21			
NPS1#, NPS2	t17	Hold Time	2			
STEN	t16	Setup Time	21			
STEN	t17	Hold Time	2			
RESETIN	t18	Setup Time	13			6.5
RESETIN	T19	Hold Time	4			

NOTES:
*Float condition occurs when maximum output current becomes less the I_{LO} in magnitude. Float delay is not test.
**Not tested at 25 pf.



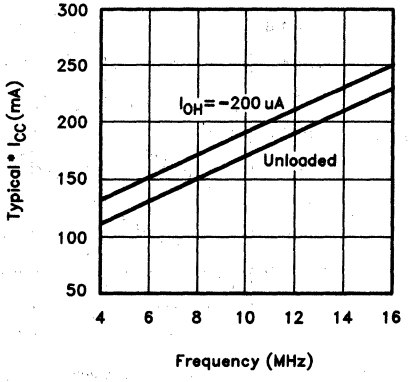
NOTES:
Graphs are not linear outside the C_L range shown.
nom = nominal value given in the AC timing table
*Typical part under worst-case conditions

Figure 6-1a. Typical Output Valid Delay vs. Load Capacitance at Max Operating Temperature



NOTES:
Graphs are not linear outside the C_L range shown.
*Typical part under worst-case conditions

Figure 6-1b. Typical Output Slew Time vs. Load Capacitance at Max Operating Temperature



NOTES:
Graphs are not linear outside the frequency range shown.
*Typical part under worst-case conditions.

Figure 6-1c. Typical I_{CC} vs. Load Capacitance at Max Operating Temperature

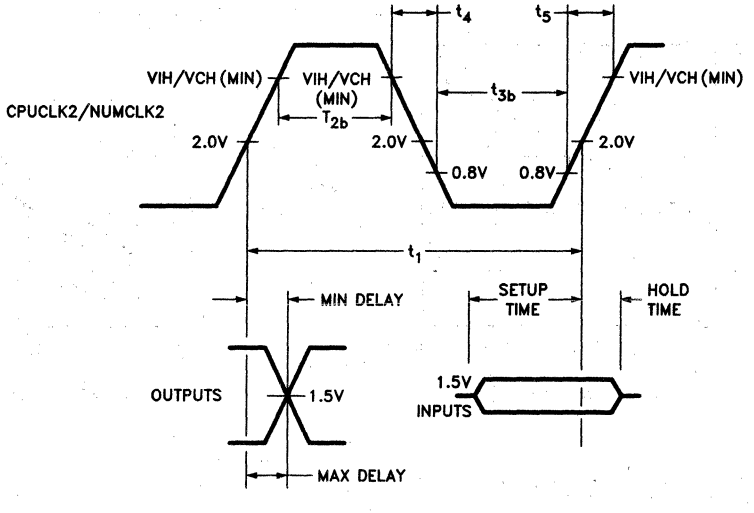


Figure 6-2. CPUCLK2/NUMCLK2 Waveform and Measurement Points for Input/Output

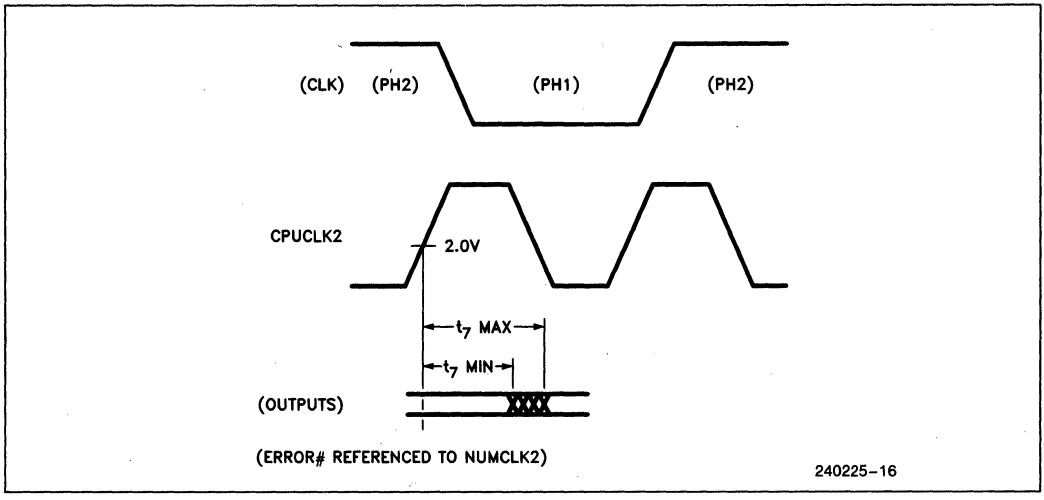


Figure 6-3. Output Signals

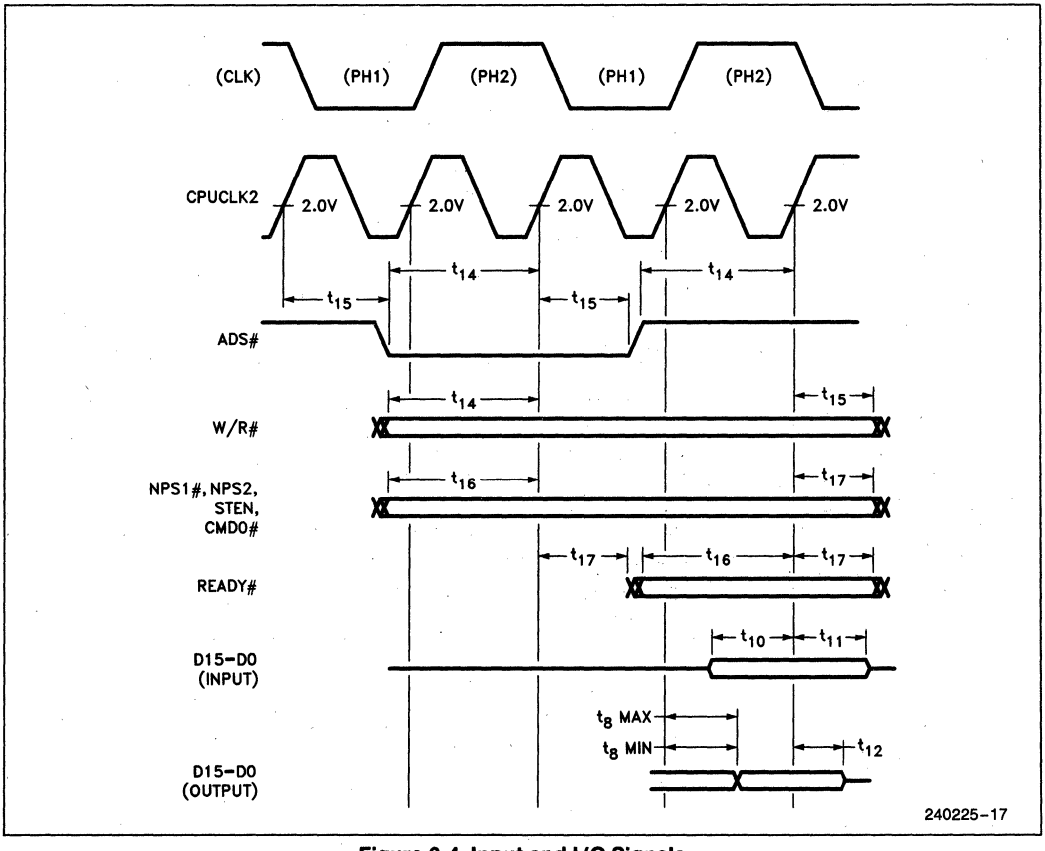
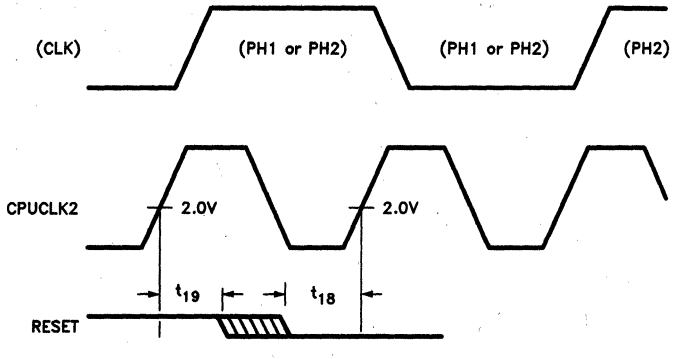


Figure 6-4. Input and I/O Signals



NOTE:
The second internal processor phase following RESET high to low transition is PH2.

Figure 6-5. RESET Signal

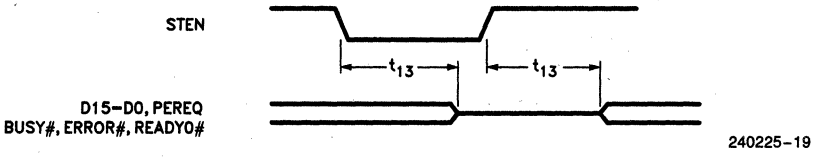
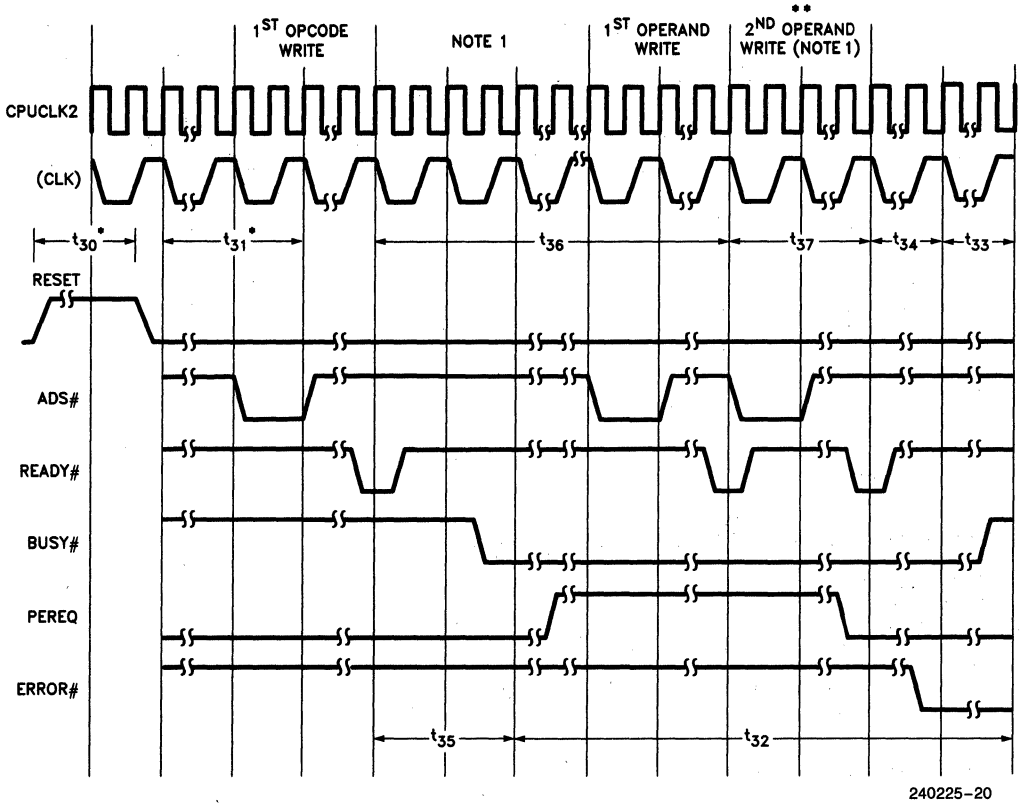


Figure 6-6. Float from STEN

Table 6-3. Other Parameters

Pin	Symbol	Parameter	Min	Max	Units
RESETIN	t30	Duration	40		NUMCLK2
RESETIN	t31	RESETIN inactive to 1st opcode write	50		NUMCLK2
BUSY#	t32	Duration	6		CPUCLK2
BUSY#, ERROR#	t33	ERROR# (in)active to BUSY# inactive	6		CPUCLK2
PEREQ, ERROR#	t34	PEREQ inactive to ERROR# active	6		CPUCLK2
READY#, BUSY#	t35	READY# active to BUSY# active	4	4	CPUCLK2
READY#	t36	Minimum time from opcode write to opcode/operand write	4		CPUCLK2
READY#	t37	Minimum time from operand write to operand write	4		CPUCLK2



* In NUMCLK2's
** or last operand

NOTE:
1. Memory read (operand) cycle is not shown.

Figure 6-7. Other Parameters

7.0 80387SX EXTENSIONS TO THE CPU'S INSTRUCTION SET

Instructions for the 80387SX assume one of the five forms shown in Table 7-1. In all cases, instructions are at least two bytes long and begin with the bit pattern 11011B, which identifies the ESCAPE class of instruction. Instructions that refer to memory operands specify addresses using the CPU's addressing modes.

MOD (Mode field) and R/M (Register/Memory specifier) have the same interpretation as the corresponding fields of CPU instructions (refer to Programmer's Reference Manual for the CPU). SIB (Scale Index Base) byte and DISP (displacement) are optionally present in instructions that have MOD and R/M fields. Their presence depends on the values of MOD and R/M, as for instructions of the CPU.

The instruction summaries that follow assume that the instruction has been prefetched, decoded, and is ready for execution; that bus cycles do not require wait states; that there are no local bus HOLD requests delaying processor access to the bus; and that no exceptions are detected during instruction execution. If the instruction has MOD and R/M fields that call for both base and index registers, add one clock.

Table 7-1. Instruction Formats

	Instruction									Optional Fields	
	First Byte				Second Byte						
1	11011	OPA		1	MOD		1	OPB	R/M	SIB	DISP
2	11011	MF		OPA	MOD		OPB*		R/M	SIB	DISP
3	11011	d	P	OPA	1	1	OPB*		ST(i)		
4	11011	0	0	1	1	1	1	OP			
5	11011	0	1	1	1	1	1	OP			
	15–11	10	9	8	7	6	5	4 3 2 1 0			

OP = Instruction opcode, possibly split into two fields OPA and OPB
MF = Memory Format
  00—32-bit real
  01—32-bit integer
  10—64-bit real
  11—16-bit integer
d = Destination
  0—Destinatino is ST(0)
  1—Destination is ST(i)
R XOR d = 0—Destination (op) Source
R XOR d = 1—Source (op) Destination
*In FSUB and FDIV, the low-order bit of OPB is the R (reversed) bit
P = POP
  0—Do not pop stack
  1—Pop stack after operation
ESC = 11011
ST(i) = Register stack element i
  000 = Stack top
  001 = Second stack element
  .
  .
  .
  111 = Eighth stack element

80387SX Extension to the 386™ Microprocessor Instruction Set

Instruction	Encoding			Clock Count Range			
	Byte 0	Byte 1	Optional Bytes 2–6	32-Bit Real	32-Bit Integer	64-Bit Real	16-Bit Integer
DATA TRANSFER							
FLD = Load[a]							
Integer/real memory to ST(0)	ESC MF 1	MOD 000 R/M	SIB/DISP	24	49–56	33	61–65
Long integer memory to ST(0)	ESC 111	MOD 101 R/M	SIB/DISP		64–75		
Extended real memory to ST(0)	ESC 011	MOD 101 R/M	SIB/DISP		52		
BCD memory to ST(0)	ESC 111	MOD 100 R/M	SIB/DISP		274–283		
ST(i) to ST(0)	ESC 001	11000 ST(i)			14		
FST = Store							
ST(0) to integer/real memory	ESC MF 1	MOD 010 R/M	SIB/DISP	49	84–98	55	82–95
ST(0) to ST(i)	ESC 101	11010 ST(i)			11		
FSTP = Store and Pop							
ST(0) to integer/real memory	ESC MF 1	MOD 011 R/M	SIB/DISP	49	84–98	55	82–95
ST(0) to long integer memory	ESC 111	MOD 111 R/M	SIB/DISP		90–107		
ST(0) to extended real	ESC 011	MOD 111 R/M	SIB/DISP		63		
ST(0) to BCD memory	ESC 111	MOD 110 R/M	SIB/DISP		522–544		
ST(0) to ST(i)	ESC 101	11001 ST (i)			12		
FXCH = Exchange							
ST(i) and ST(0)	ESC 001	11001 ST(i)			18		
COMPARISON							
FCOM = Compare							
Integer/real memory to ST(0)	ESC MF 0	MOD 010 R/M	SIB/DISP	30	60–67	39	71–75
ST(i) to ST(0)	ESC 000	11010 ST(i)			24		
FCOMP = Compare and pop							
Integer/real memory to ST	ESC MF 0	MOD 011 R/M	SIB/DISP	30	60–67	39	71–75
ST(i) to ST(0)	ESC 000	11011 ST(i)			26		
FCOMPP = Compare and pop twice							
ST(1) to ST(0)	ESC 110	1101 1001			26		
FTST = Test ST(0)	ESC 001	1110 0100			28		
FUCOM = Unordered compare	ESC 101	11100 ST(i)			24		
FUCOMP = Unordered compare and pop	ESC 101	11101 ST(i)			26		
FUCOMPP = Unordered compare and pop twice	ESC 010	1110 1001			26		
FXAM = Examine ST(0)	ESC 001	11100101			30-38		
CONSTANTS							
FLDZ = Load +0.0 into ST(0)	ESC 001	1110 1110			20		
FLD1 = Load +1.0 into ST(0)	ESC 001	1110 1000			24		
FLDPI = Load pi into ST(0)	ESC 001	1110 1011			40		
FLDL2T = Load $\log_2(10)$ into ST(0)	ESC 001	1110 1001			40		

Shaded areas indicate instructions not available in 8087/80287.

NOTE:
a. When loading single- or double-precision zero from memory, add 5 clocks.

80387SX Extension to the 386™ Microprocessor Instruction Set (Continued)

Instruction	Encoding			Clock Count Range			
	Byte 0	Byte 1	Optional Bytes 2-6	32-Bit Real	32-Bit Integer	64-Bit Real	16-Bit Integer
CONSTANTS (Continued)							
FLDL2E = Load $\log_2(e)$ into ST(0)	ESC 001	1110 1010		40			
FLDLG2 = Load $\log_{10}(2)$ into ST(0)	ESC 001	1110 1100		41			
FLDLN2 = Load $\log_e(2)$ into ST(0)	ESC 001	1110 1101		41			
ARITHMETIC							
FADD = Add							
Integer/real memory with ST(0)	ESC MF 0	MOD 000 R/M	SIB/DISP	28-36	61-76	37-45	71-85
ST(i) and ST(0)	ESC d P 0	11000 ST(i)		23-31[b]			
FSUB = Subtract							
Integer/real memory with ST(0)	ESC MF 0	MOD 10 R R/M	SIB/DISP	28-36	61-76	36-44	71-83[c]
ST(i) and ST(0)	ESC d P 0	1110 R R/M		26-34[d]			
FMUL = Multiply							
Integer/real memory with ST(0)	ESC MF 0	MOD 001 R/M	SIB/DISP	31-39	65-86	40-65	76-87
ST(i) and ST(0)	ESC d P 0	1100 1 R/M		29-57[e]			
FDIV = Divide							
Integer/real memory with ST(0)	ESC MF 0	MOD 11 R R/M	SIB/DISP	93	124-131[f]	102	136-140[g]
ST(i) and ST(0)	ESC d P 0	1111 R R/M		88[h]			
FSQRT[i] = Square root	ESC 001	1111 1010		122-129			
FSCALE = Scale ST(0) by ST(1)	ESC 001	1111 1101		67-86			
FPREM = Partial remainder	ESC 001	1111 1000		74-155			
FPREM1 = Partial remainder (IEEE)	ESC 001	1111 0101		95-185			
FRNDINT = Round ST(0) to integer	ESC 001	1111 1100		66-80			
FXTRACT = Extract components of ST(0)	ESC 001	1111 0100		70-76			
FABS = Absolute value of ST(0)	ESC 001	1110 0001		22			
FCHS = Change sign of ST(0)	ESC 001	1110 0000		24-25			

Shaded areas indicate instructions not available in 8087/80287.

NOTES:
b. Add 3 clocks to the range when d = 1.
c. Add 1 clock to **each** range when R = 1.
d. Add 3 clocks to the range when d = 0.
e. typical = 52 (When d = 0, 46-54, typical = 49).
f. Add 1 clock to the range when R = 1.
g. 135-141 when R = 1.
h. Add 3 clocks to the range when d = 1.
i. $-0 \leq ST(0) \leq +\infty$.

80387SX Extension to the 386™ Microprocessor Instruction Set (Continued)

Instruction	Encoding			Clock Count Range
	Byte 0	Byte 1	Optional Bytes 2-6	
TRANSCENDENTAL				
FCOS[k] = Cosine of ST(0)	ESC 001	1111 1111		123-772[j]
FPTAN[k] = Partial tangent of ST(0)	ESC 001	1111 0010		191-497[j]
FPATAN = Partial arctangent	ESC 001	1111 0011		314-487
FSIN[k] = Sine of ST(0)	ESC 001	1111 1110		122-771[j]
FSINCOS[k] = Sine and cosine of ST(0)	ESC 001	1111 1011		194-809[j]
F2XM1[l] = $2^{ST(0)} - 1$	ESC 001	1111 0000		211-476
FYL2X[m] = ST(1) * $\log_2$(ST(0))	ESC 001	1111 0001		120-538
FYL2XP1[n] = ST(1) * $\log_2$(ST(0) + 1.0)	ESC 001	1111 1001		257-547
PROCESSOR CONTROL				
FINIT = Initialize NPX	ESC 011	1110 0011		33
FSTSW AX = Store status word	ESC 111	1110 0000		13
FLDCW = Load control word	ESC 001	MOD 101 R/M	SIB/DISP	19
FSTCW = Store control word	ESC 101	MOD 111 R/M	SIB/DISP	15
FSTSW = Store status word	ESC 101	MOD 111 R/M	SIB/DISP	15
FCLEX = Clear exceptions	ESC 011	1110 0010		11
FSTENV = Store environment	ESC 001	MOD 110 R/M	SIB/DISP	103-104
FLDENV = Load environment	ESC 001	MOD 100 R/M	SIB/DISP	71
FSAVE = Save state	ESC 101	MOD 110 R/M	SIB/DISP	475-476
FRSTOR = Restore state	ESC 101	MOD 100 R/M	SIB/DISP	388
FINCSTP = Increment stack pointer	ESC 001	1111 0111		21
FDECSTP = Decrement stack pointer	ESC 001	1111 0110		22
FFREE = Free ST(i)	ESC 101	1100 0 ST(i)		18
FNOP = No operations	ESC 001	1101 0000		12

Shaded areas indicate instructions not available in 8087/80287.

NOTES:
j. These timings hold for operands in the range $|x| < \pi/4$. For operands not in this range, up to 76 additional clocks may be needed to reduce the operand.
k. $0 \leq |ST(0)| < 2^{63}$.
l. $-1.0 \leq ST(0) \leq 1.0$.
m. $0 \leq ST(0) < \infty$, $-\infty < ST(1) < +\infty$.
n. $0 \leq |ST(0)| < (2 - SQRT(2))/2$, $-\infty < ST(1) < +\infty$.

APPENDIX A
COMPATIBILITY BETWEEN THE 80287 AND THE 8087

The 80286/80287 operating in Real-Address mode will execute 8086/8087 programs without major modification. However, because of differences in the handling of numeric exceptions by the 80287 NPX and the 8087 NPX, exception-handling routines *may* need to be changed.

This appendix summarizes the differences between the 80287 NPX and the 8087 NPX, and provides details showing how 8086/8087 programs can be ported to the 80286/80287.

1. The NPX signals exceptions through a dedicated $\overline{\text{ERROR}}$ line to the 80286. The NPX error signal does not pass through an interrupt controller (the 8087 INT signal does). Therefore, any interrupt-controller-oriented instructions in numeric exception handlers for the 8086/8087 should be deleted.
2. The 8087 instructions FENI/FNENI and FDISI/FNDISI perform no useful function in the 80287. If the 80287 encounters one of these opcodes in its instruction stream, the instruction will effectively be ignored—none of the 80287 internal states will be updated. While 8086/8087 containing these instructions may be executed on the 80286/80287, it is unlikely that the exception-handling routines containing these instructions will be completely portable to the 80287.
3. Interrupt vector 16 must point to the numeric exception handling routine.
4. The ESC instruction address saved in the 80287 includes any leading prefixes before the ESC opcode. The corresponding address saved in the 8087 does not include leading prefixes.
5. In Protected-Address mode, the format of the 80287's saved instruction and address pointers is different than for the 8087. The instruction opcode is not saved in Protected mode—exception handlers will have to retrieve the opcode from memory if needed.
6. Interrupt 7 will occur in the 80286 when executing ESC instructions with either TS (task switched) or EM (emulation) of the 80286 MSW set (TS = 1 or EM = 1). If TS is set, then a WAIT instruction will also cause interrupt 7. An exception handler should be included in 80286/80287 code to handle these situations.
7. Interrupt 9 will occur if the second or subsequent words of a floating-point operand fall outside a segment's size. Interrupt 13 will occur if the starting address of a numeric operand falls outside a segment's size. An exception handler should be included in 80286/80287 code to report these programming errors.
8. Except for the processor control instructions, all of the 80287 numeric instructions are automatically synchronized by the 80286 CPU—the 80286 automatically tests the $\overline{\text{BUSY}}$ line from the 80287 to ensure that the 80287 has completed its previous instruction before executing the next ESC instruction. No explicit WAIT instructions are required to assure this synchronization. For the 8087 used with 8086 and 8088 processors, explicit WAITs are required before each numeric instruction to ensure synchronization. Although 8086/8087 programs having explicit WAIT instructions will execute perfectly on the 80286/80287 without reassembly, these WAIT instructions are unnecessary.
9. Since the 80287 does not require WAIT instructions before each numeric instruction, the ASM286 assembler does not automatically generate these WAIT instructions. The ASM86 assembler, however, automatically precedes every ESC instruction with a WAIT instruction. Although numeric routines generated using the ASM86 assembler will generally execute correctly on the 80286/80287, reassembly using ASM286 may result in a more compact code image.

   The processor control instructions for the 80287 may be coded using either a WAIT or No-WAIT form of mnemonic. The WAIT forms of these instructions cause ASM286 to precede the ESC instruction with a CPU WAIT instruction, in the identical manner as does ASM86.

DATA SHEET REVISION REVIEW

The following list represents the key differences between this and the -001 versions of the 80387SX Data Sheet. Please review this summary carefully.

1. A new Figure 3-1 was added to illustrate the 80387SX operating in an asynchronous mode.
2. Data type representation ranges were inaccurate, and are revised in Table 2-1.
3. The ratio of the CPUCLK2 frequency to the NUMCLK2 frequency must lie within the range 10:16 to 14:10 instead of the former 10:16 to 16:10 specification.

82310/11
Micro Channel* Compatible Peripheral Family

*Micro Channel, PS/2, AT are trademarks of IBM.

November 1988
Order Number: 290167-002

Micro Channel COMPATIBLE PERIPHERALS FAMILY

High Performance/High Integration/100% Compatibility

- **Total Solution ... High Integration VLSI Components Implement Complete Micro Channel Compatible Motherboards**
- **Single Architectural Solution for 80386 and 80386SX Systems**
- **High Performance**
  - **— 80386 Systems to 25 MHz**
  - **— Up to 16 MB of Zero Wait State Page-Interleaved DRAM**
  - **— Interface to Industry Standard 82385 Cache Controller for Maximum Performance Memory Design**
- **100% Compatible at All Levels**
  - **— Architecture Compatible**
  - **— Register Level Compatible**
  - **— Compatible with All Micro Channel Bus Timing and Drive Characteristics**
- **High Integration ... Two Chip Sets to Choose from, 82310 and More Highly Integrated 82311**
- **82310 Chip Set Includes:**
  - **— 82306 Local Channel Support Chip**
  - **— 82307 DMA Controller/Central Arbiter**
  - **— 82308 Micro Channel Bus Controller**
  - **— 82309 Address Bus Controller**
  - **— 82706 VGA Graphics Controller**
- **82311 Chip Set Includes:**
  - **— 82303 and 82304 Local I/O Channel Support Chips**
  - **— 82307 DMA Controller/Central Arbiter**
  - **— 82308 Micro Channel Bus Controller**
  - **— 82309 Address Bus Controller**
  - **— 82706 VGA Graphics Controller**
  - **— 82077 Floppy Disk Controller**

Intel's Micro Channel Peripheral Family consists of two chip sets, either of which can be used to build a high performance, 100% Micro Channel compatible motherboard. The two chip sets differ primarily in their implementation of the motherboard peripheral bus. The 82310 Chip Set supports either the 8272A or 82072 Floppy Disk Controller. The 82311 Chip Set features a more highly integrated peripheral bus, and includes the 82077 Single Chip Floppy Disk Controller. (The 82311 chip set does not support the 8272A or 82072.) Both chip sets support 80386 systems up to 25 MHz and 80386SX 16 MHz systems.

The following pages describe Intel's Micro Channel Peripheral Family. The first section presents an overview of the 82310 and 82311 chip sets, and discusses system issues such as clock requirements and Micro Channel interface logic. Following this are the individual component descriptions and specifications.

		82310 Chip Set	82311 Chip Set
82303	Local I/O Support Chip		✓
82304	Local I/O Support Chip		✓
82306	Local Channel Support Chip	✓	
82307	DMA/Micro Channel Arbitration Controller	✓	✓
82308	Micro Channel Bus Controller	✓	✓
82309	Address Bus Controller	✓	✓
82706	VGA Graphics Controller	✓	✓
82077	Floppy Disk Controller		✓

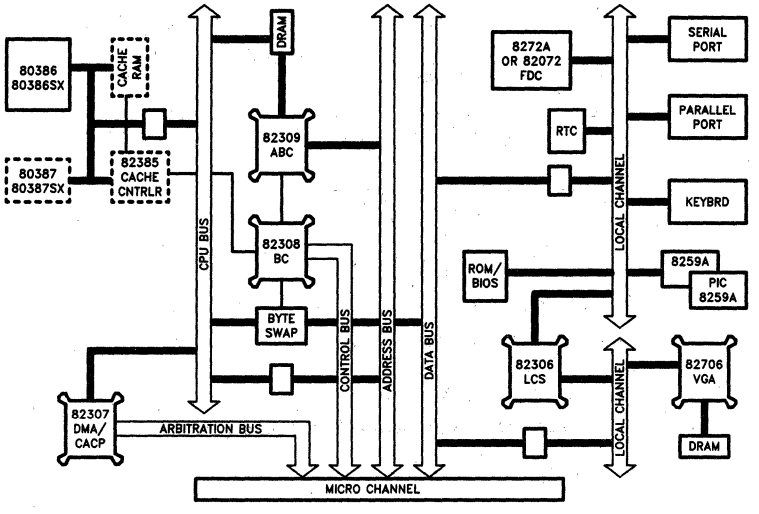
Figure 1. Micro Channel Peripheral Family

82310
Micro Channel COMPATIBLE PERIPHERAL CHIP SET

- **Highly Integrated VLSI Components to Implement Micro Channel™ Compatible Motherboard**
- **Single Architectural Solution for 80386 16 MHz, 20 MHz and 25 MHz systems, and 80386SX 16 MHz Systems**
- **Full Compatibility with IBM Micro Channel Architecture**
- **Zero-Wait State Performance**
- **Cache Interface (82385) for Highest Performance Compatible System Implementation with 80386**
- **Supports up to 16 MB of Memory on Motherboard**
  — Extended Memory for OS/2 Support
- **100% IBM Compatible VGA Graphics**
- **Flexible Memory Architecture Support**
  — Up to 4 Banks of Interleaved Page Memory
  — 256K, 1M, 4M DRAM Support
- **Multiple Floppy Disk Controller Interface to Support 3½" and 5¼" Disk Drives**
- **Keyboard and BIOS Support from 3rd Party**
- **Numeric Coprocessor(s) Interface (80387, 80387SX)**
- **Surface Mount Packaging for Small Footprint Design (0.025" Pitch)**
- **Low Power CHMOS Technology**
- **Available in 100 & 132-Pin Plastic Quad Flat Pack Packages.**
  (See Packaging Spec. # 231369)

Intel's peripheral chip family is designed to support the new generation of Micro Channel compatible systems. Intel's Micro Channel compatible peripheral solution consists of highly integrated VLSI components designed to support 80386 systems up to 25 MHz, as well as 16 MHz 80386SX systems.

The Intel solution is based on the high performance IBM Model 80 register model but it is highly integrated to provide full compatibility across all models. The specifications for 82310 VLSI components conform to architectural specifications defined for the Micro Channel Bus Architecture. The VLSI components are implemented in 1.5 micron CHMOS technology and packaged in space saving surface mount JEDEC flat pack packages.



290167-1

INTRODUCTION

The new generation of Personal Computer systems from IBM offers significant technological advantages over the PC/AT and XT systems. The most significant advancement is in the *Architectural* definition of the bus—Micro Channel Bus. Unlike the AT bus, the Micro Channel is well defined in terms of bus protocol timings. To create a compatible Micro Channel system requires adherence to the Micro Channel timings and electrical drive characteristics.

All IBM Micro Channel models have increased system functionality included on the motherboard. In the older PC/AT architecture, such functionality required the addition of peripheral cards. Specific features added to the motherboard include the Serial Port, Bi-directional Parallel Port and Video Graphics Control.

Micro Channel ARCHITECTURE

The Micro Channel Bus is defined to support an open architecture providing Multi-Master capability, Multi-Device arbitration with fairness, arbitration capability and easy configurability of the total system (Programmable Option Select-POS). Providing full details about the Micro Channel Bus Architecture is beyond the scope of this document. Please refer to IBM Technical Reference Manuals on Micro Channel systems.

To provide Multi-Master capability as defined in the Micro Channel Architecture, each Master device is responsible for driving the Address, Data, arbitration and control signals. For operation reliability and compatibility there are significant constraints in terms of timing and drive levels. These constraints are well documented in IBM's Technical Reference Manual for Micro Channel systems. Intel's chip set is designed to meet the Micro Channel timings.

The Micro Channel has four modes of Memory and I/O Bus cycles. These are Default cycle, Synchronous Extended cycle, Asynchronous Extended cycle and Matched Memory cycle. Each of these bus cycles is supported by the Intel Peripheral chip set.

COMPATIBILITY METRICS

The Intel chip set provides full compatibility with the IBM Micro Channel solution. All Bus cycles comply with the Micro Channel timings. Selection of buffers for drive level with minimum delays to meet Micro Channel timings are specified in the Intel *Designers Guide for Micro Channel Compatible Implementation.*

MEMORY PERFORMANCE

With the Intel chip set, Micro Channel compatible motherboards can be designed to provide zero-wait performance. Performance is predicated on memory design and DRAM speed selection. The Intel chip set offers flexible memory design support to meet various cost/performance goals.

SYSTEM CONSIDERATIONS

System Components

82306	Local Channel Support Chip
82307	DMA/CACP Controller
82308	Micro Channel Bus Controller
82309	Address Bus Controller
82706	VGA Graphics Controller

Note that the above part names/numbers are frequency independent; i.e., they refer to a generic functional VLSI device. To actually implement for example, a 20 MHz system, however requires an 82310-20 Chip Set as opposed to an 82310-16 Chip Set. The 25 MHz version of the 82308 (dubbed the 82308HS-25) cannot be used at 16 MHz or 20 MHz.

To implement a minimum configuration Micro Channel compatible motherboard, each of the five system components listed above are required in addition to the following components:

- 80386 or 80386SX Microprocessor
- TTL/CMOS Buffers for Various Buses in the System
- 8742 Keyboard Controller with Firmware for 101 and 102 Keyboard Interface
- 8272A or 82072 for Floppy Disk Controller
- — 8272A Required to Maintain IBM Look-Alike Motherboard with 3½" Drive Support
- — 82072 for PS/2 Compatible 3½" and AT Compatible 5¼" Disk Drive Interface
- Battery-Backed Real Time Clock with CMOS RAM
- Serial Port
- Parallel Port
- Programmable Interrupt Controllers (Two 8259s)
- Memory
- — ROM BIOS
- — DRAMs for Main Memory
- — DRAMs for VGA
- System Clock Sources
- Mechanical Connectors/Components

The Intel solution is supported by a fully compatible BIOS firmware from a third-party vendor.

82310 CHIP SET SYSTEM CLOCK REQUIREMENTS

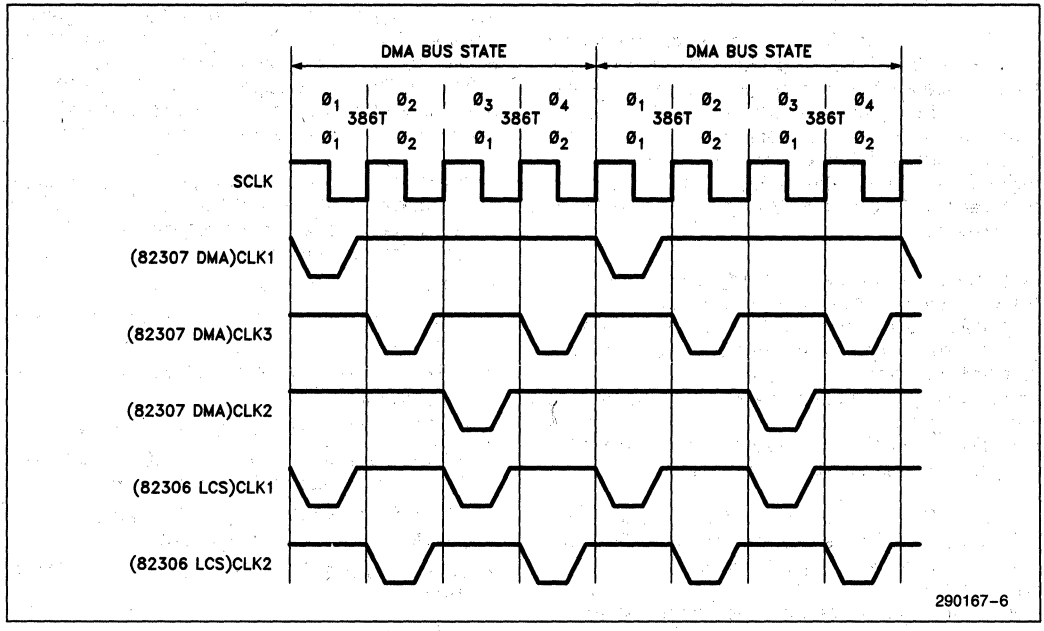
- **Introduction**
- **Clock Definitions**
- **Clock Requirements**

INTRODUCTION

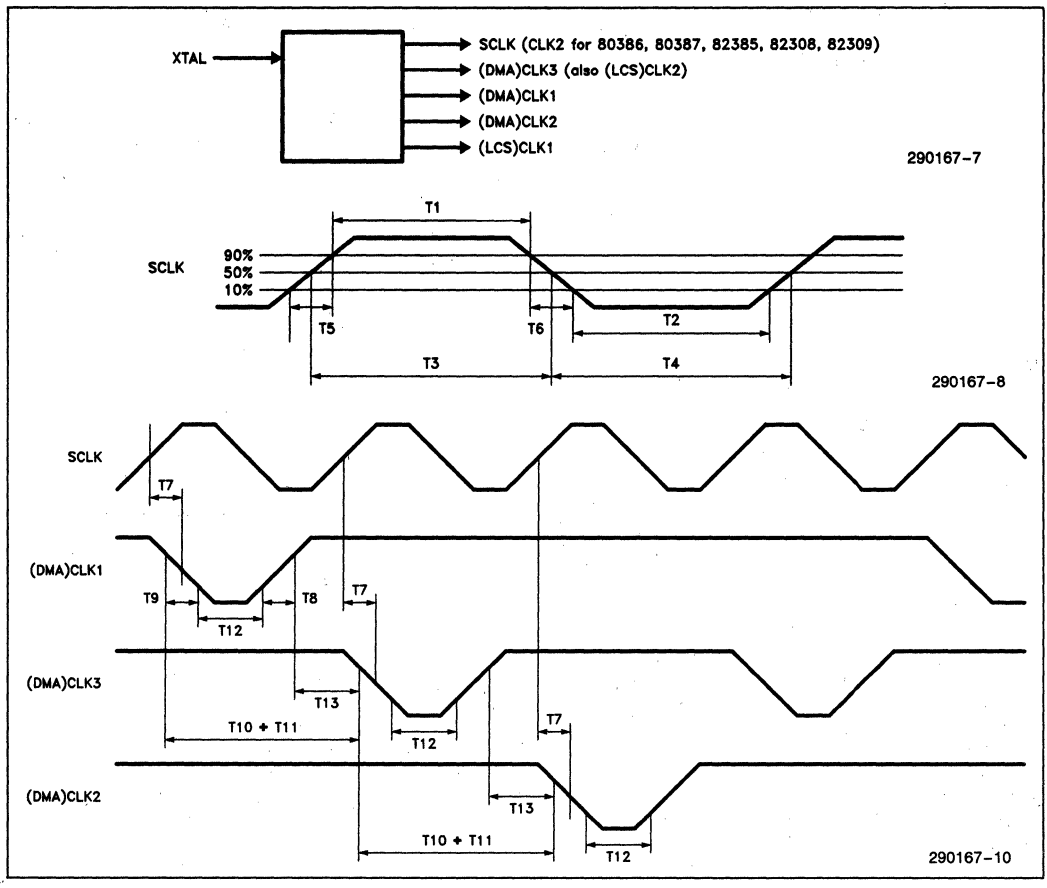
This section describes the basic clocking scheme of the host CPU (80386 or 80386SX), LCS (82306), DMA (82307), BC (82308) and ABC (82309). Although each component spec individually describes its own clock requirements, this section describes the synchronous relationship that exists between them. (Note that several other clocks exist in a Micro Channel system. However, this section describes only those clocks that are synchronously related to the CPU clock.)

The clocking scheme essentially divides the DMA bus state into four phases as depicted in the figure. Note that there is a direct 2-to-1 mapping of 80386 state to DMA state. The DMA (82307) and LCS (82306) comprehend phases by inputting distinct, active low, non-overlapping clock phases. The Address Bus Controller and Bus Controller learn the system phase by synchronously sampling the falling edge of RESET, as described in the component specifications.

BASIC FOUR-PHASE CLOCKING REQUIREMENT



CLOCK CIRCUIT DEFINITION



SYSTEM CLOCK REQUIREMENTS

Symbol	Parameter	Kit 16 MHz		Kit 20 MHz		Kit 25 MHz		Notes
		Min	Max	Min	Max	Min	Max	
T1	SCLK High Time (90%)	8		6.5		5.5		
T2	SCLK Low Time (10%)	8		6.5		5.5		
T3	SCLK High Time (50%)	12		10		9		1
T4	SCLK Low Time (50%)	12		10		9		1
T5	SCLK Rise Time		3.5		3.5		3.5	
T6	SCLK Fall Time		3.5		3.5		3.5	
T7	SCLK-To-DMACLK(N) Skew	−2	3	−2	3	−2	3	2
T8	DMACLK(N) Rise Time		2		2		2	
T9	DMACLK(N) Fall Time		2		2		2	
T10	SCLK Period							
T11	DMACLK-To-DMACLK Skew	−2	2	−2	2	−2	2	2
T12	DMACLK Low Time	15		15		12		
T13	DMACLK Non-Overlap Time	4		4		2		

NOTES:
1. Needed to enforce a duty cycle between 40% and 60%. (45% and 55% at 25 MHz.)
2. Limiting skew to this level is recommended.

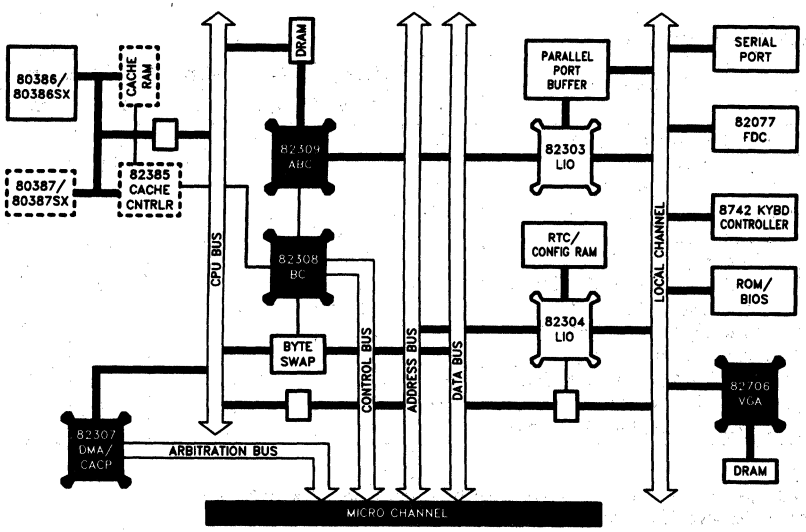
82311
HIGH INTEGRATION Micro Channel COMPATIBLE PERIPHERAL CHIP SET

- **High Integration VLSI Components to Implement Micro Channel™ Compatible Motherboard**
- **Single Architectural Solution for 80386 16 MHz, 20 MHz and 25 MHz Systems and 80386SX 16 MHz Systems**
- **Full Compatibility with IBM Micro Channel Architecture**
- **Zero-Wait State Performance**
- **Cache Interface (82385) for Highest Performance Compatible System Implementation with 80386**
- **Supports up to 16 MB of Memory on Motherboard**
  — Extended Memory for OS/2 Support
- **100% IBM Compatible VGA Graphics**
- **Flexible Memory Architecture Support**
  — Up to 4 Banks of Interleaved Page Memory
  — 256K, 1M, 4M DRAM Support
- **Supports the 82077 Single Chip Floppy Disk Controller, Which Supports 3½" and 5¼" Disk Drives**
- **Keyboard and BIOS Support from 3rd Party**
- **Numeric Coprocessor(s) Interface (80387, 80387SX)**
- **Surface Mount Packaging for Small Footprint Design (0.025" Pitch)**
- **Low Power CHMOS Technology**
- **Available in 100 & 132-Pin Plastic Quad Flat Pack Packages.**

(See Packaging Spec. # 231369)

Intel's peripheral chip family is designed to support the new generation of Micro Channel compatible systems. Intel's Micro Channel compatible peripheral solution consists of highly integrated VLSI components designed to support 80386 systems up to 25 MHz, as well as 16 MHz 80386SX systems.

The Intel solution is based on the high performance IBM Model 80 register model but it is highly integrated to provide full compatibility across all models. The specifications for 82311 VLSI components conform to architectural specifications defined for the Micro Channel Bus Architecture. The VLSI components are implemented in 1.5 micron CHMOS technology and packaged in space saving surface mount JEDEC flat pack packages.



290167-82

INTRODUCTION

The new generation of Personal Computer systems from IBM offers significant technological advantages over the PC/AT and XT systems. The most significant advancement is in the *Architectural* definition of the bus—Micro Channel Bus. Unlike the AT bus, the Micro Channel is well defined in terms of bus protocol timings. To create a compatible Micro Channel system requires adherence to the Micro Channel timings and electrical drive characteristics.

All IBM Micro Channel models have increased system functionality included on the motherboard. In the older PC/AT architecture, such functionality required the addition of peripheral cards. Specific features added to the motherboard include the Serial Port, Bi-directional Parallel Port and Video Graphics Control.

Micro Channel ARCHITECTURE

The Micro Channel Bus is defined to support an open architecture providing Multi-Master capability, Multi-Device arbitration with fairness, arbitration capability and easy configurability of the total system (Programmable Option Select-POS). Providing full details about the Micro Channel Bus Architecture is beyond the scope of this document. Please refer to IBM Technical Reference Manuals on Micro Channel systems.

To provide Multi-Master capability as defined in the Micro Channel Architecture, each Master device is responsible for driving the Address, Data, arbitration and control signals. For operation reliability and compatibility there are significant constraints in terms of timing and drive levels. These constraints are well documented in IBM's Technical Reference Manual for Micro Channel systems. Intel's chip set is designed to meet the Micro Channel timings.

The Micro Channel has four modes of Memory and I/O Bus cycles. These are Default cycle, Synchronous Extended cycle, Asynchronous Extended cycle and Matched Memory cycle. Each of these bus cycles is supported by the Intel Peripheral chip set.

COMPATIBILITY METRICS

The Intel chip set provides full compatibility with the IBM Micro Channel solution. All Bus cycles comply with the Micro Channel timings. Selection of buffers for drive level with minimum delays to meet Micro Channel timings are specified in the Intel *Designers Guide for Micro Channel Compatible Implementation.*

MEMORY PERFORMANCE

With the Intel chip set, Micro Channel compatible motherboards can be designed to provide zero-wait performance. Performance is predicated on memory design and DRAM speed selection. The Intel chip set offers flexible memory design support to meet various cost/performance goals.

SYSTEM CONSIDERATIONS

System Components

82303	Local I/O Support Chip
82304	Local I/O Support Chip
82307	DMA/CACP Controller
82308	Micro Channel Bus Controller
82309	Address Bus Controller
82706	VGA Graphics Controller
82077	Floppy Disk Controller

Note that the above names/numbers are frequency independent; i.e., they refer to a generic functional VLSI device. To actually implement for example, a 20 MHz system, however, requires an 82311-20 Chip Set as opposed to an 82311-16 Chip Set. The 25 MHz version of the 82308 (dubbed the 82308HS-25) cannot be used at 16 MHz or 20 MHz.

To implement a minimum configuration Micro Channel compatible motherboard, each of the seven system components listed above are required in addition to the following components:

- 80386 or 80386SX Microprocessor
- TTL Buffers for Various Buses in the System
- 8742 Keyboard Controller with Firmware for 101 and 102 Keyboard Interface
- Battery-Backed Real Time Clock with CMOS RAM
- Serial Port
- Memory
  - — ROM BIOS
  - — DRAMs for Main Memory
  - — DRAMs for VGA
- System Clock Sources
- Mechanical Connectors/Components

The Intel solution is supported by a fully compatible BIOS firmware from a third-party vendor.

82311 CHIP SET SYSTEM CLOCK REQUIREMENTS

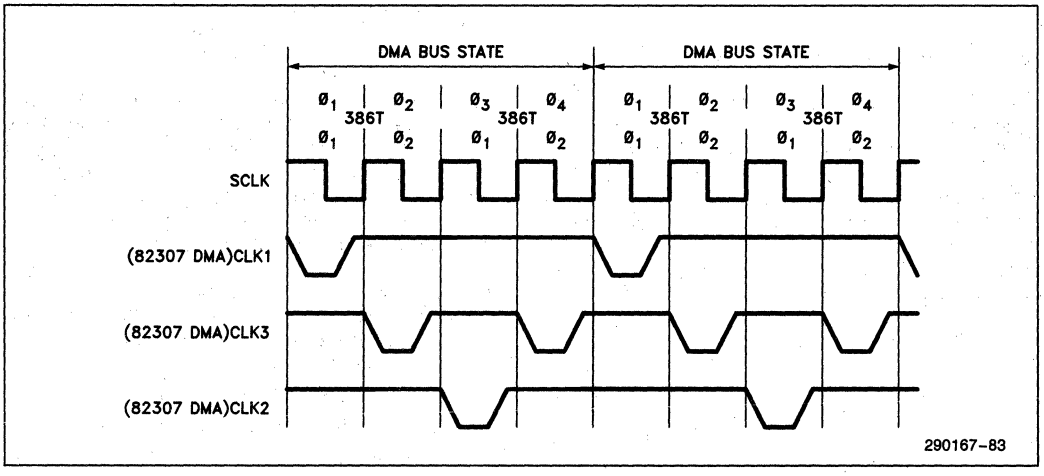
- **Introduction**
- **Clock Definitions**
- **Clock Requirements**

INTRODUCTION

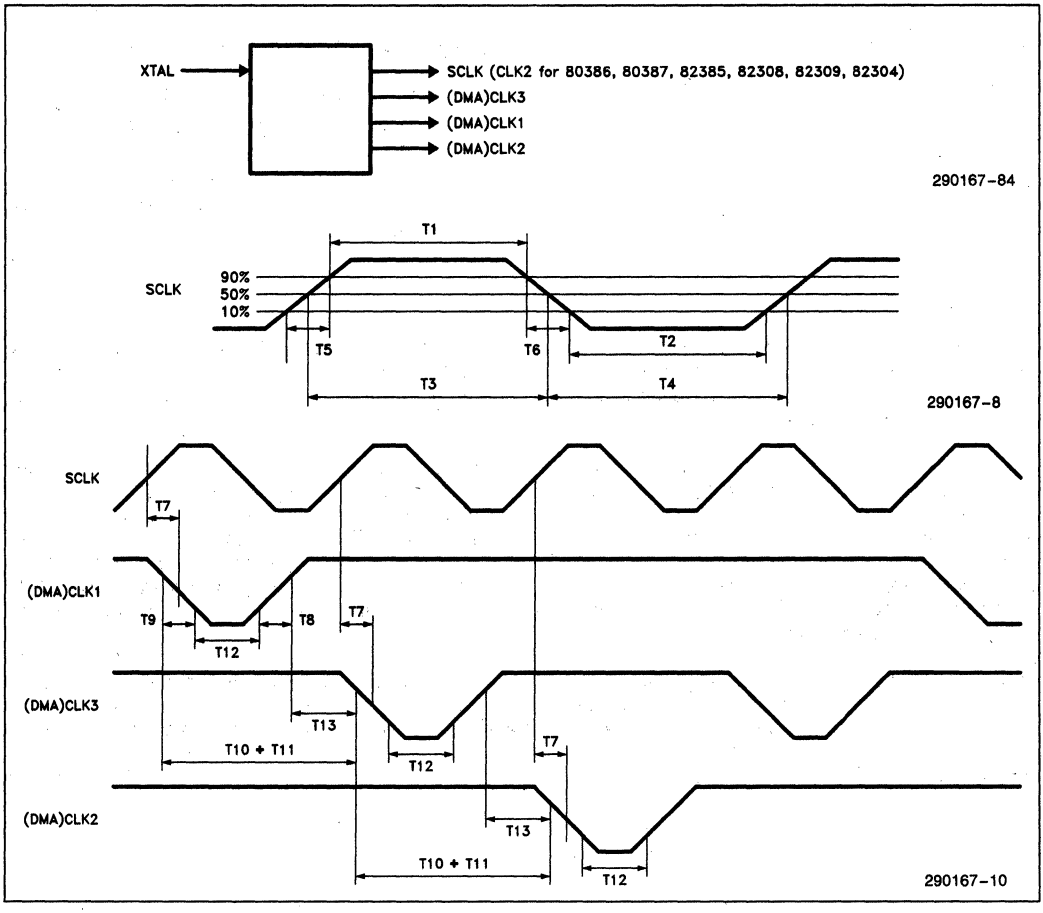
This section describes the basic clocking scheme of the host CPU (80386 or 80386SX), LIO (82304), DMA (82307), BC (82308) and ABC (82309). Although each component spec individually describes its own clock requirements, this section describes the synchronous relationship that exists between them. (Note that several other clocks exist in a Micro Channel system. However, this section describes only those clocks that are synchronously related to the CPU clock.)

The clocking scheme essentially divides the DMA bus state into four phases as depicted in the figure. Note that there is a direct 2-to-1 mapping of 80386 state to DMA state. The DMA (82307) comprehends phases by inputting distinct, active low, non-overlapping clock phases. The Address Bus Controller, Bus Controller and LIO device learn the system phase by synchronously sampling the falling edge of RESET, as described in the component specifications.

BASIC FOUR-PHASE CLOCKING REQUIREMENT



CLOCK CIRCUIT DEFINITION



SYSTEM CLOCK REQUIREMENTS

Symbol	Parameter	Kit 16 MHz		Kit 20 MHz		Kit 25 MHz		Notes
		Min	Max	Min	Max	Min	Max	
T1	SCLK High Time (90%)	8		6.5		5.5		
T2	SCLK Low Time (10%)	8		6.5		5.5		
T3	SCLK High Time (50%)	12		10		9		1
T4	SCLK Low Time (50%)	12		10		9		1
T5	SCLK Rise Time		3.5		3.5		3.5	
T6	SCLK Fall Time		3.5		3.5		3.5	
T7	SCLK-To-DMACLK(N) Skew	−2	3	−2	3	−2	3	2
T8	DMACLK(N) Rise Time		2		2		2	
T9	DMACLK(N) Fall Time		2		2		2	
T10	SCLK Period							
T11	DMACLK-To-DMACLK Skew	−2	2	−2	2	−2	2	2
T12	DMACLK Low Time	15		15		12		
T13	DMACLK Non-Overlap Time	4		4		2		

NOTES:
1. Needed to enforce a duty cycle between 40% and 60% (45% and 55% at 25 MHz).
2. Limiting skew to this level is recommended.

Micro Channel INTERFACE AND SPECIFICATIONS

- **Introduction**
- **Micro Channel Specifications**
- **Micro Channel Interface Logic Requirements**
  - **— 80386 System Data Path**
  - **— 80386 System Address/Command Path**
  - **— 80386SX System Data Path**
  - **— 80386SX System Address Command Path**

INTRODUCTION

This section describes the interface between the host CPU (80386, 80386SX), DMA (82307), Bus Controller (82308) and Micro Channel Bus. This interface provides 100% compliance to published Micro Channel timings, driver type requirements, drive levels, and drive current capability. Timings meet the full capacitive load allowed on the Micro Channel.

The Micro Channel Specifications included in this section assume the specific TTL Data, Address, and Command Path interfaces depicted in the accompanying figures. Timing analysis was based on the Bus Controller AC specifications included in the 82308 Bus Controller section. Worst case TTL analysis was used, except when two related signals share a path through the same physical chip. (For example, since MMCCMD#, S0#, and S1# propagate through the same 74F241 package in an 80386 system, one signal will not experience a worst case delay while the other sees a best case. Rather, it is assumed that the signals will track within 2 ns of each other.) For this reason, it is important to follow the recommendations detailed at the end of this section in the Interface Logic Notes.

The F and AS TTL logic is typically specified into a 50 pF load, worst case delays were derated at 1 ns per 50 pF for loads greater than 50 pF. As an example, the 74F241 published maximum delay is specified as 7 ns. To meet Micro Channel bus loading of 250 pF, a 4 ns derating factor was added, resulting in an effective worst case delay of 11 ns.

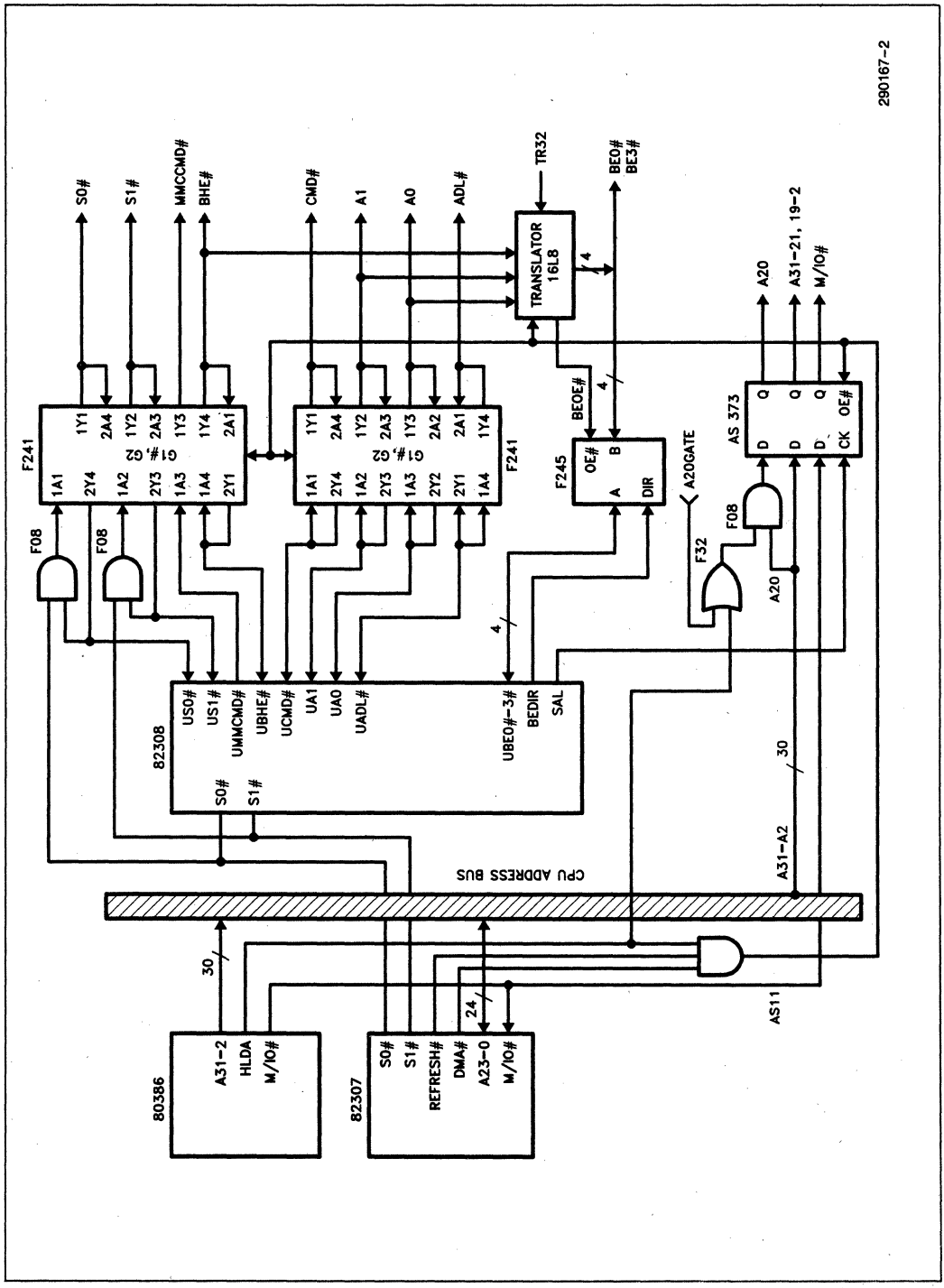
DEFAULT CYCLE SPECIFICATIONS — ALL KITS

Symbol	Parameter	Min	Max
T1	Status active from ADDR,M/IO#,REFRESH#	10	
T2	CMD# active from Status active	55	
T3	ADL# active from ADDR,M/IO#,REFRESH#	45	
T4	ADL# active to CMD# active	40	
T5	ADL# active from Status active	12	
T6	ADL# pulse width	40	
T7	Status hold from ADL# inactive	25	
T8	ADDR,M/IO#,REFRESH#,SBHE# hold frm ADL# INACTIVE	25	
T9	ADDR,M/IO#,REFRESH#,SBHE# hold frm CMD# ACTIVE	30	
T10	Status hold from CMD# active	30	
T11	SBHE# setup to ADL# inactive	40	
T12	SBHE# setup to CMD# active	40	
T13	CDDS16/32 active from ADDR,M/IO#,REFRESH#		55
T14	CDSFDBK# active from ADDR,M/IO#,REFRESH#		60
T15	CMD# active from ADDRESS valid	85	
T16	CMD# pulse width	90	
T17	Write data setup to CMD# active	0	
T18	Write data hold from CMD# inactive	30	
T19	Status to Read Data valid (Access Time)		125
T20	Read Data valid from CMD# active		60
T21	Read Data hold from CMD# inactive	0	
T22	Read Data bus tri-state from CMD# INACTIVE		40
T23	CMD# active to next CMD# active	190	
T23A	CMD# inactive to next CMD# active	80	
T23B	CMD# inactive to next ADL# active	40	
T24	Next Status active from Status Inactive	30	
T25	Next Status active to CMD# Inactive		20
T26	CHRDY INACTIVE FROM ADDR VALID		60
T27	CHRDY INACTIVE FROM STATUS ACTIVE		30
T28	CHRDY RELEASE FROM CMD# ACTIVE		30
T28D	READ DATA VALID FROM CMD# ACTIVE		160
T29S	READ DATA VALID FROM CHRDY RELEASE		60
T31	BE#(0–3) from Addr valid (32-Bit Masters Only)		40
T32	BE#(0–3) active from SBHE#,A0,A1 active		30
T33	BE#(0–3) active to CMD# active	10	

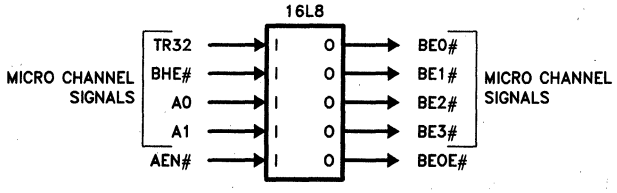
MATCHED MEMORY CYCLE SPECIFICATIONS — ALL KITS

Symbol	Parameter	Min	Max
T1	ADDR VALID TO STATUS ACTIVE	10	
T2	Status valid to MMCCMD# active	82	
T3	ADDR hold from MMCCMD# active	20	
T4	Status hold from MMCCMD# active	25	
T5	CDDS16/32 active from ADDR valid		55
T6	MMCR# active from ADDR valid		55
T7	CDSFDBK# active from ADDR valid		60
T8	ADDR valid to MMCCMD# active	100	
T9	MMCCMD# pulse width	85	
T10	Write Data valid to MMCCMD# active	0	
T11	Write Data hold from MMCCMD# inactive	30	
T12A	Read Data valid from Status active		145
T12B	for non-aligned xfers (16b < = = > 32b)		145
T13A	Read Data valid from MMCCMD# active		60
T13B	for non-aligned xfers (16b < = = > 32b)		60
T14	Read Data hold from MMCCMD# inactive	0	
T15	Read Data off dly from MMCCMD# inactive		40
T16	MMCCMD# active to next MMCCMD# active	180	
T17	CDCHRDY valid from ADDR valid		70
T18	CDCHRDY valid from Status active		30
T23	Status inactive pulse width	30	
T24	MMCCMD# inactive to Status active	5	
T25	MMCCMD# inactive pulse width	85	
T26	MMCCMD# ACTIVE TO NEXT STATUS ACTIVE	90	

80386 SYSTEM ADDRESS/COMMAND PATH



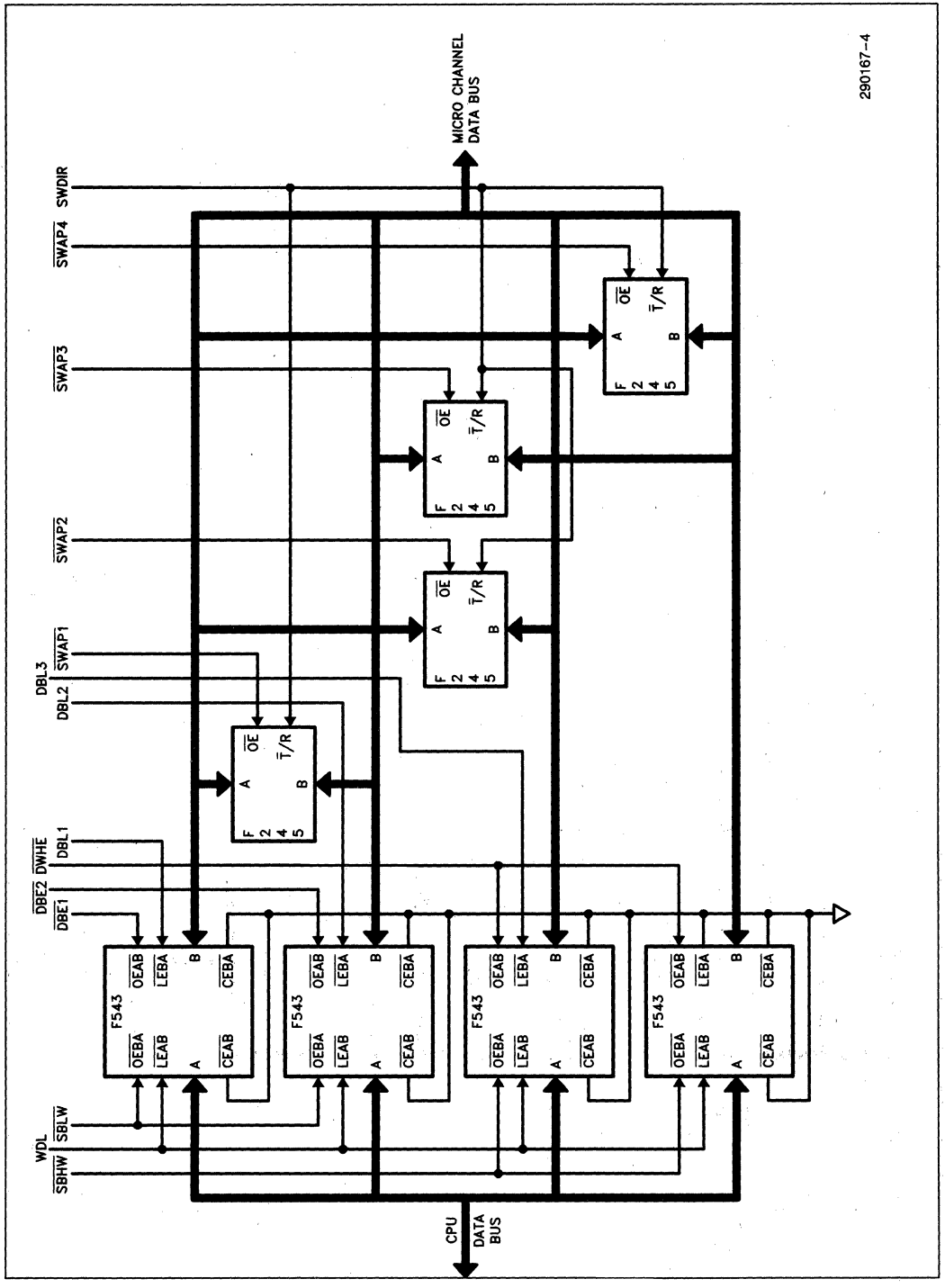
80386 SYSTEM BYTE ENABLE TRANSLATOR PAL



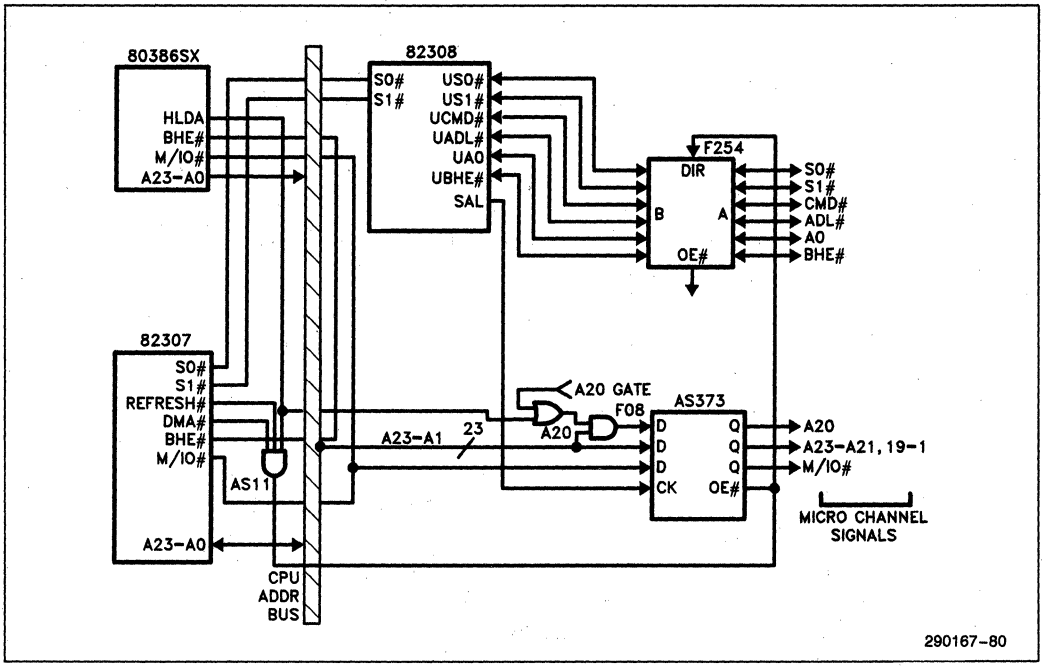
TR32	AEN#	BHE#	A1	A0	BE3#	BE2#	BE1#	BE0#	BEOE#
1	1	0	0	0	1	1	0	0	1
1	1	0	0	1	1	1	0	1	1
1	1	0	1	0	0	0	1	1	1
1	1	0	1	1	0	1	1	1	1
1	1	1	0	0	1	1	1	0	1
1	1	1	0	1	1	1	0	1	1
1	1	1	1	0	1	0	1	1	1
1	1	1	1	1	0	1	1	1	1
0	X	X	X	X	TS	TS	TS	TS	0
X	0	X	X	X	TS	TS	TS	TS	0

TS = TRISTATE

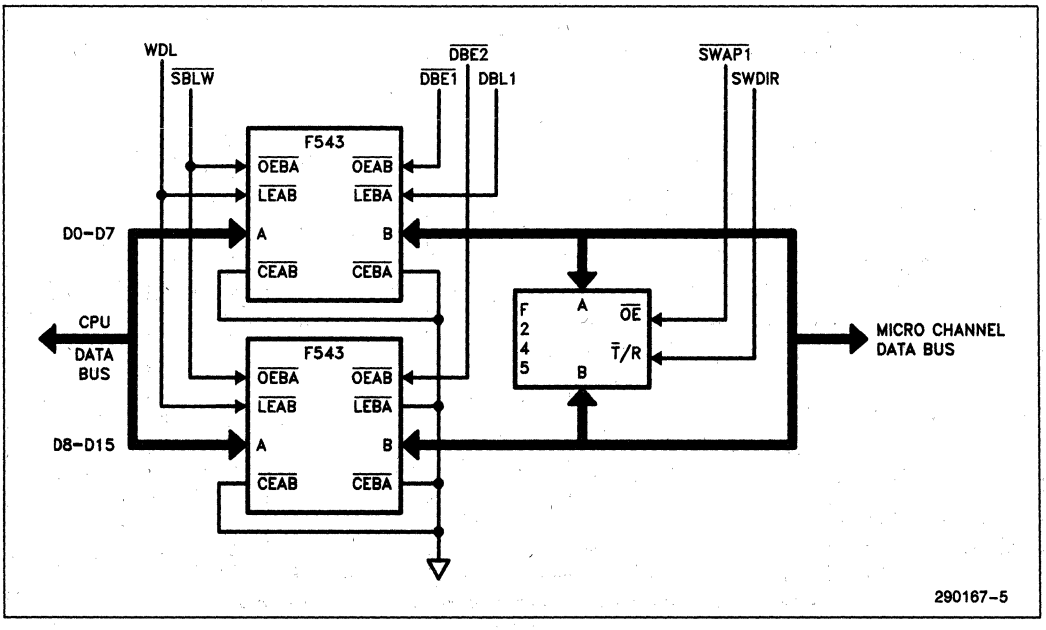
80386 SYSTEM DATA PATH



80386SX SYSTEM ADDRESS/COMMAND PATH



80386SX SYSTEM DATA PATH



Micro Channel Interface Logic Notes

80386 SYSTEM

1. The F08 gates in the S0#, S1# path are required at 20 MHz and at 25 MHz. They should not be used at 16 MHz.
2. In an 82385 system, the A20 gate logic is on the 80386 local bus, and is thus not required on the 82385 local bus as shown in the diagram.
3. The F08 gates in the S0#, S1# path and the F08 in the A20 path should all be from the same TTL package.
4. It is important that S0#, S1#, BHE#, and MMCCMD# go through the same F241 package, and that CMD#, ADL#, A0, and A1 go through the same package.

80386SX SYSTEM

1. The F08 gates in the S0#, S1# path and in the A20 path should all be from the same package.

PLASTIC PACKAGING INFORMATION

(See Packaging Spec. Order # 231369)

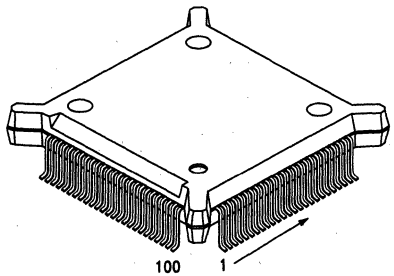
Introduction

The individual components of Intel's Micro Channel Compatible Peripheral Chip Sets come in JEDEC standard Gull Wing packages (25 MIL pitch), with "bumpers" on the corners for ease of handling. Please refer to the accompanying table for the package associated with each device, and to the individual component specifications for pinouts. (Note that the individual pinouts are numbered consistently with the numbering scheme depicted in the accompanying figures.)

MICRO CHANNEL COMPATIBLE PERIPHERAL FAMILY COMPONENT PACKAGES

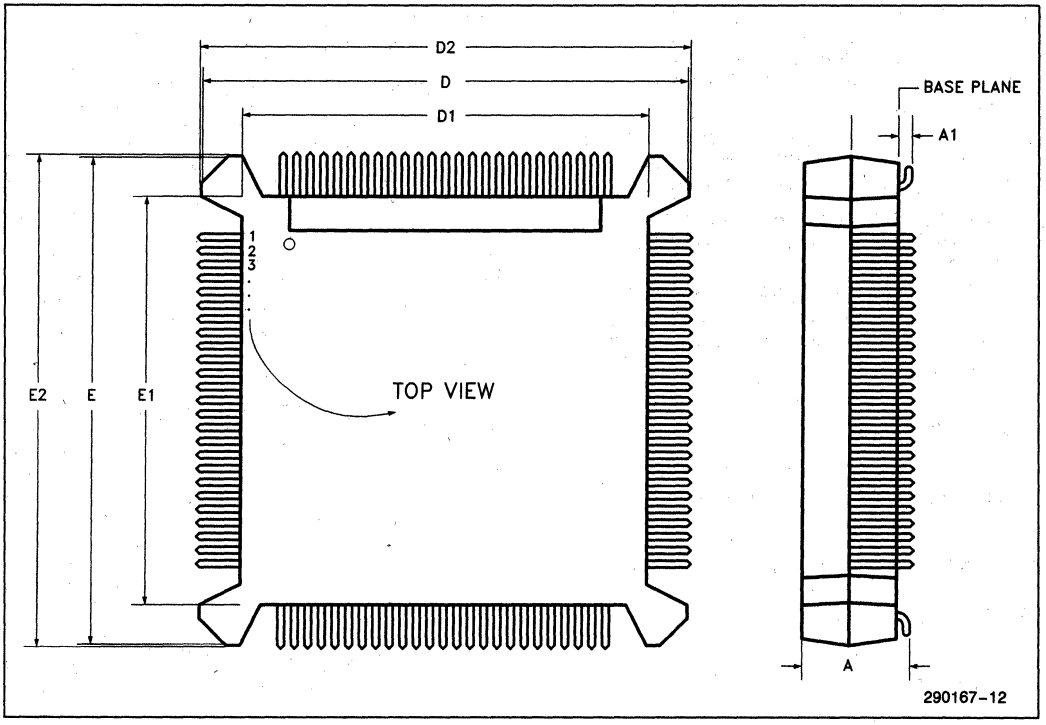
Component	Package
82303	100 Pin PQFP
82304	132 Pin PQFP
82306	100 Pin PQFP
82307	132 Pin PQFP
82308	100 Pin PQFP
82309	100 Pin PQFP
82706	132 Pin PQFP
82077	68-Pin PLCC, See Component Data Sheet

PLASTIC QUAD FLAT PACK (PQFP)

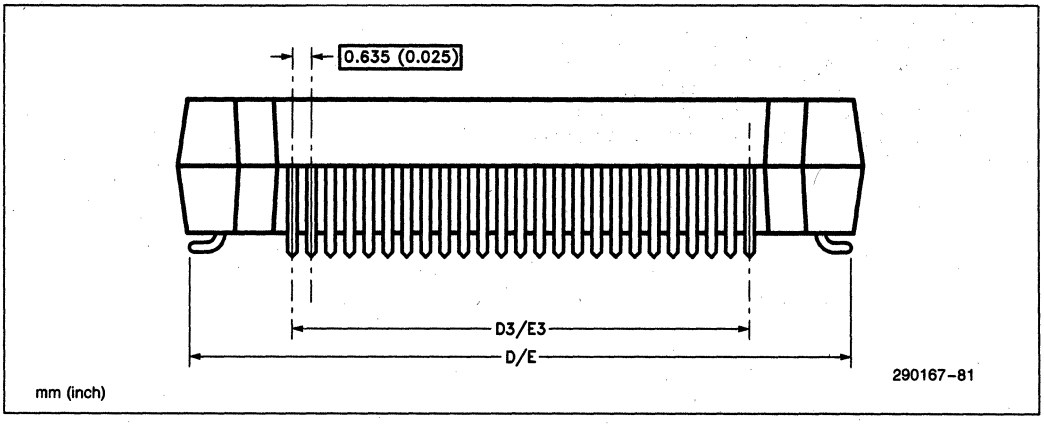


290167-11

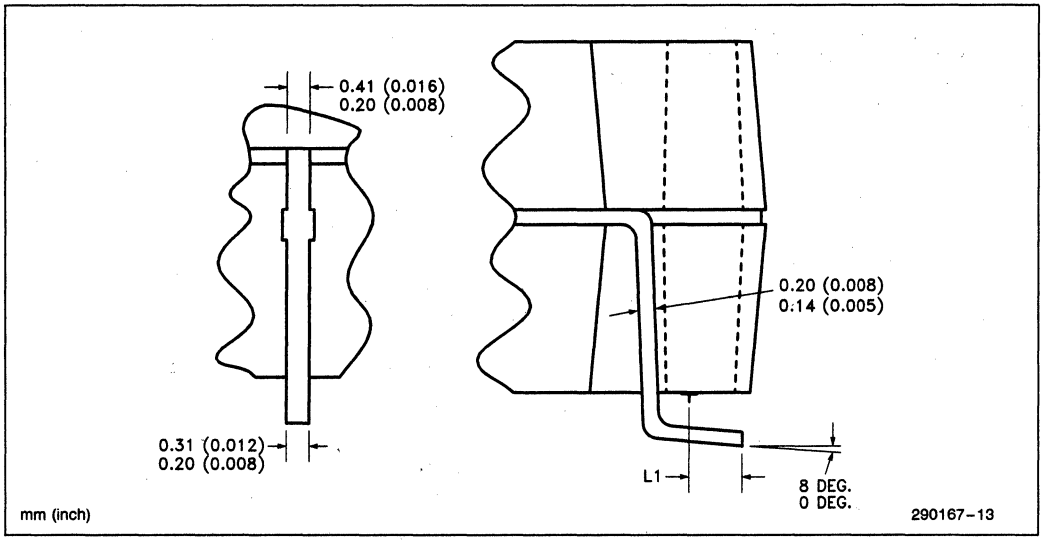
PRINCIPAL DIMENSIONS & DATUMS



TERMINAL DETALS



TYPICAL LEAD



Case Outline Drawings
Plastic Fine Pitch Chip Carrier

		0.025 inch Pitch				0.84 mm Pitch			
Symbol	**Description**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
N	Lead Count	100		132		100		132	
A	Package Height	0.160	0.170	0.160	0.170	4.06	4.32	4.06	4.32
A1	Standoff	0.020	0.030	0.020	0.030	0.51	0.76	0.51	0.76
D, E	Terminal Dimension	0.875	0.885	1.075	1.085	22.23	22.48	27.31	27.56
D1, E1	Package Body	0.747	0.753	0.947	0.953	18.97	19.13	24.05	24.21
D2, E2	Bumper Distance	0.897	0.903	1.097	1.103	22.78	22.94	27.86	28.02
D3, E3	Lead Dimension	0.600 Ref		0.800 Ref		15.24 Ref		20.32 Ref	
L1	Foot Length	0.020	0.030	0.020	0.030	0.51	0.76	0.51	0.76
		Inch				mm			

REVISION HISTORY

The A.C. Specifications of the 82306, 82307, 82308 and 82309 chips have been modified with respect to the 82310 Data Sheet Order Number 290167-001. These modifications apply to the 16 MHz and 20 MHz kits only.

82306 Spec Revisions

— T1 changed from 14 ns to 15 ns at 16 MHz ... from 12 ns to 15 ns at 20 MHz

— T2 changed from 10 ns to 4 ns at 16 MHz ... from 7 ns to 4 ns at 20 MHz

— T16 changed from 180 ns to 230 ns at 16 MHz and 20 MHz

— T20 changed from 200 ns to 120 ns at 16 MHz and 20 MHz

82307 Spec Revisions

— T1 changed from 14 ns to 15 ns at 16 MHz ... from 12 ns to 15 ns at 20 MHz

— T2 changed from 10 ns to 4 ns at 16 MHz ... from 7 ns to 4 ns at 20 MHz

— T33 cap load C(L) changed from 50 pF to 25 pF

82308 Spec Revisions

— T5A changed from 37 ns to 30 ns at 16 MHz and 20 MHz

— T6A broken into two specs ... T6A for FLUSH, and T6C for SNOOP#

— T44 broken into two specs ... T44A for Setup to SCLK and T44B for Setup to UCMD# ... T44A left at 0 ns, but T44B changed to 3 ns

— T47C changed from 35 ns to 40 ns

82309 Spec Revisions

— T18A changed from 45 ns to 30 ns at 16 MHz

— T18B changed from 50 ns to 38 ns at 16 MHz

— T18C changed from 50 ns to 35 ns at 16 MHz

— T27B changed from 30 ns to 33 ns

— T32B changed from 50 ns to 55 ns

— T32E (Min) changed from 8 ns to 4 ns

— T32E (Max) changed from 30 ns to 27 ns

— T32G (Min) changed from 6 ns to 4 ns

— T32I changed from 50 ns to 40 ns

— T33 (Min) changed from 8 ns to 5 ns

— T34 (Min) changed from 6 ns to 3 ns

— T34 (Max) changed from 26 ns to 27 ns

— T35 changed to 115 ns to 100 ns at 16 MHz and from 90 ns at 20 MHz

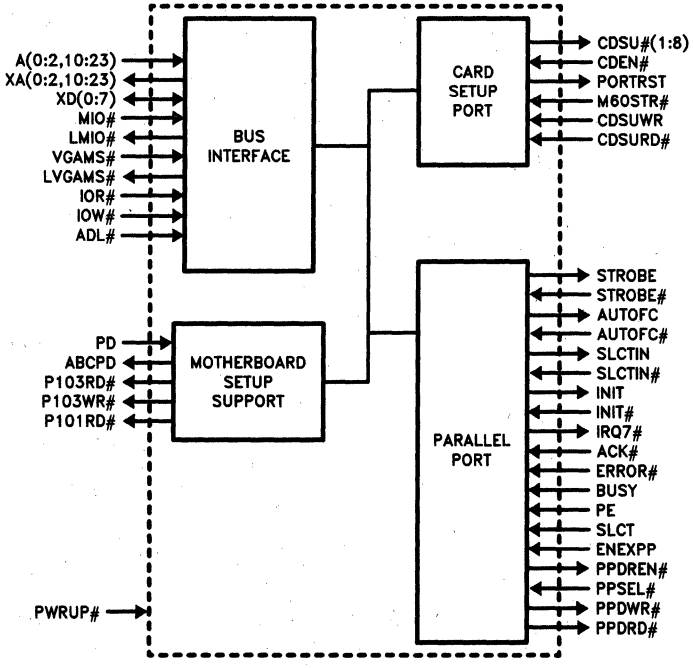
— T45 changed from 26 ns to 32 ns

82303
LOCAL I/O SUPPORT CHIP

- **High Integration—The 82304, 82303 and 82077 Floppy Disk Controller Replace 50 IC's in IBM Design**
- **Integrated Parallel Port**
- **Integrated Card Setup Port (96H)**
- **Supports System Board Setup**
- **Integrated Peripheral Bus Address Latches**
- **Low Power CHMOS Technology**
- **100-Pin Plastic Quad Flat Package**

The 82303 Local Channel Support Chip, along with its companion chip (the 82304) and the 82077 Floppy Disk Controller, significantly reduce system cost, design effort, and form factor constraints by replacing 50 IC devices in an equivalent IBM system.

The 82303 integrates most all logic required to implement a parallel port. This port operates either as a standard parallel port or as a Microchannel architecture compatible "extended mode" (bi-directional) port. The 82303 also integrates the Card Setup Port (96H) and several peripheral bus address latches, and provides signals in support of system setup functions.



290184-1

November 1988
Order Number: 290184-001

Introduction

The 82303 is a high integration device intended for Microchannel compatible system designs. It integrates the Microchannel Card Setup Port, a parallel port, several peripheral bus address latches, and a variety of system board setup functions. The 82303, in conjunction with its sister chip the 82304 and the 82077 Floppy Disk Controller, replaces approximately 50 IC devices in an equivalent IBM system. Included as an appendix to this data sheet is a functional logic diagram of the 82303 that will facilitate understanding of the part. Note that the 82304 and 82303 integrate a variety of system ports. For programming and register level details, please refer to the IBM Technical Reference Manual.

Bus Interface

The Bus Interface unit interfaces the 82303 to the Microchannel and peripheral busses. It inputs the unlatched Microchannel address, latches it for internal use, and makes the latched version available externally for other peripheral bus resources. It also provides additional latches for decodes generated from the Microchannel address.

Parallel Port

The 82303 integrates most all logic required to implement a standard or "extended-mode" parallel port. The only logic not integrated is that which directly drives the physical parallel port connector, specifically one '05 (open collector) inverter package and one '652 data buffer. (This allows the system design to stay clear of directly exposing a VLSI component to an external connector.) The parallel port can serve as LPT1, LPT2, or LPT3, as dictated by the decode received via the input parallel port decode PPSEL#.

Card Setup Port

The 82303 integrates the Card Setup Port (96H), which generates the card setup lines to the individual Microchannel connectors. This port also features a software generated reset capability that resets the Microchannel, serial port, and parallel port independently of the rest of the system.

Motherboard Setup Support

The 82303 generates decoded read/write strobes for system board setup port 103H, and a read strobe for setup port 101H. It also generates a version of the system board POS decode (ABCPD) that is then forwarded to the 82309 Address Bus Controller. Note that other system board setup ports can be easily implemented externally using the same PD (POS Decode) that the 82303 uses.

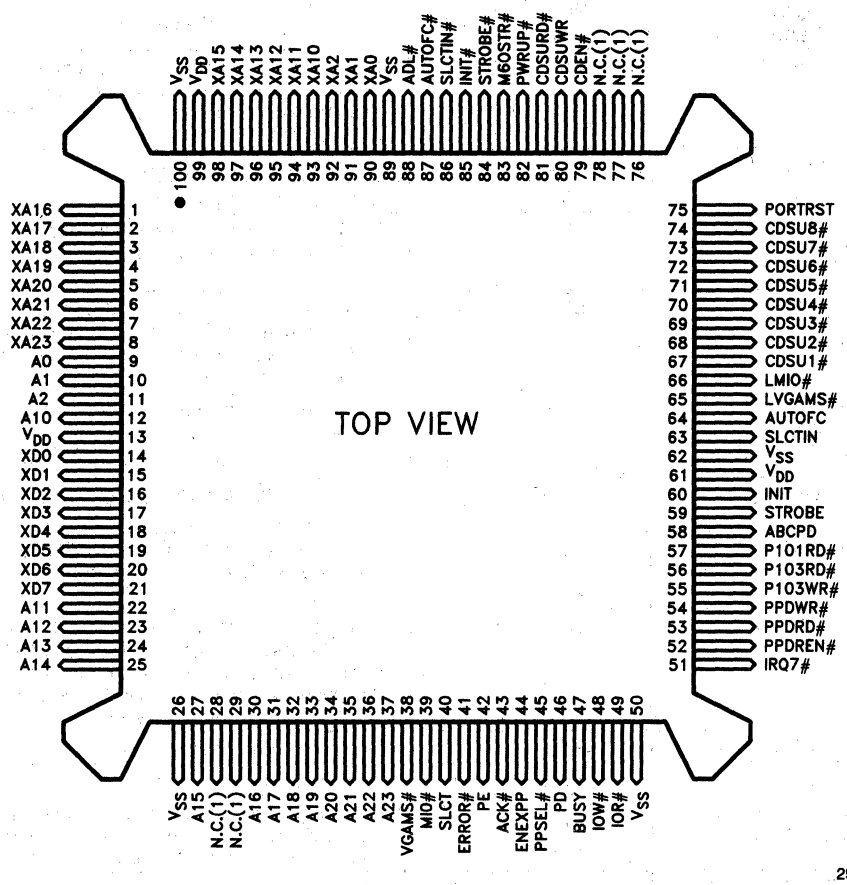
82303 Local Channel Support Chip Pin Definitions

Signal Name	Pin Number	I/O	Description
PWRUP#	82	I	Power-up reset input. Brings 82303 to initial known state.
A[0:2, 10:23]	9–12, 22–25, 27, 30–37	I	Microchannel address inputs. These signals are internally latched. (Note that in systems in which a full 24-bit peripheral address is not required, the upper significant address latches may be used as general purpose decode latches.)
XA[0:2, 10:23]	90–98, 1–8	O	Peripheral Bus Address. These outputs are latched versions of the Microchannel address inputs.
XD[0:7]	14–21	I/O	Bi-directional peripheral data bus.
MIO#	39	I	Microchannel MIO# indicator.
LMIO#	66	O	Latched Microchannel MIO# indicator. The MIO#/LMIO# pin combination may be used as a general purpose latch if LMIO# is not required.
VGAMS#	38	I	VGA memory buffer decode.

82303 Local Channel Support Chip Pin Definitions (Continued)

Signal Name	Pin Number	I/O	Description
LVGAMS#	65	O	Latched VGA memory buffer decode. The VGAMS#/LVGAMS# pin combination may be used as a general purpose latch if LVGAMS# is not required.
IOR#, IOW#	49, 48	I	82303 read/write strobes.
ADL#	88	I	Microchannel ADL# input.
PD	46	I	POS decode. Decode driven in response to accesses to system board setup Ports 100, 101, 103–107H.
ABCPD	58	O	Address Bus Controller (82309) POS decode. This is simply the PD input gated by an active IOW# signal, and insures that the 82309 does not see a decoding glitch.
P101RD#, P103RD#, P103WR#	57, 56, 55	O	Various system board setup port read/write strobes.
CDSU#[1:8]	67–74	O	Card setup signals to the Microchannel slots.
CDEN#	79	I	Port 100-107H decode used as qualifier for card setup signals.
PORTRST	75	O	Microchannel reset signal. The "OR" of the power-up reset and the reset function built into Port 96H.
M60STR#	83	I	Model 60 strap. When low, the 82303 will drive Port 96H data in either a Port 96 or 97H read. (This is in keeping with the Model 50/60 definition.) When high, the 82303 will remain tri-stated during a Port 97H read.
CDSUWR	80	I	Port 96–97H write strobe.
CDSURD#	81	I	Port 96–97H read strobe.
STROBE, AUTOFC, SLCTIN, INIT	59, 64, 63, 60	O	Parallel port control outputs. These signals are externally buffered with open collector inverters before driving the parallel port connector.
STROBE#, AUTOFC#, SLCTIN#, INIT#	84, 87, 86, 85	I	Parallel port control inputs.
IRQ7#	51	O	Parallel port interrupt request.
ACK#, ERROR#, BUSY, PE, SLCT	43, 41, 47, 42, 40	I	Parallel port status inputs.
ENEXPP	44	I	Enable parallel port extended mode. Allows parallel port to operate bi-directionally.
PPSEL#	45	I	Parallel port chip select.
PPDREN#	52	O	Enables the external '652 parallel port data buffer to be used bi-directionally. This signal is a function of the Control port direction bit (bit 5) and the ENEXPP input.
PPDWR#, PPRD#	54, 53	O	Parallel port data buffer write/read strobes.
V_{DD}	13, 61, 99		Power.
V_{SS}	26, 50, 62, 89, 100		Ground.
N.C.	28, 29, 76, 77, 78		No Connect.

82303 100-Pin PQFP Pinout



290184-2

NOTE:
1. N.C. pins must be left not connected.

82303 PARAMETRICS ABSOLUTE MAXIMUM RATINGS*

Case Temperature Under Bias .... −40°C to +85°C
Storage Temperature ..... −65°C to +150°C
Voltage to any Pin
with Respect to Ground −0.3V to +(V_{CC} +0.3)V
DC Supply Voltage (V_{CC}) ..... −0.3V to +7.0V
DC Input Current ..... ±10 mA

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS T_C = 0°C to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Min	Max	Units	Notes
V_{IL}	Input Low Voltage		0.8	V	
V_{IH}	Input High Voltage	2.0		V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 4 mA (Note 1)
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 4 mA (Note 1)
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2 mA (Note 2)
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 2 mA (Note 2)
I_{CC}	Power Supply Current		180	mA	No DC Loads
I_{LI}	Input Leakage Current		±10	μA	$V_{SS} < V_{IN} < V_{CC}$
I_{OZ}	TRI-STATE Output Leakage Current		±10	μA	$V_{SS} < V_{OUT} < V_{CC}$

NOTES:
1. CDSU# [1:8], XA [0:2, 10:23], XD[0:7].
2. All outputs other than those listed in Note 1.

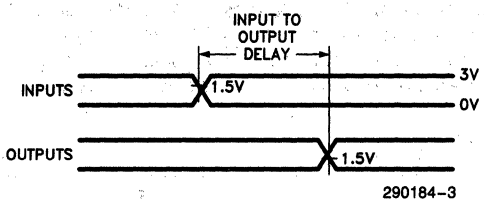
82303 A.C. SPECIFICATIONS T_C = 0°C to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Min	Max	C_L (pF)	Notes
T_1	PWRUP#, Pulse Width	500			
T_2	IOR#, IOW#, CDSURD#, CDSUWR Pulse Width	170			
T_3	Write Data Setup	25			(Note 1)
T_4	Write Data Hold	10			(Note 2)
T_5	Read Data Valid Delay		60	100	(Note 3)
T_6	Read Data Float Delay		40	100	(Note 5)
T_7	Status Inputs to XD[0:7]		35	100	(Note 6)
T_8	Write Strobe Delays		35	50	(Note 7)
T_9	Read Strobe Delays		40	50	(Note 8)
T_{11}	PPSEL#, PD Setup to IOR#, IOW# ↓	20			
T_{12}	PPSEL#, PD Hold from IOR#, IOW# ↑	5			
T_{13}	XA[0:2, 10:23] DLY from ADL# ↓		35	100	
T_{14}	A[0:2, 10:23], VGAMS#, MIO# Setup to ADL# ↑	30			
T_{17}	CDSU# [1:8] Delay from CDEN#		28	75	
T_{18}	IOR# ↑ to ADL# ↓	30			
T_{19}	LVGAMS#, LMIO# Delay from ADL# ↓		35	50	

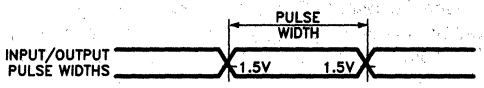
NOTES:
1. To IOW# or CDSUWR active, whichever is appropriate.
2. From IOW# or CDSUWR inactive, whichever is appropriate.
3. From IOR# or CDSURD# active, whichever is appropriate.
5. From IOR# or CDSURD# inactive, whichever is appropriate.
6. Parallel port status inputs include SLCT, PE, BUSY, ERROR#, and ACK#.
7. Write strobes include P103WR# and PPDWR#.
8. Read strobes include P103RD#, P101RD#, and PPDRD#.

82303 Drive Levels/Measurement Points for A.C. Specifications

Input/Output Pulse Widths

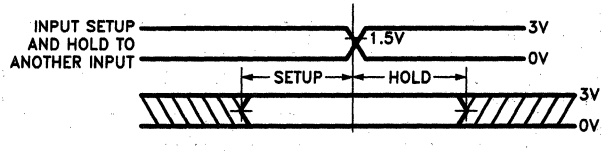


290184-3



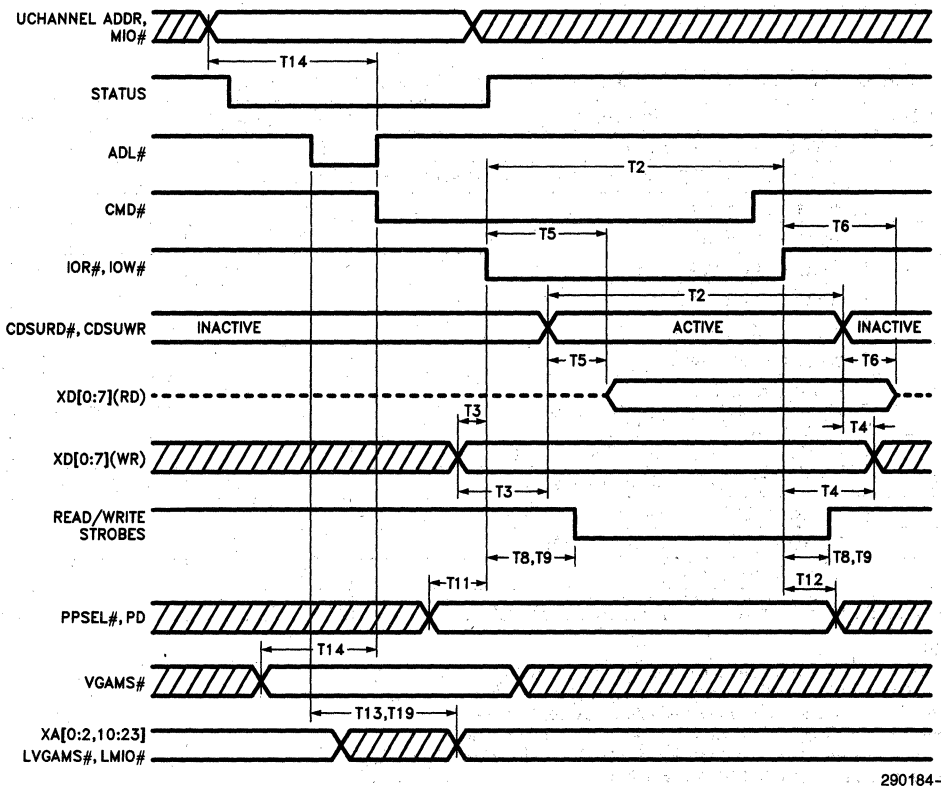
290184-4

Input Setup and Hold to another Input

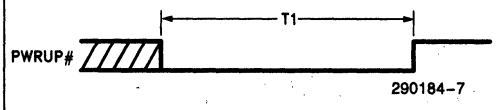


290184-5

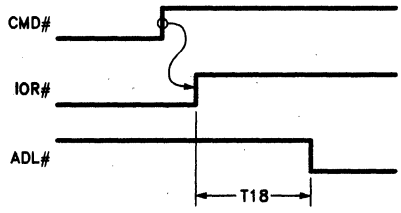
NOTE:
Input Waveforms have $T_R \leq 2.0$ ns from 0.8V to 2.0V.



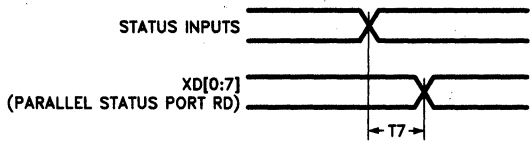
290184-6



290184-7

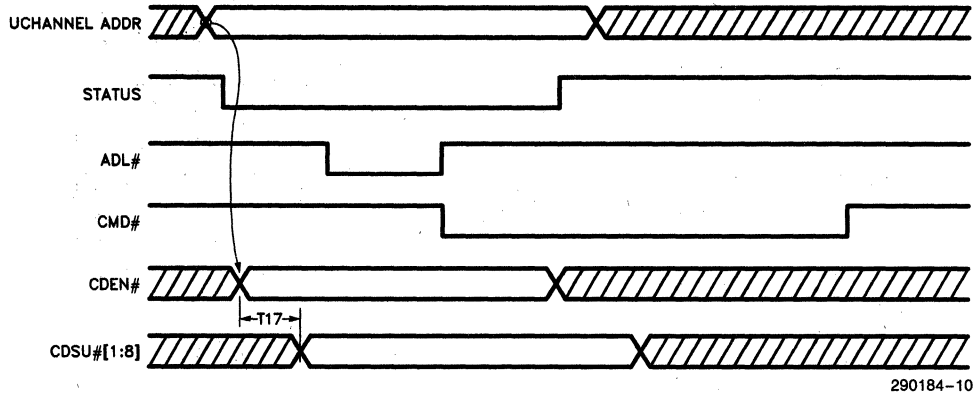


290184-8



290184-9

NOTE:
T7 is the flow through propagation delay, and thus assumes T5A, is not the limiting parameter.



290184-10

APPENDIX
82303 INTERNAL LOGIC DIAGRAMS

These logic diagrams are provided to aid in understanding the basic functionality of the 82303, and should not be used to estimate signal loading, propagation delays, or any other timing behavior.

The clocked latches in the diagrams are functionally equivalent to 7474 type TTL latches. The transparent latches are equivalent to 74373 type TTL latches except that the gate input is active low rather than active high.

The truth table for the combinatorial PAL is as follows:

RD

PPSEL#	IOR#	XA0	XA1	PPDR#	PPCR#	PSRD#
0	0	0	0	0	1	1
0	0	0	1	1	0	1
0	0	1	0	1	1	0

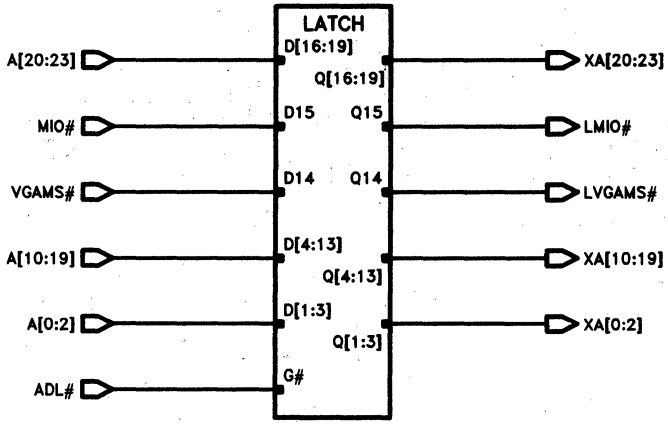
SELPOS	IOR#	XA2	XA1	XA0	PPOS1D#	PPOS13R#
1	0	0	0	1	0	1
1	0	0	1	1	1	0

WR

PPSEL#	IOW#	XA0	XA1	PPDW#	PCWR#
0	0	0	0	0	1
0	0	0	1	1	0

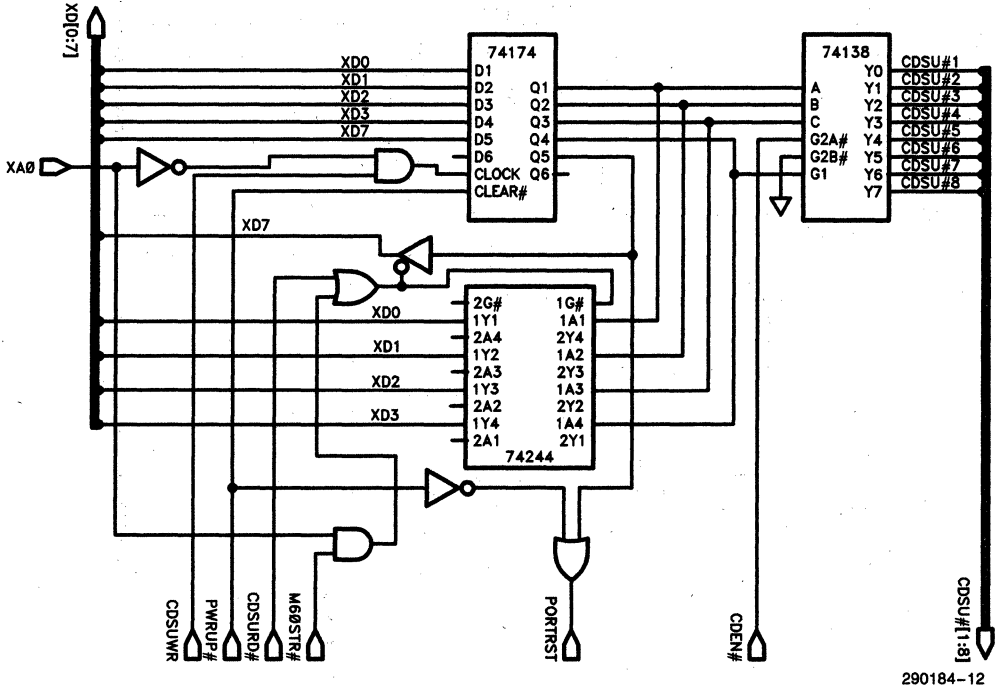
SELPOS	IOW#	XA2	XA1	XA0	POS103WR#
1	0	0	1	1	0

Address Latch
Transparent Latch × 19

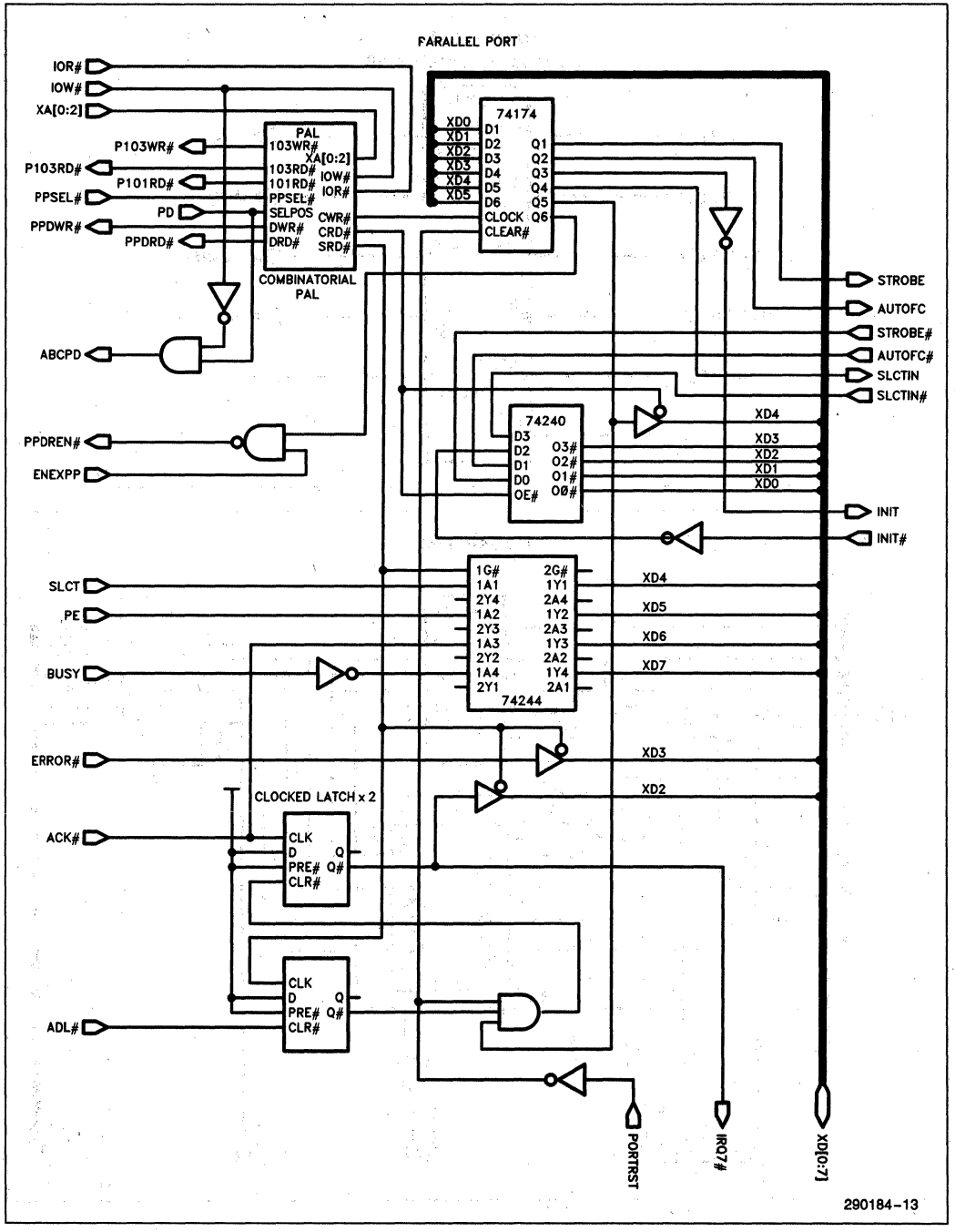


290184-11

Card Setup Port



290184-12

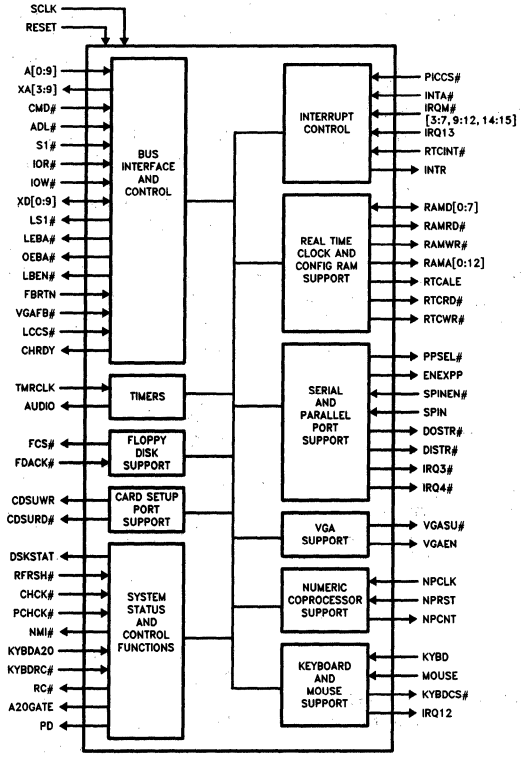

PARALLEL PORT
IOR#
IOW#
XA[0:2]
P103WR#
P103RD#
P101RD#
PPSEL#
PD
PPDWR#
PPDRD#
PAL
103WR#
XA[0:2]
103RD#
IOW#
101RD#
IOR#
PPSEL#
SELPOS
CWR#
DWR#
CRD#
DRD#
SRD#
COMBINATORIAL PAL
ABCPD
PPDREN#
ENEXPP
74174
XD0
XD1
XD2
XD3
XD4
XD5
D1
D2
D3
D4
D5
D6
CLOCK
CLEAR#
Q1
Q2
Q3
Q4
Q5
Q6
74240
D3
D2
D1
D0
OE#
O3#
O2#
O1#
O0#
1G#
1A1
2Y4
1A2
2Y3
1A3
2Y2
1A4
2Y1
2G#
1Y1
2A4
1Y2
2A3
1Y3
2A2
1Y4
2A1
74244
SLCT
PE
BUSY
ERROR#
ACK#
ADL#
CLOCKED LATCH x 2
CLK
D
PRE#
CLR#
Q
Q#
STROBE
AUTOFC
STROBE#
AUTOFC#
SLCTIN
SLCTIN#
INIT
INIT#
PORTRST
IRQ7#
XD[0:7]
290184-13

82304
LOCAL I/O SUPPORT CHIP

- **High-Integration—The 82304, 82303 and 82077 Floppy Disk Controller Replace 50 IC's in IBM Design**
- **Supports I/O Peripherals . . . Keyboard/ Mouse Controller, Serial/Parallel Ports, Configuration RAM, and Real Time Clock**
- **Integrates Two 8259 PIC's and All Associated Logic**
- **Integrates Programmable Timer Counters 0, 2 and 3**
- **Supports VGA Controller on the Local Channel**
- **Integrates the OS/2 Optimized HOT A20 and HOT RESET Functions**
- **Integrates Variety of System Status/ Control Ports and Functions**
- **Low Power CHMOS Technology/132-Pin PQFP Package**

The 82304 Local Channel Support Chip, along with its companion chip (the 82303) and the 82077 Floppy Disk Controller, significantly reduce system cost, design effort, and form factor constraints by replacing 50 IC devices in an equivalent IBM system.

The 82304 integrates logic to support local bus I/O peripherals and the VGA Controller. Also integrated are three programmable timer/counters, two "8259-like" programmable interrupt controllers, and a variety of system status/control ports and functions. Integrated along with the 8259 PIC's is all logic required to make the PIC's Microchannel architecture compatible.



290185-1

November 1988
Order Number: 290185-001

INTRODUCTION

The 82304 is a high integration device intended for Microchannel compatible system designs. It essentially integrates the 82306 Local Channel Support chip, two 8259 Programmable Interrupt Controllers, and a wide assortment of TTL circuitry. The 82304, in conjunction with its sister chip the 82303 and the 82077 Floppy Disk Controller, replaces approximately 50 IC devices in an equivalent IBM system. Included as an appendix to this data sheet is a functional logic diagram of the 82304 that should facilitate understanding of the part. Note that the 82304, 82303 and 82077 integrate a variety of system ports. For programming and register level details, please refer to the IBM Technical Reference Manual, 82077 data sheet, and 8259A data sheet.

BUS INTERFACE AND CONTROL

The Bus Interface and Control unit interfaces the 82304 to the Microchannel and peripheral busses. It inputs the unlatched Microchannel address, latches it for internal use, and makes the latched version available externally for other peripheral bus resources. It also provides signals to control an external 74F543 latching data transceiver that sits between the Microchannel and peripheral data busses. The bus interface unit also provides functions such as cycle extension on behalf of slower peripherals, and support of the Microchannel architecture's system feedback function.

SYSTEM TIMERS

The 82304 integrates the timers required for multitask time slice interrupt (timer 0), audio tone generation (timer 2), and "watch-dog" function (timer 3). These timers are accessed via ports 40, 42, 43, 44, and 47H.

FLOPPY DISK SUPPORT

The 82304 provides the decode signal required by the 82077 Floppy Disk Controller. The decode addresses ports 3F0–3F7H. The 82304 also inputs the 82077's DMA acknowledge in support of the system feedback function.

VGA SUPPORT

The 82304 supports the VGA setup and enable/disable functions. Specifically, bit 5 of integrated port 94H is used to put the VGA into setup mode, and this mode is reflected to the VGA on the 82304's VGASU# pin. Also, the 82304 integrates bit 0 of port 3C3H, which is used to enable/disable the system board VGA subsystem as indicated by the 82304's VGAEN output.

CARD SETUP PORT SUPPORT

The 82304 provides decoded read/write strobes for ports 96–97H. Port 96H is the card setup port, which is integrated on the 82303 chip. Port 97H is currently reserved by IBM.

INTERRUPT CONTROL

The 82304 integrates two 8259 Programmable Interrupt Controllers, and all additional logic required to make these interrupt controllers Microchannel architecture compatible. Specifically, the Microchannel definition requires that interrupts be active low and level sensitive. This allows a wire-OR system implementation. Integrated logic includes inverters for incoming interrupts, as the 8259 treats level sensitive interrupts as active high. Additionally, logic to inhibit the 8259's from being programmed in edge-triggered mode is integrated.

REAL TIME CLOCK AND CONFIGURATION RAM SUPPORT

The 82304 integrates all logic required to support an external battery backed up real time clock chip and static RAM. Note that while the IBM implementation supports a 2K RAM, the 82304 makes provision to support either a 2K or 8K RAM. The real time clock is accessed via ports 70–71H, while the RAM is accessed via ports 74–76H. (RAM data is accessed via port 76H, while ports 74H and 75H serve as an indirect address latch for the RAM.) The 82304 also integrates the logic required to enforce the Microchannel architecture's password security function. Specifically, writes to port 70H are monitored. If a write to 70H is attempting to access offsets 38–3FH in the real time clock chip's onboard RAM, and if the security bit in 92H indicates that these offsets are off limits, then no address latch signal is generated to the real time clock chip.

SERIAL AND PARALLEL PORT SUPPORT

The 82304 provides various functions in support of an external serial port chip (the 16550A), and a par-

allel port (integrated on the 82303 chip). The 82304 provides decoded read/writes strobes for the serial port chip, as well as converting a serial port interrupt into either IRQ3# or IRQ4#, depending on whether the serial port is configured as COMM1 or COMM2. When configured as COMM1, the serial port is decoded at ports 3F8–3FFH. As COMM2, the port is decoded from 2F8–2FFH. Configuration is done via the integrated system setup port 102H.

The 82304 generates a parallel port chip select that maps to LPT1, LPT2, or LPT3, depending on how system setup port 102H is programmed. As LPT1, the parallel port is decoded at ports 3BC, 3BD, 3BE, and 3BFH. LPT2 maps to ports 378, 379, 37A, and 37BH. LPT3 maps to 278, 279, 27A, and 27BH. The 82304 also generates a signal that indicates whether the parallel port is to operate in its normal output-only mode, or its "extended" bi-directional mode. The mode is selected via system setup port 102H.

NUMERIC COPROCESSOR SUPPORT

The 82307 DMA Controller, in response to a software command, issues a pulse to reset the 80387 or 80387SX Numeric Coprocessor. The 82304 inputs this pulse and effectively stretches it out to insure that the 80387 reset input pulse is long enough to meet its internal reset requirements. (Note that the 80387 reset pulse out of the 82304 must be externally synchronized to the 80387 clock so as to convey the system phase to the 80387.) The 82304 also manipulates CHRDY to extend the bus cycle that initiates the reset so as to tie up the CPU until the 80387's reset and initialization requirements are met.

KEYBOARD AND MOUSE SUPPORT

The 82304 provides a chip select for the 8742 Keyboard Controller. This decode maps to ports 60 and 64H. The 82304 also integrates the logic required to both latch and subsequently clear keyboard and mouse interrupts.

SYSTEM STATUS AND CONTROL FUNCTIONS

The 82304 integrates a variety of system status and control functions and ports. Integrated ports include:

Port 61H System Control Port B
Port 92H System Control Port A
Port 91H Card Selected Feedback Register
Port 94H System Board Setup Port
Port 102H System Board POS Port
Port 70H NMI Enable (Write Only)

The functions and register level details of these ports are documented in the IBM Technical Reference.

82304 LOCAL CHANNEL SUPPORT CHIP PIN DEFINITIONS

Symbol	Pin No.	Type	Description
SCLK	50	I	**INPUT CLOCK:** Tied to same clock as host CPU.
RESET	64	I	**SYNCHRONIZED POWER-UP RESET:** Resets 82304 and synchronizes internal clock to system phase.
A[0:9]	85–81, 78–74	I	**MICROCHANNEL ADDRESS:** Address Lines are internally latched.
XA[3:9]	73–67	O	**PERIPHERAL BUS ADDRESS:** These are latched versions of the Microchannel address.
CMD#	97	I	**MICROCHANNEL CMD# INPUT**
ADL#	100	I	**MICROCHANNEL ADL# INPUT**
S1#	1	I	**MICROCHANNEL S1# INPUT**
IOR#, IOW#	96, 95	I	**82304 READ AND WRITE STROBES**
XD[0:7]	94–90, 88–86	I/O	**BI-DIRECTIONAL DATA BUS**
LS1#	131	O	**LATCHED VERSION OF MICROCHANNEL S1# INPUT**
LEBA#, OEBA#	130, 129	O	**EXTERNAL 74543 DATA BUFFER CONTROL SIGNALS:** These signals control data timing on the peripheral data bus.
LBEN#	52	O	**LOCAL (PERIPHERAL) BUS ENABLE:** The 82304 generates this in response to address decodes of peripheral bus ports. It is typically "OR"ed with other system qualifiers to enable the peripheral bus data buffer.

82304 LOCAL CHANNEL SUPPORT CHIP PIN DEFINITIONS (Continued)

Signal Name	Pin No.	I/O	Description
FBRTN	41	I	**SYSTEM FEEDBACK:** This input receives the OR of the system feedback signals of the Microchannel slots. It is internally latched and "OR"ed with other feedback sources, and the result made available via a Port 91H read (Bit 0).
VGAFB#	42	I	**VGA FEEDBACK.**
LCCS#	37	I	**LOCAL CHANNEL CHIP SELECT:** Activated for the I/O address range 0–3FFH (CPU or DMA master) or 100–3FFH (Microchannel Master). LCCS# is internally latched.
CHRDY	5	O	**CHANNEL READY:** The 82304 deasserts CHRDY to extend accesses to certain peripheral bus resources, specifically the keyboard controller, real time clock and serial port. Also, CHRDY is used to tie up the CPU during numeric coprocessor resets.
TMRCLK	45	I	**1.193 MHz CLOCK INPUT:** Drives clock inputs of system timers 0 and 2.
AUDIO	128	O	**OUTPUT OF SYSTEM TIMER 2 GATED BY BIT 1 OF PORT 61H:** It drives the Microchannel audio sum node.
FCS#	46	O	**FLOPPY DISK CONTROLLER (82077) CHIP SELECT:** Responds to I/O range 3F0–3F7H.
FDACK#	49	I	**FLOPPY DISK CONTROL DMA ACKNOWLEDGE:** Internally latched and "OR"ed with other system feedback sources.
VGASU#	44	O	**VGA SETUP:** Puts the VGA into setup mode when active, according to Bit 5 of Port 94H.
VGAEN	43	O	**VGA ENABLE:** Enables/Disables motherboard VGA according to Bit 0 of Port 3C3H.
CDSUWR	126	O	**CARD SETUP WRITE STROBE:** Active High command generated during writes to Port 96–97H.
CDSURD#	125	O	**CARD SETUP READ STROBE:** Active low command generated during reads from Port 96–97H.
DSKSTAT	127	O	**FIXED DISK STATUS:** Controls the fixed disk activity light. It is active when either Bit 6 or Bit 7 of Port 92H is set.
RFRSH#	51	I	**REFRESH CYCLE INDICATOR:** Diagnostics can monitor refresh activity via Bit 4 of Port 61H.
CHCK#	54	I	**MICROCHANNEL CHECK INDICATOR:** Used to report adapter errors.
PCHCK#	55	I	**DRAM PARITY ERROR:** Driven in response to motherboard memory parity errors.
NMI#	53	O	**NON-MASKABLE INTERRUPT REQUEST TO CPU:** This acts as an open drain output that allows for an external wire "OR" with other NMI sources.
KYBDA20	60	I	**A20 GATE SIGNAL OUT OF THE KEYBOARD CONTROLLER:** Internally "OR"ed with the alternate A20 switch incorporated in Bit 1 of Port 92H.
KYBDRC#	58	I	**CPU RESET SIGNAL OUT OF THE KEYBOARD CONTROLLER:** It is internally "OR"ed with the alternate reset function of Bit 0 of Port 92H.
RC#	59	O	**RESET CPU:** Resets CPU via Port 92H (Bit 0) or the KYBDRC# input.
A20GATE	57	O	**A20 GATE SIGNAL:** The "OR" of Bit 1 of Port 92H and the KYBDA20 input.

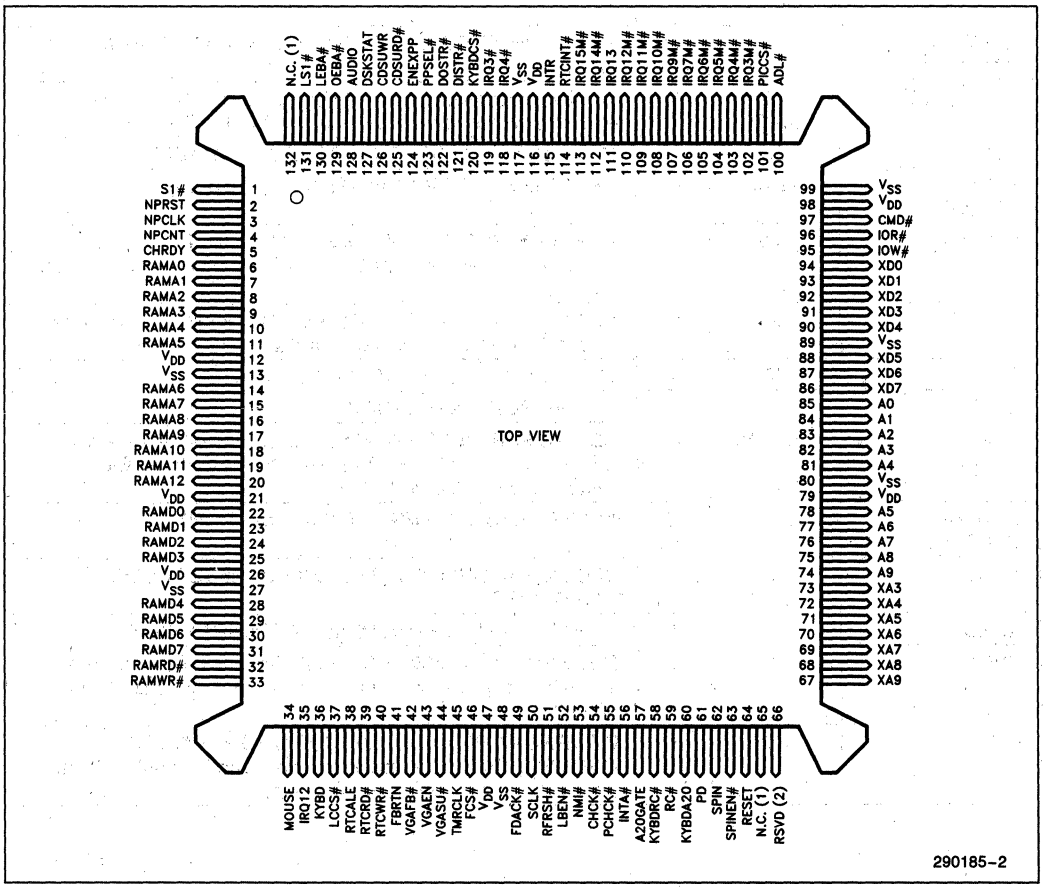
82304 LOCAL CHANNEL SUPPORT CHIP PIN DEFINITIONS (Continued)

Signal Name	Pin No.	I/O	Description
PD	61	O	**POS DECODE:** An active high decode of system board setup ports 100, 101, 103–107H. (Port 102H is integrated on the 82304.)
PICCS#	101	I	**CHIP SELECT FOR THE INTEGRATED 8259 PROGRAMMABLE INTERRUPT CONTROLLERS (PIC)**
INTA#	56	I	**INTERRUPT ACKNOWLEDGE:** Generated by the bus controller during interrupt acknowledge cycles.
IRQM# [3:7, 9:12, 14:15]	102–110, 112–113	I	**MICROCHANNEL INTERRUPT INPUTS.**
IRQ13	111	I	**INTERRUPT INPUT USED TO REPORT NUMERIC COPROCESSOR ERRORS.**
RTCINT#	114	I	**INTERRUPT INPUT FROM REAL TIME CLOCK.**
INTR	115	O	**MASKABLE INTERRUPT REQUEST TO CPU.**
RAMD[0:7]	22–25, 28–31	I/O	**REAL TIME CLOCK AND CONFIGURATION RAM DATA BUS.**
RAMRD#, RAMWR#	32, 33	O	**READ/WRITE STROBES TO CONFIGURATION RAM:** Generated during accesses to Port 76H.
RAMA[0:12]	6–11, 14–20	O	**CONFIGURATION RAM ADDRESS BUS:** Internal RAM address latches are written to via Ports 74–75H.
RTCALE	38	O	**REAL TIME CLOCK ADDRESS LATCH ENABLE.**
RTCRD#, RTCWR#	39, 40	O	**REAL TIME CLOCK READ/WRITE STROBES.**
PPSEL#	123	O	**PARALLEL PORT CHIP SELECT:** Maps to LPT1, LPT2, or LPT3 as controlled by Bits 5 and 6 of system board setup Port 102H.
ENEXPP	124	O	**PARALLEL PORT EXTENDED MODE ENABLE:** This mode is controlled via bit 7 of system board setup Port 102H.
SPINEN#	63	I	**SERIAL PORT INTERRUPT ENABLE.**
SPIN	62	I	**SERIAL PORT INTERRUPT.**
IRQ3#, IRQ4#	119, 118	O	**SERIAL PORT INTERRUPT:** Configured to either COMM1 (IRQ4#) or COMM2 (IRQ3#). Selection is done via Bit 3 of system board setup Port 102H.
DOSTR#, DISTR#	122, 121	O	**WRITE/READ STROBES FOR SERIAL PORT.**
NPCLK	3	I	**CLOCK FOR NUMERIC PROCESSOR RESET PULSE STRETCHER.**
NPRST	2	I	**NUMERIC PROCESSOR RESET REQUEST INPUT**
NPCNT	4	O	**NUMERIC PROCESSOR COUNT:** Numeric processor reset signal typically synchronized externally and fed to 80387 or 80387SX.
KYBD	36	I	**INTERRUPT REQUEST INPUT FROM KEYBOARD CONTROLLER:** It is internally latched, and then subsequently cleared by a keyboard controller read.

82304 LOCAL CHANNEL SUPPORT CHIP PIN DEFINITIONS (Continued)

Signal Name	Pin No.	I/O	Description
MOUSE	34	I	**INTERRUPT REQUEST INPUT FROM KEYBOARD CONTROLLER'S MOUSE PORT:** It is internally latched and subsequently cleared by a keyboard controller read.
KYBDCS#	120	O	**KEYBOARD CONTROLLER CHIP SELECT.**
IRQ12	35	O	**LATCHED VERSION OF MOUSE INPUT INTERRUPT REQUEST.**
V_{DD}	12, 21, 26, 47, 79, 98, 116		**POWER.**
V_{SS}	13, 27, 48, 80, 89, 99, 117		**GROUND.**
NC	65, 132		**NO CONNECT.**
RSVD	66		**RESERVED.**

82304 132-Pin PQFP Pinout



NOTES:
1. N.C. pins must be left not connected.
2. This pin is reserved . . . must be tied to ground in system.

82304 Parametrics

Absolute Maximum Ratings*

Case Temperature Under Bias .... −40°C to +85°C
Storage Temperature ..... −65°C to +150°C
Voltage to any Pin with
Respect to Ground .... −0.3V to +(V_{CC} +0.3)V
DC Supply Voltage (V_{CC}) ..... −0.3V to +7.0V
DC Input Current ..... ±10 mA

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. Electrical Characteristics T_C = 0°C to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Min	Max	Units	Notes
V_{IL}	Input Low Voltage		0.8	V	
V_{IH}	Input High Voltage	2.0		V	
V_{IL}	Input Low Voltage		0.8	V	SCLK
V_{IH}	Input High Voltage	V_{CC}−0.8		V	SCLK
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 4 mA (Note 1)
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 4 mA (Note 1)
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2 mA (Note 2)
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 2 mA (Note 2)
I_{CC}	Power Supply Current		180	mA	No DC Loads
I_{LI}	Input Leakage Current		±10	μA	$V_{SS} < V_{IN} < V_{CC}$
I_{OZ}	TRI-STATE Output Leakage Current		±10	μA	$V_{SS} < V_{OUT} < V_{CC}$

NOTES:
1. DSKSTAT, XA[3:9], XD[0:7].
2. All outputs other than those listed in Note 1.

82304 A.C. Electrical Specifications T_C = 0°C to +70°C, V_{CC} = 5V ±10%

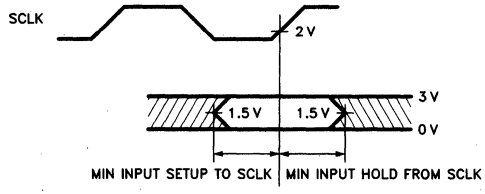
Symbol	Parameter	KIT-16		KIT-20		KIT-25		C_L (pF)	Notes
		Min	Max	Min	Max	Min	Max		
T_1	SCLK Period	31.25		25		20			
T_{2A}	SCLK High/Low Time	12		10		8			
T_{2B}	SCLK High/Low Time	8		6.5		6			
T_3	Reset Setup	10		10		10			
T_4	Reset Hold	3		3		3			
T_5	Reset Pulse Width	500		500		500			
T_6	RC# Pulse Width	75	150	75	150	75	150	50	
T_7	TMRCLK High/Low Time	300		300		300			
T_8	PICCS#, FDACK# Setup	30		30		30			
T_9	PICCS#, FDACK# Hold	0		0		0			
T_{10}	IOR#, IOW#, INTA# Pulse Width	170		170		170			

82304 A.C. Electrical Specifications T_C = 0°C to +70°C, V_{CC} = 5V ±10% (Continued)

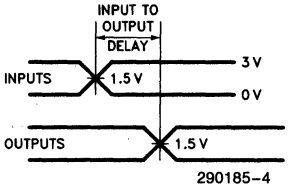
Symbol	Parameter	KIT-16		KIT-20		KIT-25		C_L (pF)	Notes
		Min	Max	Min	Max	Min	Max		
T_{11}	Write Data Setup	25		25		25			
T_{12}	Write Data Hold	10		10		10			
T_{13}	Read Data Valid Delay	0	90	0	90	0	90	100	
T_{14}	Read Data Float Delay	0		0		0		100	
T_{15}	RAMD[0:7] to XD[0:7] Delay		36		36		36	100	
T_{17}	CHRDY Delay	0	80	0	80	0	80	25	
T_{18}	CHRDY Inactive Pulse Width	280		280		230		25	5
T_{19}	Address Decode Delays from ADL# ↓	0	62	0	62	0	62	50	1
T_{20A}	Write Strobe Delays from IOW# ↓	0	40	0	40	0	40	50	2
T_{20B}	CDSUWR, DOSTR# Delays from IOW# ↑	0	35	0	35	0	35	50	
T_{20C}	RTCWR#, RAMWR# Delay from CMD# ↑	0	33	0	33	0	33	50	
T_{21}	Read Strobe Delays	0	40	0	40	0	40	50	6
T_{22}	RTCALE Min Pulse Width	120		120		110		50	3
T_{23}	XA[3:9] Delay from ADL# ↓		35		35		35	100	5
T_{24}	S1#, LCCS#, A[0:9] Setup to ADL# ↑	30		30		30			
T_{25}	FBRTN Setup to CMD# ↓	15		15		15			
T_{26}	VGAFB# Setup to CMD# ↑	15		15		15			
T_{27}	LEBA# Delay from CMD#		26		26		26	25	
T_{28}	OEBA# Delay from CMD# ↓		30		30		30	25	
T_{30}	CHRDY ↓ Delay		38		38		38	25	4
T_{31}	NPCNT ↑ Delay	128 NPCLKS		128 NPCLKS		128 NPCLKS		50	5
T_{32}	CHRDY ↑ Delay	192 NPCLKS		192 NPCLKS		192 NPCLKS		25	5
T_{33}	NPCLK High/Low Time	12		12		12			
T_{34}	LS1# Delay from ADL# ↓		35		35		35	50	

NOTES:
1. Address decodes include FCS#, PPSEL#, KYBDCS#, PD and LBEN#.
2. Write strobes include RTCWR#, COSUWR#, DOSTR#, and RAMWR#.
3. Read strobes include RTCRD#, CDSURD#, DISTR# and RAMRD#.
4. From later or NPRST ↑ or CMD# ↓.
5. Functional Specification . . . Not tested.
6. CMD# ↑ causes RTCWR# ↑ and RAMWR# ↑, while IOW# ↑ causes RAMD[0:7] to float. The 82304 insures that CMD#-to-RAMWR#/RTCWR# is at least 5 ns faster than IOW# to RAMD[0:7] float, assuming loading on RAMD[0:7] is greater than or equal to loading on RAMWR# or RTCWR#. This provides a minimum of 5 ns data hold time for the real time clock and SRAM, assuming CMD# and IOW# reach the 82304 at the same instant. Typically, more than 5 ns is provided, since IOW# is generated from, and thus delayed from CMD#.
7. Specification applies to software reset generated via port 92H.

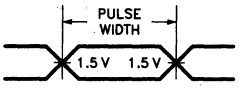
82304 DRIVE LEVELS/MEASUREMENT POINTS FOR A.C. SPECIFICATIONS



290185-3

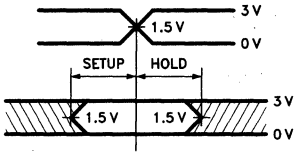


290185-4



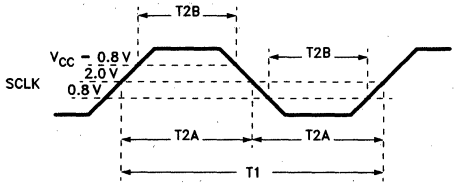
290185-5

Input/Output Pulse Widths,
Input Clock (Other than SCLK)
HIGH/LOW Times

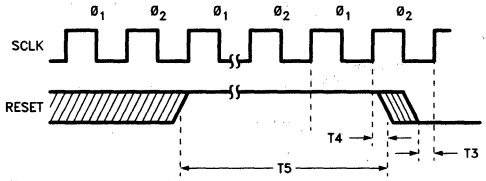


290185-6

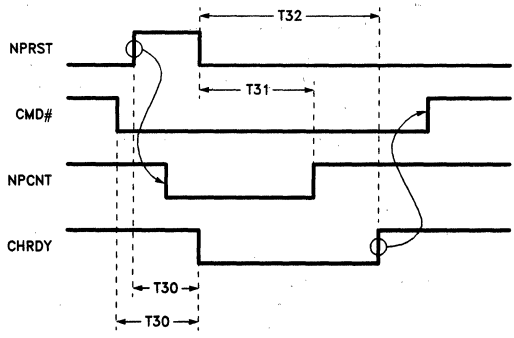
Input Setup and Hold to
another Input (Other than SCLK)



290185-7

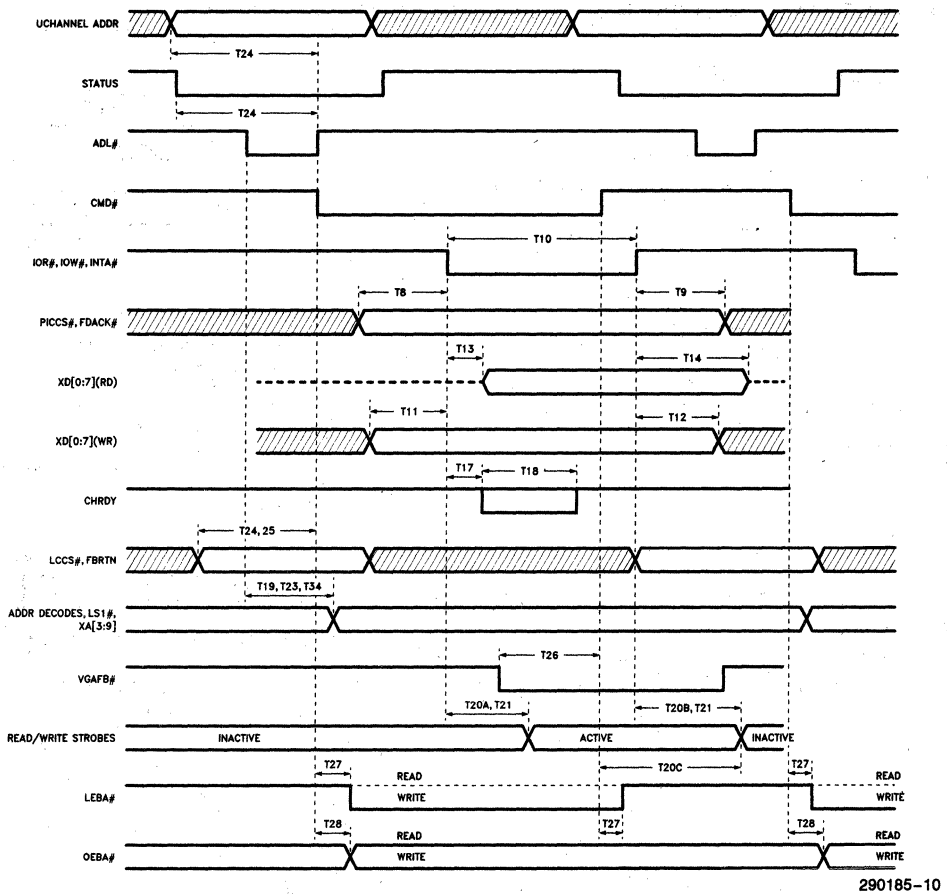


290185-8

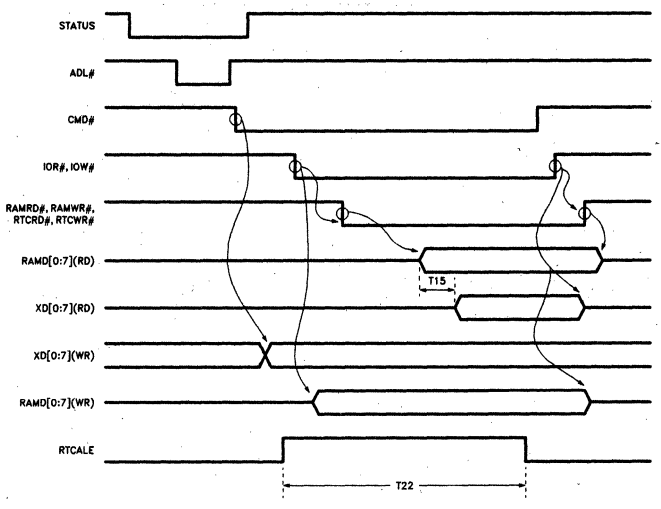


290185-9

NOTE:
Input Waveforms have $T_R \leq 2.0$ ns from 0.8V to 2.0V.



290185-10



290185-11

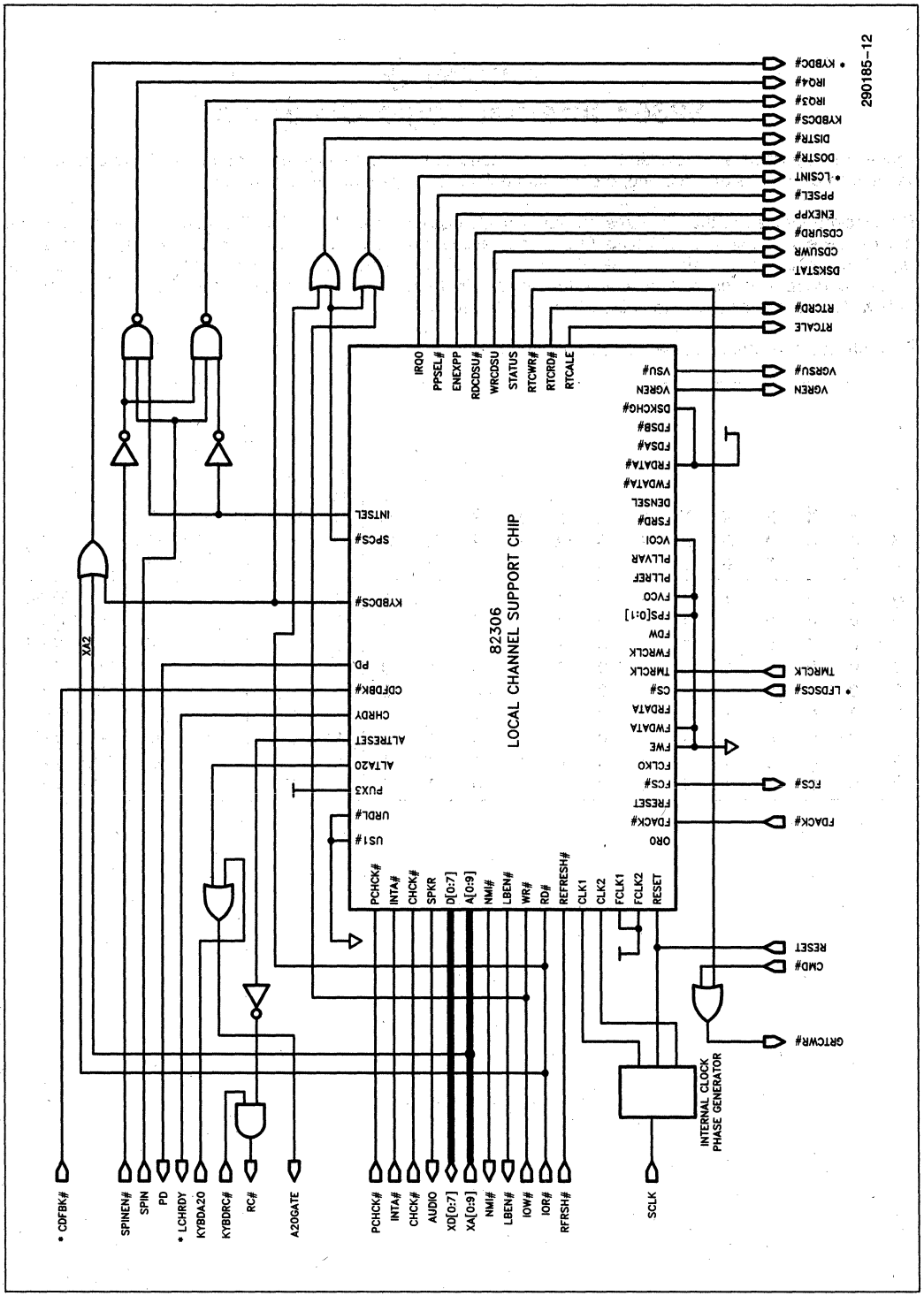
APPENDIX
82304 Internal Logic Diagrams

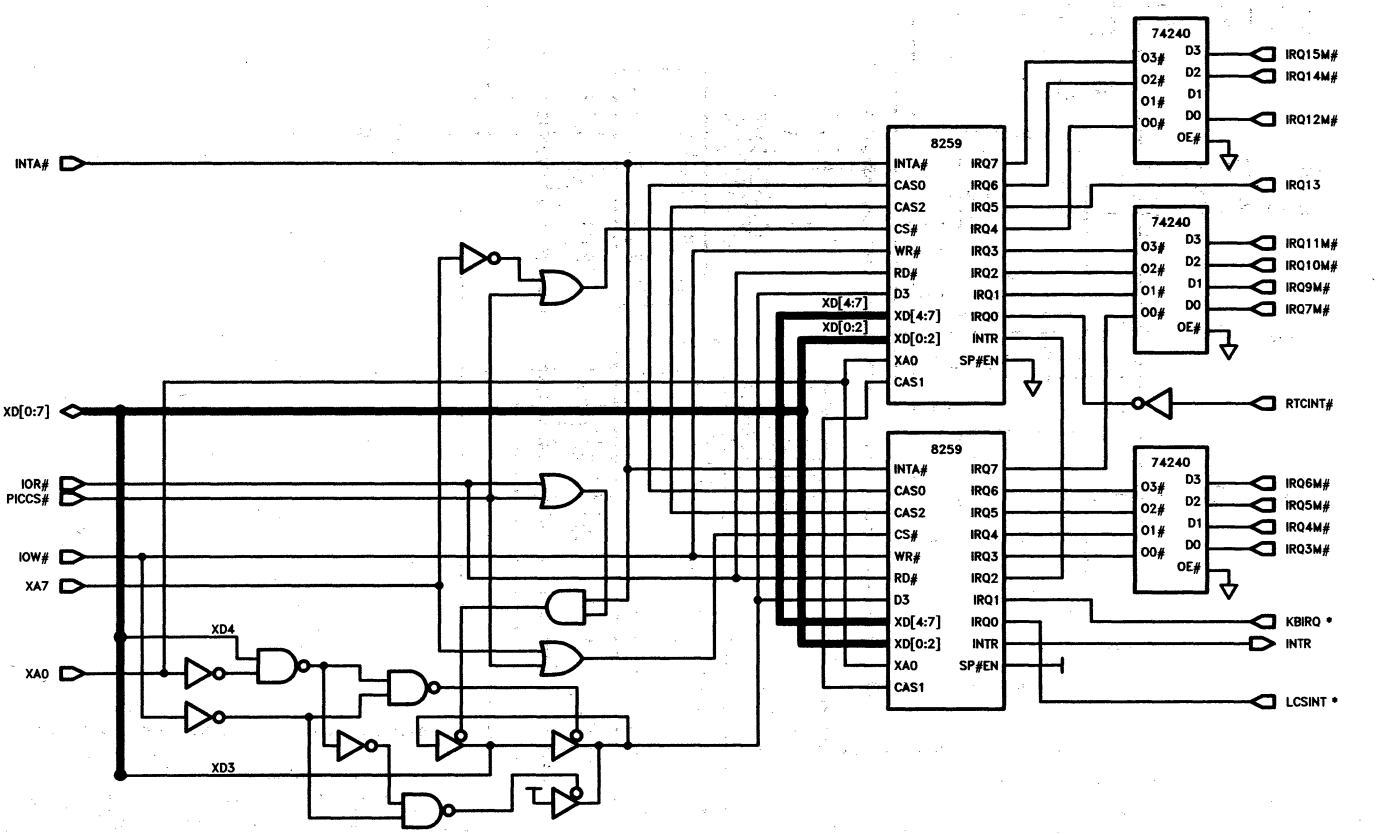
These logic diagrams are provided to aid in understanding the basic functionality of the 82304, and should not be used to estimate signal loading, propagation delays, or any other timing behavior.

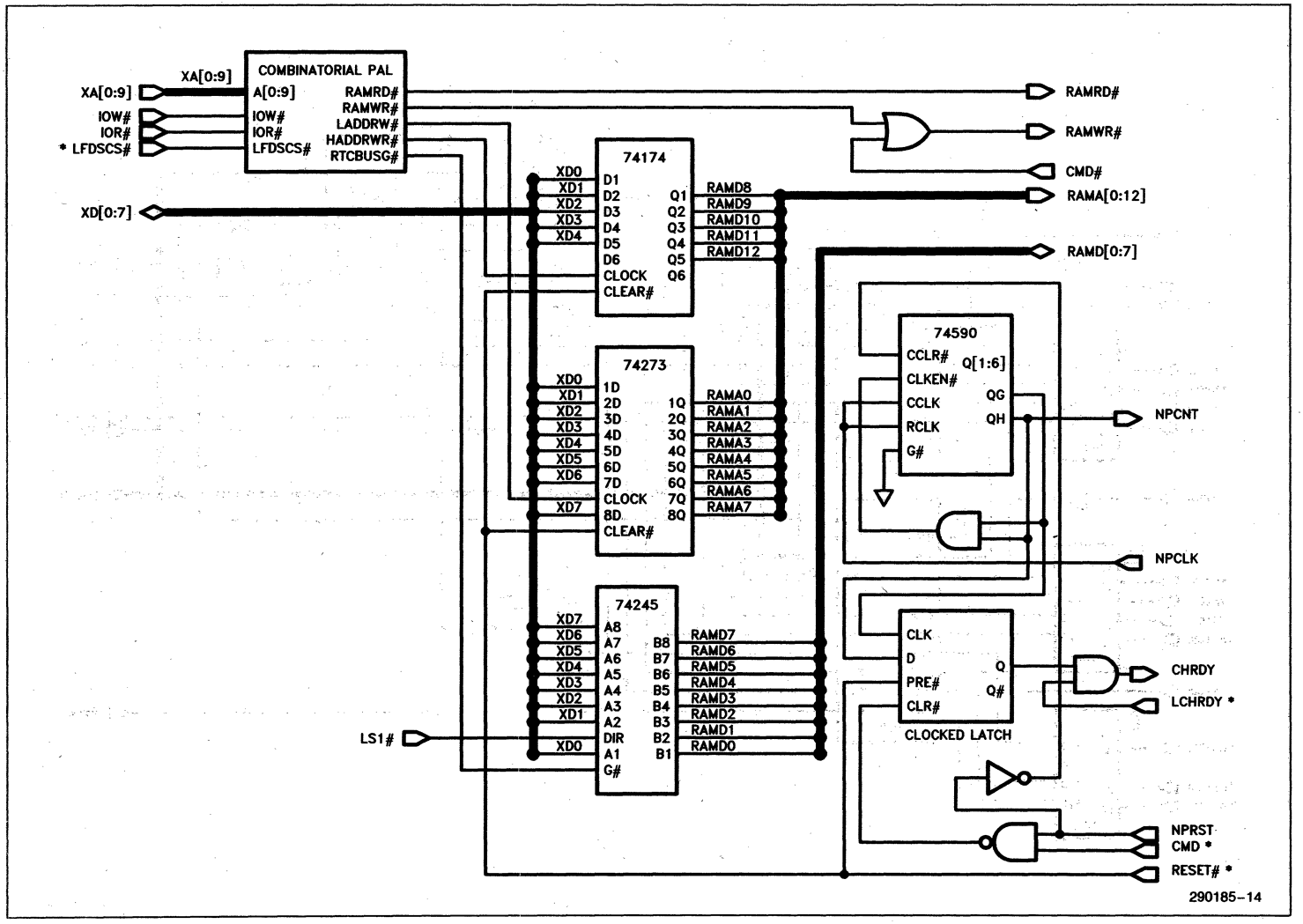
The clocked latches in the diagrams are functionally equivalent to 7474 type TTL latches. The transparent latches are equivalent to 74373 type TTL latches except that the gate input is active low rather than active high. The signals marked with asterisks (*) are not actually available external to the 82304, but simply serve as page-to-page references. Note however, that the XA[0:9] internal address bus is not marked with an asterisk. Only XA[3:9] are available externally, while XA[0:2] are not.

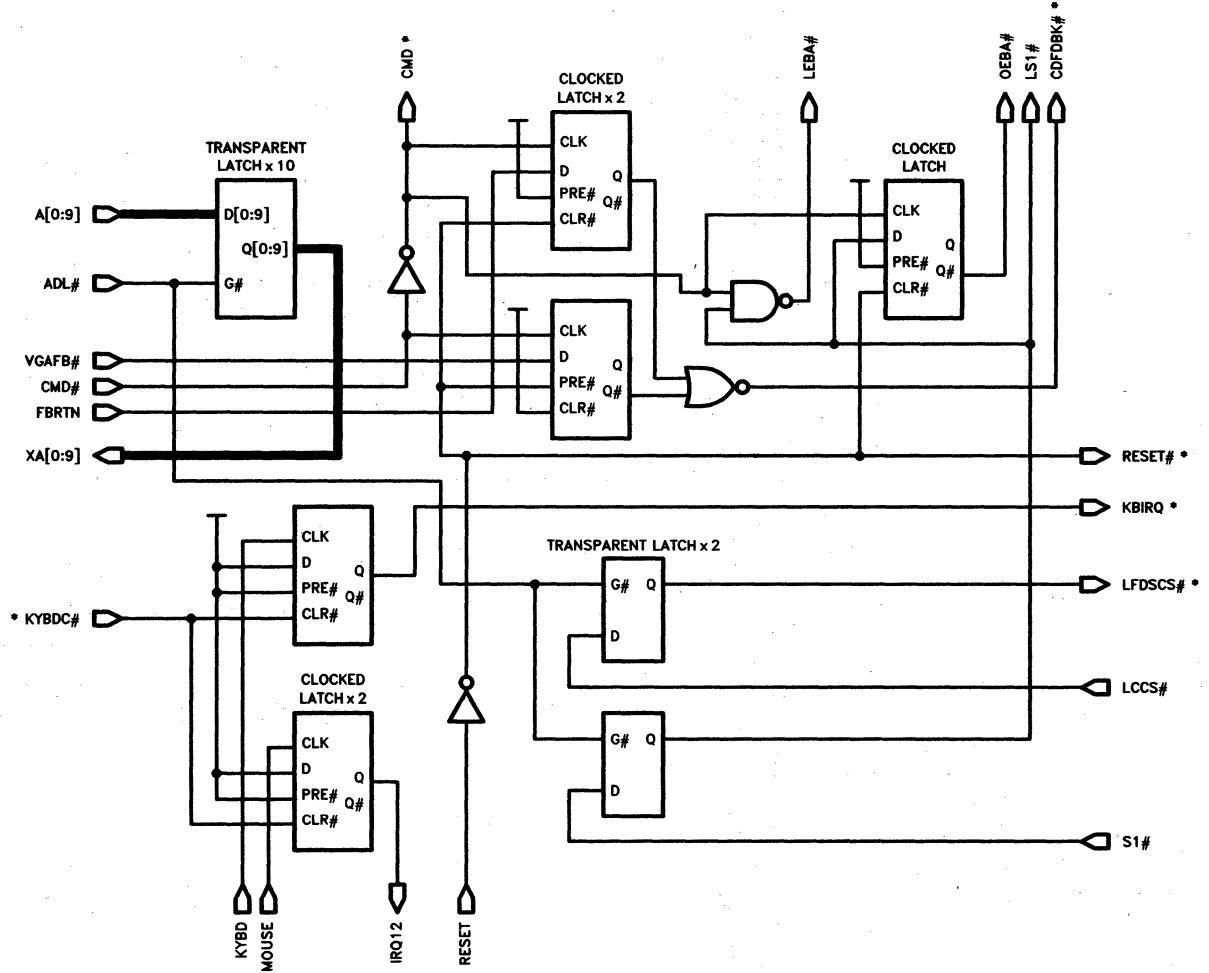
The truth table for the combinatorial PAL is as follows:

LFDSCS#	XA9	XA8	XA7	XA6	XA5	XA4	XA3	XA2	XA1	XA0	IORC#	IOWC#	RAMRD#	RAMWR#	HADDRWR#	LADDRWR#	RTCBUSG#	
0	0	0	0	1	1	1	0	1	1	0	0	1	0	1	1	1	1	76H READ
0	0	0	0	1	1	1	0	1	1	0	1	0	1	0	1	1	1	76H WRITE
0	0	0	0	1	1	1	0	1	0	1	1	0	1	1	0	1	1	75H WRITE
0	0	0	0	1	1	1	0	1	0	0	1	0	1	1	1	0	1	74H WRITE
0	0	0	0	1	1	1	0	0	0	0	1	0	1	1	1	1	0	70H WRITE
0	0	0	0	1	1	1	0	0	0	1	1	0	1	1	1	1	0	71H WRITE
0	0	0	0	1	1	1	0	1	1	0	1	0	1	1	1	1	0	76H WRITE
0	0	0	0	1	1	1	0	0	0	1	0	1	1	1	1	1	0	71H READ
0	0	0	0	1	1	1	0	1	1	0	0	1	1	1	1	1	0	76H READ


290185-12
82306
LOCAL CHANNEL SUPPORT CHIP
INTERNAL CLOCK
PHASE GENERATOR
XA2
* KYBDC#
IR04#
IR03#
KYBDCS#
DISTR#
DOSTR#
* LCSINT
PPSEL#
ENEXPP
CDSURD#
CDSUWR
DSKSTAT
RTCRD#
RTCALE
VGRSU#
VGREN
TMRCLK
* LFDSCS#
FCS#
FDACK#
RESET
CMD#
GRTCWR#
IRQ0
PPSEL#
ENEXPP
RDCDSU#
WRCDSU
STATUS
RTCWR#
RTCRD#
RTCALE
VSU#
VGREN
DSKCHG#
FDSB#
FDSA#
FRDATA#
FWDATA#
DENSEL
FSRD#
VCOI
PLLVAR
PLLREF
FVCO
FPS[0:1]
FDW
FWRCLK
TMRCLK
CS#
FRDATA
FWDATA
FWE
FCLKO
FCS#
FRESET
FDACK#
ORO
INTSEL
SPCS#
KYBDCS#
PD
CDFDBK#
CHRDY
ALTRESET
ALTA20
PUX3
URDL#
US1#
PCHCK#
INTA#
CHCK#
SPKR
D[0:7]
A[0:9]
NMI#
LBEN#
WR#
RD#
REFRESH#
CLK1
CLK2
FCLK1
FCLK2
RESET
* CDFBK#
SPINEN#
SPIN
PD
* LCHRDY
KYBDA20
KYBDRC#
RC#
A20GATE
PCHCK#
INTA#
CHCK#
AUDIO
XD[0:7]
XA[0:9]
NMI#
LBEN#
IOW#
IOR#
RFRSH#
SCLK


INTA#
XD[0:7]
IOR#
PICCS#
IOW#
XA7
XA0
XD4
XD3
8259
INTA#
CAS0
CAS2
CS#
WR#
RD#
D3
XD[4:7]
XD[0:2]
XA0
CAS1
IRQ7
IRQ6
IRQ5
IRQ4
IRQ3
IRQ2
IRQ1
IRQ0
INTR
SP#EN
XD[4:7]
XD[0:2]
74240
O3#
O2#
O1#
O0#
D3
D2
D1
D0
OE#
IRQ15M#
IRQ14M#
IRQ12M#
IRQ13
IRQ11M#
IRQ10M#
IRQ9M#
IRQ7M#
RTCINT#
IRQ6M#
IRQ5M#
IRQ4M#
IRQ3M#
KBIRQ *
INTR
LCSINT *
290185-13


COMBINATORIAL PAL
XA[0:9]
A[0:9]
IOW#
IOR#
LFDSCS#
RAMRD#
RAMWR#
LADDRW#
HADDRWR#
RTCBUSG#
* LFDSCS#
XD[0:7]
74174
D1
D2
D3
D4
D5
D6
CLOCK
CLEAR#
Q1
Q2
Q3
Q4
Q5
Q6
RAMD8
RAMD9
RAMD10
RAMD11
RAMD12
74273
1D
2D
3D
4D
5D
6D
7D
8D
1Q
2Q
3Q
4Q
5Q
6Q
7Q
8Q
RAMA0
RAMA1
RAMA2
RAMA3
RAMA4
RAMA5
RAMA6
RAMA7
74245
A8
A7
A6
A5
A4
A3
A2
DIR
A1
G#
B8
B7
B6
B5
B4
B3
B2
B1
RAMD7
RAMD6
RAMD5
RAMD4
RAMD3
RAMD2
RAMD1
RAMD0
LS1#
RAMD[0:7]
74590
CCLR#
CLKEN#
CCLK
RCLK
G#
Q[1:6]
QG
QH
NPCNT
NPCLK
CLK
D
PRE#
CLR#
Q
Q#
CLOCKED LATCH
CHRDY
LCHRDY *
NPRST
CMD *
RESET# *
RAMRD#
RAMWR#
CMD#
RAMA[0:12]
290185-14


CMD *
CLOCKED LATCH x 2
LEBA#
OEBA#
LS1#
CDFDBK# *
TRANSPARENT LATCH x 10
CLOCKED LATCH
A[0:9]
D[0:9]
Q[0:9]
ADL#
G#
CLK
D
PRE#
CLR#
Q
Q#
VGAFB#
CMD#
FBRTN
XA[0:9]
RESET# *
KBIRQ *
TRANSPARENT LATCH x 2
LFDSCS# *
* KYBDC#
LCCS#
CLOCKED LATCH x 2
S1#
KYBD
MOUSE
IRQ12
RESET

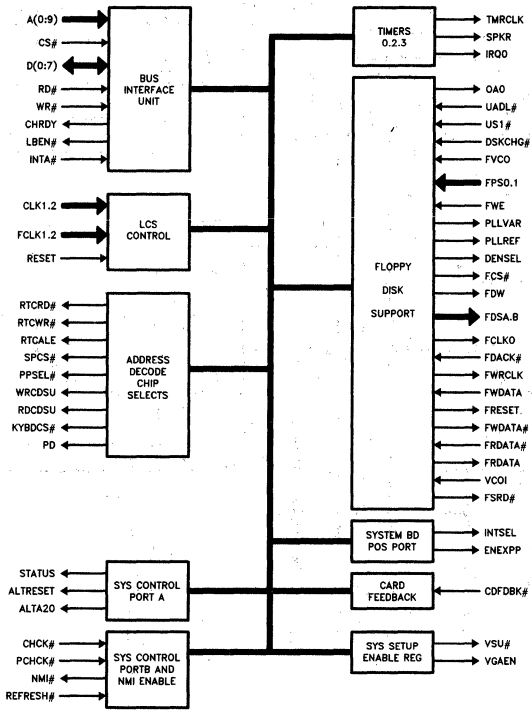
290185-15

82306
LOCAL CHANNEL SUPPORT CHIP

- **Supports I/O Peripherals on the Local Channel**
- **Supports VGA Controller on the VGA Graphics Channel**
- **Floppy Disk Sub-System Support**
  - **— 8272A Interface for IBM Micro Channel Compatible 3½" Drives (Dual Speed Drives −250/500 Kbps)**
  - **— 82072 Interface for IBM Micro Channel Compatible 3½" Drives (250/500 Kbps) and AT Compatible 5¼" Drive (250/300/500 Kbps)**
- **Integrated Programmable Timer/ Counters (0, 2, 3)**
- **Integrates System Registers and Ports**
- **Low Power CHMOS Technology**
- **100-Pin Plastic Quad Flat Pack Packaging**
  (See Packaging Guide Order # 231369)

The 82306 Local Channel Support chip is the register level implementation of the equivalent VLSI device in the IBM Micro Channel systems. It provides FDC interface to support IBM compatible 3½" disk drives when used with 8272A FDA and 3½" & 5¼" (AT Compatible) disk drives when used with the 82072 FDC.

The 82306 also has integrated I/O ports and registers for miscellaneous system board functions, Integrated Address decoder for generaing chip selects for the I/O devices on the Local Channel and 8254 like programmable timers (0, 2, 3) to support speaker tone generation, watch-dog timer and periodic interrupts.



290183-1

October 1988
Order Number: 290183-001

FLOPPY DISK CONTROLLER INTERFACE

The Floppy Disk Controller (FDC) function of the 82306 Local Channel Support chip has two FDC interfaces: 8272A FDC interface as used in IBM Micro Channel systems Model 50/60 and 80, and 82072 FDC interface for value-added performance and compatibility.

The 82306 Local Channel Support chip integrates glue logic required to support the 8272A and the 82072 Floppy Disk Controllers. The FDC interface includes the pre-compensation logic, digital portion of the read data separator logic, and status registers (03F1H, 03F2H, 03F7 ports). The integrated prescaler supports 250 Kbits/sec and 500 Kbits/sec as required in the IBM Micro Channel compatible system. With the 82072 FDC, however, an additional 300 Kbits/sec data stream is supported for interfacing to a standard AT 5¼" drive.

The Floppy Disk subsystem support includes generation of the chip select for the FDC when ports 03F4H or 03F5H are referenced. The Floppy status register (Read only) is implemented externally on the motherboard. To read this register, the 82306 generates the decoded read signal for address 03F0H.

The clock for 8272A is generated by the 82306 Local Channel Support chip, from an external 16 MHz frequency source. The 82072 utilizes its own fixed clock source of 24 MHz.

LOCAL CHANNEL ADDRESS DECODER

The 82306 Local Channel Support provides the Address decoding for the following devices and addresses.

- 8742 Keyboard Interface Chip
- Serial Port
- Parallel Port
- Ports 096H and 097H
- Real-Time Clock (Read, Write and Address Latch Enable)

SYSTEM TIMERS 0, 2, 3

The timers used for periodic interrupt (timer 0), tone generation for audio (timer 2) and watch-dog function (timer 3) are integrated on the 82306 Local Channel Support chip. Timer 0 and 2 are identical in their functionality as the timers in IBM PC/AT, XT and PC systems. The watch-dog timer 3 provides error detection capability and via BIOS, the watch-dog function can be enabled and disabled.

These timers can be programmed via ports 40, 42, 43, 44 and 47.

INTEGRATED SYSTEM REGISTER AND PORTS

The 82306 Local Channel Support chip has the following registers and ports integrated on the chip:

- System Control Port A & B (092H and 061H)
- Card Selected Feedback Register (091H)
- System Board POS Port (102H)
- Port 070H (Write Only)
- System Board Set Up (094H)

The POS register space as defined in the Micro Channel architecture is 100H to 107H. The 82306 integrates 102H on chip and generates PD signal during set-up mode for external implementation of POS ports 100, 101 and 103–107H. POS port 102H is programmed during set-up to enable serial, parallel port and the Floppy Disk Controller.

VGA ENABLE PORT 3C3H

When bit 0 of port 3C3H is programmed as 1, the VGA sub-system is enabled. The chip enable VGAEN is deactivated when a zero is written to the bit. On power up or reset, the VGA sub-system is enabled.

I/O DEVICE MAP

The following table lists the ports and registers integrated on the 82306 Local Channel Support chip.

I/O Register Address	Function
03F1H	Floppy Status Register B
03F2H	Floppy Digital Output Register
03F7H	Digital Input/Config. Register
061H	System Control Port B
092H	System Control Port A
091H	Card Selected Feedback Register
102H	System Board POS Port
070H	NMI Enable (Write Only)
094H	System Board Set-Up
3C3H	VGA Enable Port
40H, 42H, 43H, 44H, 47H	System Timer Ports

For programming and register level details, please refer to IBM technical reference manual.

82306 Local Channel Support Chip Pin Definitions

Signal Name	Pin Number	I/O	Description
A<0:9>	98–89	I	Address inputs.
D<0:7>	86–79	B	Bi-directional data bus.
CS#	78	I	Device chip select.
RD#, WR#	70, 69	I	I/O read and write command inputs.
REFRESH#	67	I	Refresh request generated by the DMA Controller.
RESET	5	I	System power-up reset.
CHRDY	6	O	Channel Ready signal. Driven low (not ready) by the Local Channel Support to extend accesses to its internal ports and to other local I/O bus devices that require a longer cycle time.
CDFDBK#	7	I	Latched card feedback signal.
TMRCLK	9	I	1.193 MHz clock input generated by the Address Bus Controller. It drives the clock inputs of system timers 0 and 2.
SPKR	10	O	Output of system timer 2 gated by bit 1 of Port 61H. It drives the Micro Channel audio sum node.
CLK1, CLK2	72, 3	I	Clock inputs.
FCLK1, FCLK2	71, 4	I	16 MHz (tied high for 82072 interface), clock inputs for 8272A interface.
VSU#	23	O	VGA Setup. It puts the VGA into set-up mode when low.
VGAEN	8	O	VGA chip enable. (Active High) A low level disables the VGA subsystem.
ENEXPP	22	O	Parallel port "extended mode" enable. When high, the parallel port can function bi-directionally. When low, the port is in "compatible mode" i.e., write only.
SPCS#	31	O	Serial port chip select. Can be mapped to COMM1 or COMM2.
INTSEL	32	O	Selects either IRQ3# (COMM2) or IRQ4# (COMM1) to serve as the interrupt request for the serial port.
STATUS	33	O	Floppy Disk Active Status
RTCWR# RTCRD# RTCALE	34 36 35	O	Write, Read and Address latch enable signal to the real time clock chip.
FSRD#	37	O	Read command for the Read-only floppy status port 3F0H.
WRCDSU	40	O	Write CD Setup Register. Active high write command generated on writes to port 96H (adapter enable setup register).
RDCDSU#	49	O	Read CD setup register. Active low signal reads port 96H.
LBEN#	42	O	Local Bus Engage. It is "OR"ed with the LBEN# output of the DMA/CACP to become one of the qualifiers that enable the data buffer between Micro Channel Bus and motherboard I/O bus. The LCS enable this buffer for accesses to the local I/O bus.
KYBDCS#	43	O	8742 keyboard controller chip select.
ALTRESET	44	O	Processor reset under software control. Except for a shorter reset pulse width, it is identical in function to the reset generated under software control by the 8742. (Optimized for switching between Real-mode and Protected-mode tasks.)

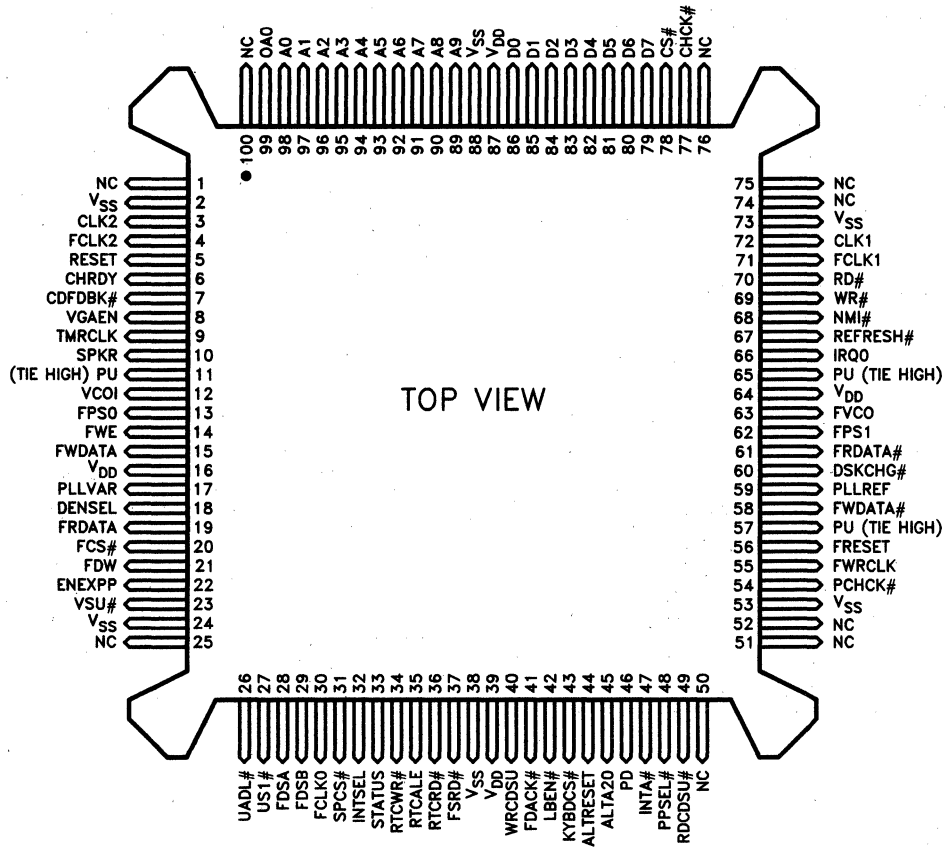
82306 Local Channel Support Chip Pin Definitions (Continued)

Signal Name	Pin Number	I/O	Description
ALTA20	45	O	Alternate A20 bit-Controls address bit A20 in a manner similar to the way it is controlled by the 8742. (Optimized for switching between Real-mode and Protected-mode tasks).
PD	46	O	POS Decode output. When the system board is in setup mode, this output acts as an (active high) chip select for system board POS ports 100H, 101H, and 103H through 107H. (POS port 102H is integrated on the Local Channel Support.)
INTA#	47	I	Interrupt acknowledge generated by the Bus controller.
PPSEL#	48	O	Parallel port chip select. Can be programmed to map the parallel port to LPT1, LPT2, or LPT3.
PCHCK#	54	I	DRAM Parity Error. It is driven by the Bus Controller upon detection of a motherboard DRAM parity error.
IRQ0	66	O	System Timer 0 timeout Interrupt Request.
NMI#	68	O	NMI request to the CPU. This is an open drain output that allows for an external wire "OR" with other NMI sources.
CHCK#	77	I	Micro Channel channel check indicator, for reporting adapter errors.
FPS0, FPS1	13, 62	I	Low and high order pre-compensation select bits. They are driven by the 8272A floppy disk controller. (Note that the pre-compensation logic is integrated on the Local Channel Support). Tied to GND when using 82072.
FWE	14	I	Write Enable for floppy. Generated by the FDC to enable the write data stream to disk.
FWDATA	15	I	FDC output write data stream. It goes through the LCS integrated pre-compensation logic, and then is fed to the drive over the LCS WRDATA# output. In 82072, the FWDATA is inverted and fed into the drive controller.
FWDATA#	58	O	Pre-compensated write data stream to disk for 8272. For 82072 it is simply an inverted output of FWDATA
FRDATA#	61	I	Read data stream from disk.
FRDATA	19	O	Buffered read data output to the FDC. No connect for 82072.
FDSA, FDSB	28, 29	O	Drive select/motor enable outputs.
DENSEL	18	O	Density Select.
FCS#	20	O	Device select for the FDC.
FDACK#	41	I	DMA acknowledge to the FDC from the DMA/CACP.
FCLKO	30	O	8272A clock. It is 8 MHz for high density, or 4 MHz for low density. Not required for 82072.
FWRCLK	55	O	8272A write clock input. It oscillates at twice the data rate; i.e., 1 MHz for a rate of 500 Kbits/sec and 500 KHz for a rate of 250 Kbits/sec. No connect for 82072.
FRESET	56	O	FDC Reset.
DSKCHG#	60	I	Disk changed signal.
VCOI	12	I	Buffered output of the 4024 voltage controlled oscillator. Grounded for 82072.

82306 Local Channel Support Chip Pin Definitions (Continued)

Signal Name	Pin Number	I/O	Description
PLLVAR	17	O	Divided down version of VCOI. It is fed back and used for phase comparison against the PLLREF output. No connect for 82072.
PLLREF	59	O	Phase Lock Loop Reference Clock. No connect for 82072.
FVCO	63	I	Valid Read Data Stream Indicator. It is driven by the 8272A, and defines a valid read data stream. Grounded for 82072.
FDW	21	O	Data window input of 8272A. It defines the valid sample points in the read data stream. No connect for 82072.
OA0	99	O	Output A0 signal for use by the 82072 disk controller. No connect for 8272A.
UADL#	26	I	Micro Channel Address decode latch. An Active low signal used to latch US1# on the trailing edge for support of the OA0 signal. Grounded for 8272A.
US1#	27	I	Micro Channel status bit 1. Used to distinguish a write operation from a read operation for support of the OA0 signal. Grounded for 8272A.
PU	11, 57, 65	I	Pull Up
V_{DD}	16, 39, 64, 87		Power
V_{SS}	2, 24, 38, 53, 73, 88		Ground
NC	1, 25, 50, 51, 52, 74, 75, 76, 100		No Connect

82306 Local Channel Support Chip



290183-2

NOTES:
PU = Pull-Up
NC = No Connect
—Pull-Up Resistor Value = 2K to 10K
—No more than three nodes to a single pull-up resistor.

82306 PARAMETRICS

ABSOLUTE MAXIMUM RATINGS*

Case Temperature under Bias .... −40°C to +85°C
Storage Temperature ..... −65°C to +150°C
Voltage to Any Pin with Respect to Ground ..... −0.3V to (V_{CC}+0.3)V
DC Supply Voltage (V_{CC}) ..... −0.3V to +7.0V
DC Input Current ..... ±10 mA

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS

T_C = 0°C to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Min	Max	Units	Conditions
V_{IL}	Input Low Voltage		0.8	V	
V_{IH}	Input High Voltage	2.0		V	
V_{IL}	Input Low Voltage		0.8	V	CLK1, CLK2
V_{IH}	Input High Voltage	V_{CC} − 0.8		V	CLK1, CLK2
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 2 mA
I_{CC}	Power Supply Current		180	mA	No DC Loads
I_{LI}	Input Leakage Current		±10	μA	$V_{SS} < V_{IN} < V_{CC}$
I_{OZ}	Tri-State Output Leakage Current		±10	μA	$V_{SS} < V_{OUT} < V_{CC}$

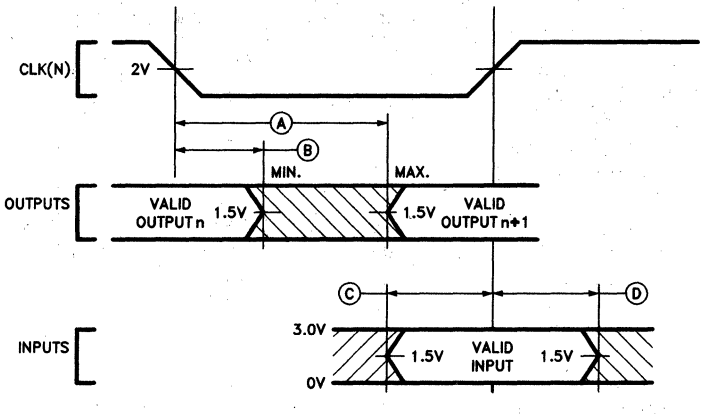
82306 LCS A.C. SPECS

T_C = 0°C to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Kit 16 MHz		Kit 20 MHz		Kit 25 MHz		C_L	Notes
		Min	Max	Min	Max	Min	Max	(pF)	
T1	CLK1, CLK2 LOW TIME	15		15		14			
T2	CLK1, CLK2 NON-OVERLAP	4		4		0			
T3	FCLK1, FCLK2 LOW TIME	10		10		10			
T4	FCLK1, FCLK2 NON-OVERLAP	10		10		10			
T5	RESET PULSE WIDTH	500		500		500			
T6	ALTRESET PULSE WIDTH	75	150	75	150	75	150	75	
T7	TMRCLK HIGH/LOW TIME	300		300		300			
T8	A9–A0, CS#, FDACK# SETUP	30		30		30			
T9	A9–A0, CS#, FDACK# HOLD	10		10		10			
T10	RD#, WR#, INTA# PULSE WIDTH	170		170		170			
T11	WRITE DATA SETUP	25		25		25			
T12	WRITE DATA HOLD	0		0		0			
T13	READ DATA VALID DELAY	0	50	0	50	0	50	75	
T14	READ DATA FLOAT DELAY	0	35	0	35	0	35	75	
T15	CHRDY DELAY	0	80	0	80	0	80	75	5
T16	CHRDY INACTIVE PULSE WIDTH	230		230		180		75	5, 10
T17	ADDRESS DECODE DELAYS	0	50	0	50	0	50	75	1
T18	WRITE STROBE DELAYS	0	40	0	33	0	30	75	2
T19	READ STROBE DELAYS	0	40	0	40	0	40	75	3
T20	RTCALE MIN PULSE WIDTH	120		120		110		75	10
T21	DATA SETUP TO INTA#	25		25		25			
T22	DATA HOLD FROM INTA#	5		5		5			
T23A	FCLKO HIGH TIME	45	63	45	63	45	63	75	6
T23B	FCLKO HIGH TIME	90	150	90	150	90	150	75	6
T23C	FCLKO VALID DELAY		35		35		35	75	
T24A	FWRCLK HIGH TIME	200	300	200	300	200	300	75	
T24B	FWRCLK VALID DELAY		35		35		35	75	
T25	FWDATA# PULSE WIDTH	350		350		350		75	
T26	FRDATA# PULSE WIDTH	40		40		40		75	
T27	FRDTA PULSE WIDTH	125		125		125		75	
T28A	FDW SAMPLE PERIOD	1 µs		1 µs		1 µs		75	4, 7
T28B	FDW SAMPLE PERIOD	2 µs		2 µs		2 µs		75	4, 7
T29	VCOI FREQUENCY	4 MHz		4 MHz		4 MHz			4
T30A	PLLVAR,PLLREF FREQUENCY	1 MHz		1 MHz		1 MHz		75	4, 7, 8
T30B	PLLVAR,PLLREF FREQUENCY	500 KHz		500 KHz		500 KHz		75	4, 7, 8
T31	US1# SETUP TO UADL#	25		25		25			9

NOTES:
1. Address decode delays include SPCS#, LBEN#, KYBDCS#, FCS#, PD, PPSEL#, and OA0. (OA0 supports 82072 interface.)
2. Write Strobe delays include RTCWR# and WRCDSU.
3. Read Strobe delays include RTCRD#, FSRD#, and RDCDSU#.
4. Typical values . . . not tested.
5. LCS extends cycles to the 8242 keyboard controller, serial port, real time clock, and 8272 Floppy Disk Controller.
6. T23A applies to FCLKO = 8 MHz (500 KBPS Data Rate).
   T23B applies to FCLKO = 4 MHz (250 KBPS Data Rate).
7. T28A and T30A apply to 500 KBPS. The T28B and T30B apply to 250 KBPS.
8. VCOI/4 for 500 KBPS. VCOI/8 for 250 KBPS. PLLREF applies only when PLL is locked.
9. 82072 Support.
10. Functional Spec . . . not tested.

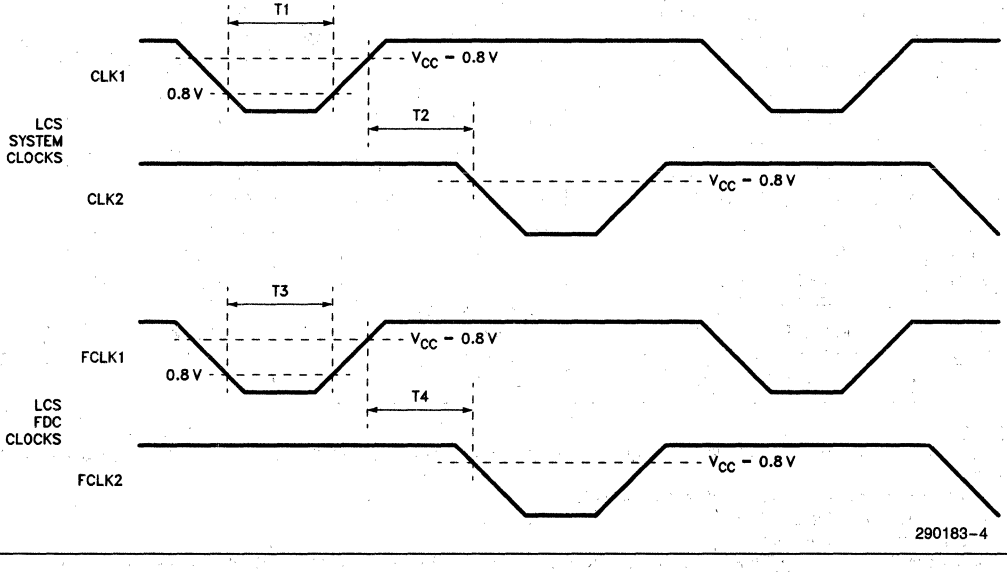
DRIVE LEVELS AND MEASUREMENT POINTS FOR A.C. SPECIFICATIONS



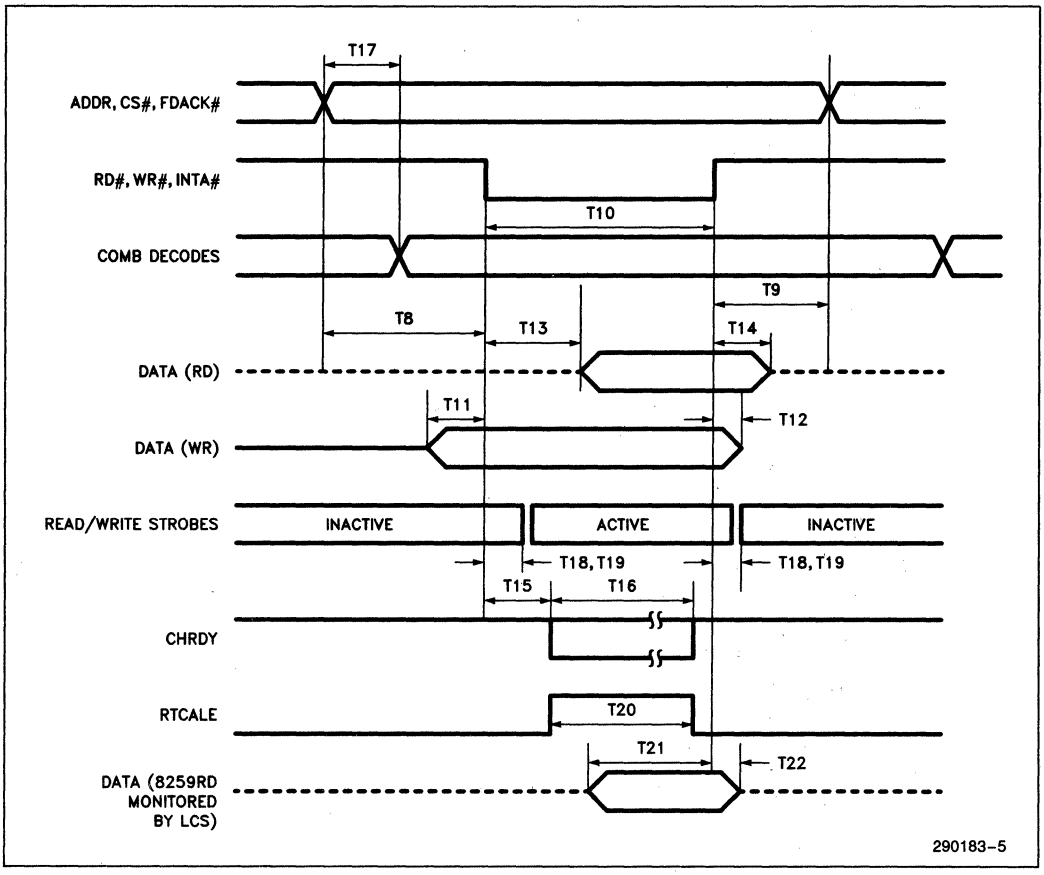
LEGEND:
A. Maximum Output Delay Specification.
B. Minimum Output Delay Specification.
C. Minimum Input Setup Specification.
D. Minimum Input Hold Specification.

Input waveforms have tr ≤ 2.0 ns from 0.8V to 2.0V.

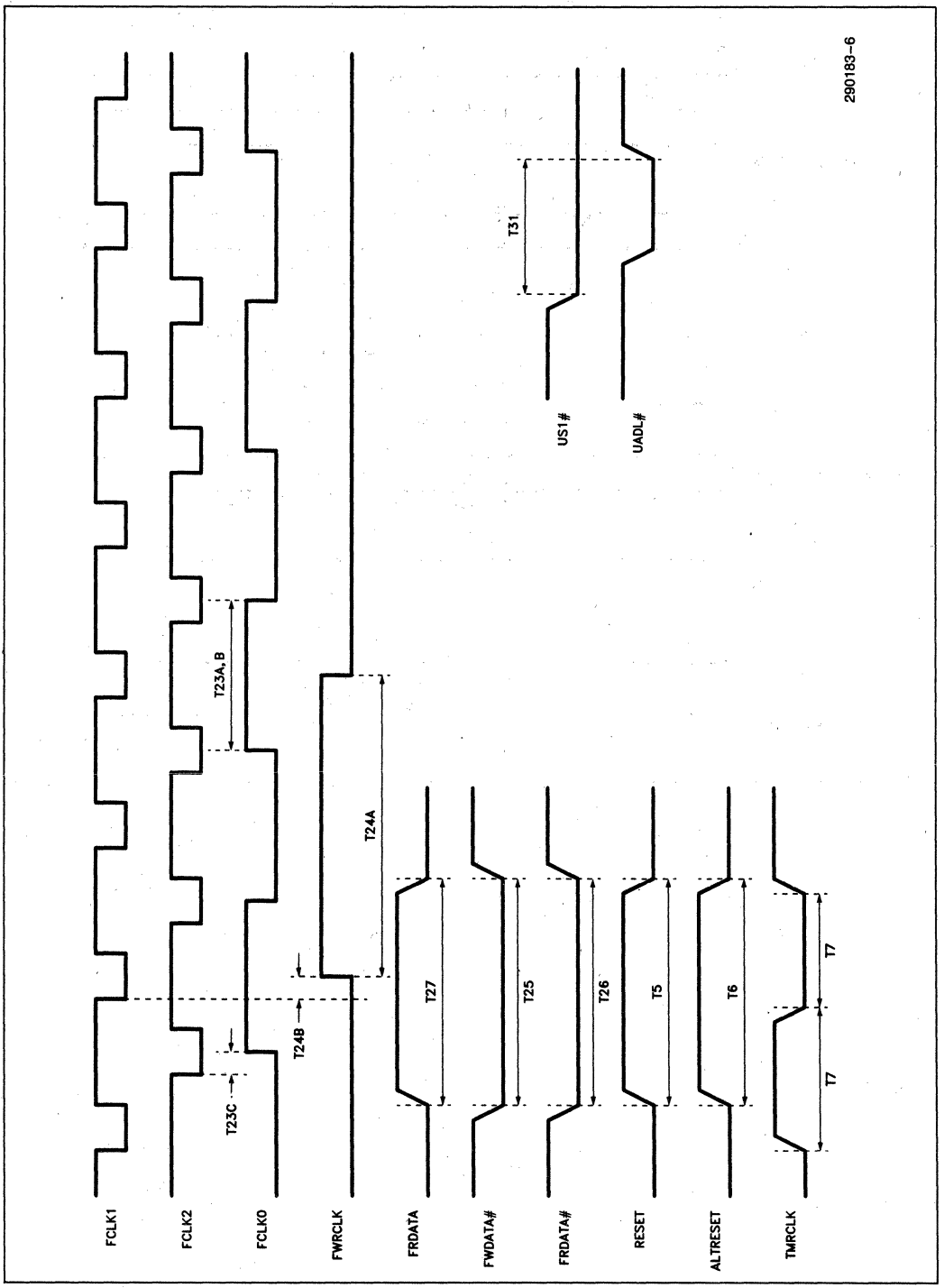
LCS CLOCKS



SYSTEM INTERFACE



MISCELLANEOUS TIMINGS



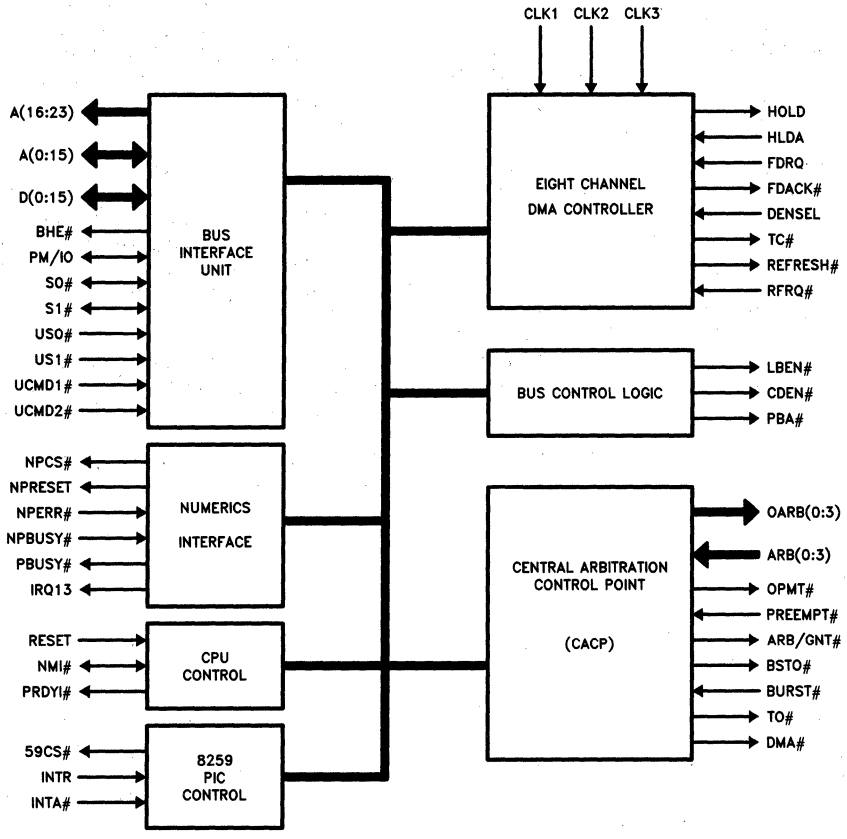
82307
DMA/Micro Channel ARBITRATION CONTROLLER

- **8 Channel DMA Controller (8/16-Bit)**
- **Integrated Central Arbitration Control Point**
- **Refresh Address Generation/Cycling**
- **Numerics Co-processor Interface**
- **Address Decoding**
  - **— Numeric Coprocessor**
  - **— Interrupt Controller**
  - **— POS Address Space for Expansion Slots**
- **Low Power CHMOS Technology**
- **132-Pin Plastic Quad Flat Pack Packaging**

  (See Packaging Spec., Order # 231369)

The 82307 DMA/Micro Channel Arbitration Controller is a register level implementation of the equivalent VLSI device in IBM Micro Channel systems. The Central Arbitration Control Point (CACP) as defined in the Micro Channel Architecture for bus arbitration is integrated in this VLSI device.

The 82307 also integrates the Address decoder logic for generating decodes for numeric coprocessor, Interrupt controllers and POS address space.



290186-1

November 1988
Order Number: 290186-001

DMA FUNCTION

The 82307 features eight 8/16-bit channels, 24-bit addressing capability, and operates in two-cycle transfer mode as defined in the Micro Channel architecture. The DMA controller owns the bus for both halves of the transfer cycle.

The DMA function in the 82307 also supports the motherboard Floppy Disk Controller. Upon receiving the DMA request from the FDC, the DMA controller arbitrates for the Micro Channel bus on behalf of the FDC. The FDC is acknowledged once the bus is granted to initiate the data transfer.

Micro Channel ARBITRATION

The other major function of the 82307 DMA controller is to provide Micro Channel Arbitration. It provides full Micro Channel bus arbitration capability according to the 18-level priority scheme. During normal operation priority 0–15 are assigned with 15 being the lowest priority for the CPU. Priority level −1 which has the higher priority than 0 is also assigned to CPU (switched from 15 to −1) during NMI error recovery. The highest priority −2 is used for refresh.

The bus arbitration priority is asserted via the ARB0–ARB3 signals by the requesting masters to gain control. Bus granting is assigned by the priority level. During an arbitration cycle no Micro Channel master is allowed to drive the bus.

The bus can be preempted by the 82307 when arbitrating on behalf of DMA, or when it is requested to run a refresh cycle, or to respond to NMI error recovery. The preempting of the bus can also be initiated by Micro Channel masters.

The CACP Control Port 090H is integrated on chip.

Micro Channel REFRESH ADDRESS GENERATION/CYCLING

The actual refresh request is generated by the 82309 Address Bus Controller. Upon receiving this request the 82307 DMA Controller gains bus control and executes the refresh cycle. Address generation for the Micro Channel refresh cycle is generated by the 82307 DMA controller.

NUMERICS COPROCESSOR INTERFACE

The 82307 DMA controller supports the numerics coprocessor interface. It provides software transparency required to interface an 80387 to 80386 processor. Ports F0H and F1H are integrated on the 82307.

The numerics coprocessor interface support includes the chip select decode for coprocessor internal register accesses of addresses F8H, FAH and FCH. The 82307 also alerts the CPU of any coprocessor error output generated by asserting the interrupt request IRQ13.

ADDRESS DECODER

The Address Decoder logic decodes the chip select for the 8259 Interrupt Controllers and generates POS Address Space output for addresses 100H through 107H to support the Card set-up signals for the expansion slots.

The chip select output is for both 8259s in the system, so it must be externally gated with local channel address bit A7 to select each actual device.

For programming and register level details, please refer to IBM PS/2 Technical Reference Manual.

82307 DMA/Micro Channel Arbitration Controller Pin Definitions

Signal Name	Pin Number	I/O	Description
A<0:23>	13-2, 130-124, 122-118	A0-A15 B A16-A23 O	Processor local address bus. A16-A23 are output only and are driven when the DMA controller is bus master. A0-A15 are bi-directional. They are inputs when the CPU is master, allowing the CPU access to the chip's internal ports. They are outputs when the DMA is master.
D<0:15>	18-21, 23-31, 35-37	B	Processor local data bus. When the DMA is master, it drives this bus in a write cycle, and samples it during a read cycle. When the CPU is master, the bus is used to access DMA's internal registers.
S0#, S1#	87, 85	B	CPU or DMA cycle status indicators. The DMA drives these signals when it is bus master. When a slave, the DMA inputs these signals to track CPU cycles.
BHE#	84	O	Byte high enable. It is driven when the DMA owns the bus and tristated otherwise. This signal ties directly to the CPU BHE# output in an 80386SX machine.
PM/IO#	81	B	CPU memory / I/O indicator. The DMA drives PM/IO# when bus master, and inputs it when it is slave.
US0#, US1#	43, 44	I	Micro Channel status pins. Generated by the Bus controller when the CPU or DMA is master. When a slot-resident master owns the bus, it generates US0M# and US1M#, and the DMA inputs these so as to recognize when the slot-resident master relinquishes the bus. (The slot-resident master end-of-transfer is recognized when US0M#, US1M#, the channel CMD# signal, and the channel BURST# signal are all negated.)
OARB0-OARB3	58-55	O	DMA/CACP arbitration bus outputs. These signals are driven by the DMA/CACP to arbitrate on behalf of a floppy disk service at priority level 2.
PBA#	107	O	Processor Bus Access. The signal indicates a CPU bus access to the numeric coprocessor or to one of the DMA/CACP registers.
ARB3-ARB0	59-62	I	DMA/CACP arbitration bus inputs. These signals tie directly to the Micro Channel. All competing masters including the DMA/CACP drive these during an arbitration cycle, and the master with the highest priority takes control of the Micro Channel after the arbitration cycle is complete.
OPMT#	92	O	Preempt Bus Master. DMA/CACP drives this output whenever it wishes to preempt the current bus master. This can occur when arbitrating on behalf of a DMA channel service or when arbitrating on behalf of a refresh request, or when arbitrating on behalf of the CPU so as to let it respond to a NMI (non-maskable interrupt) request.
PREEMPT#	45	I	Wired "OR" of the PREEMPT# signals from all Micro Channel masters, including the DMA/CACP (PMTO#). It signifies that a master wishes to force an arbitration cycle.
ARB/GNT#	63	O	Arbitration Cycle indicator. The DMA/CACP drives this Micro Channel signal high to signify an arbitration cycle. During the arbitration cycle, all competing masters drive their priorities onto the arbitration bus (ARB03-ARB00). The falling edge of ARBGNT# signifies the end of the arbitration cycle, at which time the master with the highest priority takes control of the bus. If no master competes for the bus, the pullups on ARB03-ARB00 will read binary 1111 by default, which is the normal operating priority of the CPU.

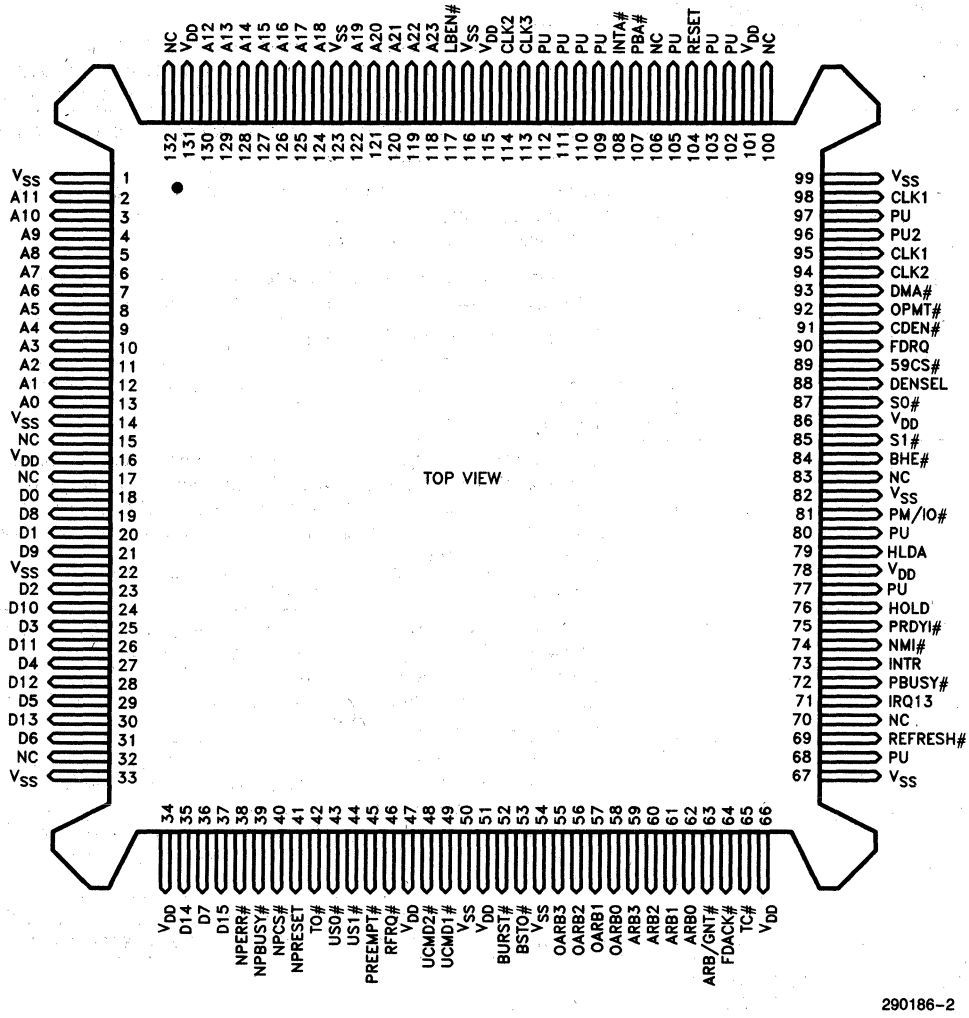
82307 DMA/Micro Channel Arbitration Controller Pin Definitions (Continued)

Signal Name	Pin Number	I/O	Description
BSTO#	53	O	Burst Output. The DMA/CACP drives this output in order to own the Micro Channel for multiple cycles. Specifically, since all PS/2 DMA cycles are two-cycle, BSTO# is driven to allow the DMA controller to own the bus for both halves of a two-cycle transfer.
BURST#	52	I	Burst Request Input. It is an input to the CACP from the current master wishing to own the bus for multiple cycles. It is derived by "OR"ing the Micro Channel BURST# signal with the DMA/CACP BSTO# signal.
HOLD, HLDA	76, 79	(HOLD = O) (HLDA = I)	Hold/Hold Acknowledge to the CPU.
FDRQ, FDACK#	90, 64	(FDRQ = I) (FDACK# = O)	Floppy DMA Request, Acknowledge signals. The motherboard FDC requests DMA service via FDRQ. In response, the DMA/CACP arbitrates for the Micro Channel on behalf of the floppy disk system. Once the DMA/CACP has gained control of the bus, it acknowledges the FDC via FDACK#.
TC#	65	O	DMA Transfer Complete.
UCMD1#, UCMD2#	49, 48	I	Micro Channel Command Inputs. These inputs are driven directly by the Micro Channel CMD# signal.
RFRQ#	46	I	Refresh Cycle Request from the Address Bus Controller.
REFRESH#	69	O	Refresh Cycle Signal. The DMA/CACP drives this output active during refresh cycles. This signal is buffered to become the Micro Channel REFRESH# signal.
59CS#	89	O	Interrupt Controller Chip Select (8259s). Note that this output is activated if either interrupt controller is selected. It is then externally gated with the local I/O channel address bit A7 to distinguish between controller 1 and controller 2.
DENSEL	88	I	Density Selected for the motherboard FDC
CDEN#	91	O	Card Setup Enable. During system setup, a bit pattern is written to port 96H to select a particular slot for configuration. CDEN# enables the decode of these bits to send an active CDSETUP# signal to the selected slot. CDEN# is simply a combinatorial (non-clocked) decode of ports 100H–107H.
CLK1, CLK2, CLK3	95, 98, 94, 114, 113	I	Clock Inputs
RESET	104	I	Power-up System Reset
INTR, INTA#	73, 108	I	Interrupt Request/Acknowledge. The PS/2's 8259 based interrupt system generates interrupt requests to the CPU via INTR. In response, the CPU fetches the appropriate interrupt vector from the interrupt controller in an interrupt acknowledge cycle. The Bus controller decodes the CPU status outputs, and drives INTA# to identify a CPU interrupt acknowledge cycle. The DMA/CACP monitors this activity via its INTR and INTA# inputs. In response to INTA#, the DMA/CACP drives LBEN# so as to enable the 8259 vector onto the Micro Channel. The CACP uses INTR to ensure that the CPU has an opportunity to service an interrupt within one "fairness" cycle; i.e., it prevents the CPU from being totally locked out by higher priority arbiters.

82307 DMA/Micro Channel Arbitration Controller Pin Definitions (Continued)

Signal Name	Pin Number	I/O	Description
NMI#	74	B	Non-Maskable Interrupt to force arbitration cycle to allow CPU bus ownership. As an output, this appears to the system as an open drain, which allows for an external wire "OR" with other NMI sources.
PRDYI#	75	I	Processor Ready Input. The Bus controller generates this signal to terminate CPU and DMA cycles.
NPCS#	40	O	Chip select for the Numeric Coprocessor. It is an unlatched decode that acts as a chip select for CPU accesses to the numeric coprocessor's internal registers.
NPRESET	41	O	Numeric Coprocessor Reset. It resets the numeric coprocessor either upon a system reset or under software control.
NPERR#	38	I	Numeric Coprocessor Error Input. It is an input from the numeric coprocessor error output. The DMA/CACP uses it to generate an interrupt request (IRQ13) to inform the CPU of a coprocessor error.
NPBUSY#	39	I	Numeric Coprocessor Busy
PBUSY#	72	O	Processor Busy Output. It drives the CPU numeric coprocessor busy input. It is activated normally when the coprocessor is busy executing an instruction, but is also activated when a coprocessor error is detected. The CPU will not attempt to utilize the coprocessor as long as PBUSY# is active.
IRQ13	71	O	Numeric Coprocessor Error Interrupt
LBEN#	117	O	Local Bus Enable. This signal is used to enable the data buffers between the Micro Channel and local I/O bus. It is activated for decoded accesses to the 8259 interrupt controllers, as well as for interrupt acknowledge cycles. It is also driven during the DMA acknowledge cycle to the FDC.
DMA#	93	O	DMA/CACP as the Bus Master. It is driven low at the end of an arbitration cycle (ARB/GNT# falling) to indicate that the DMA controller has gain control of the Micro Channel. It is negated during arbitration cycles, and is negated when either the CPU or slot-resident master owns the bus. It is also negated during refresh cycles.
TO#	42	O	Bus Timeout signal. The DMA/CACP also issues an NMI to the CPU in response to the Bus timeout, and forces an arbitration cycle.
V_{DD}	16, 34, 47, 51, 66, 78, 86, 101, 115, 131		Power
V_{SS}	1, 14, 22, 33, 50, 54, 67, 82, 99, 116, 123		Ground
NC	15, 17, 32, 70, 83, 100, 106, 132		No Connect
PU	68, 77, 80, 97, 102, 103, 105, 109–112	I	Pull Up
PU2	96	I	Pull Up. This input must have its own pullup.

82307 DMA/Micro Channel Arbitration Controller



290186-2

NOTES:
Important!! No other node allowed to share pull-up with PU2.
NC = No Connect
PU = Pull-Up
—Pull-Up Resistor Valve = 2K to 10K
—No more than three nodes to a single pull-up resistor.

82307 PARAMETRICS

ABSOLUTE MAXIMUM RATINGS*

Case Temperature under Bias .... −40°C to +85°C
Storage Temperature ..... −65°C to +150°C
Voltage to Any Pin with Respect to Ground ..... −0.3V to (V_{CC}+0.3)V
DC Supply Voltage (V_{CC}) ..... −0.3V to +7.0V
DC Input Current ..... ±10 mA

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS

T_C = 0°C to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Min	Max	Units	Conditions
V_{IL}	Input Low Voltage		0.8	V	
V_{IH}	Input High Voltage	2.0		V	
V_{IL}	Input Low Voltage		0.8	V	CLK1, CLK2, CLK3
V_{IH}	Input High Voltage	V_{CC} − 0.8		V	CLK1, CLK2, CLK3
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 2 mA
I_{CC}	Power Supply Current		180	mA	No DC Loads
I_{LI}	Input Leakage Current		±10	μA	$V_{SS} < V_{IN} < V_{CC}$
I_{OZ}	Tri-State Output Leakage Current		±10	μA	$V_{SS} < V_{OUT} < V_{CC}$

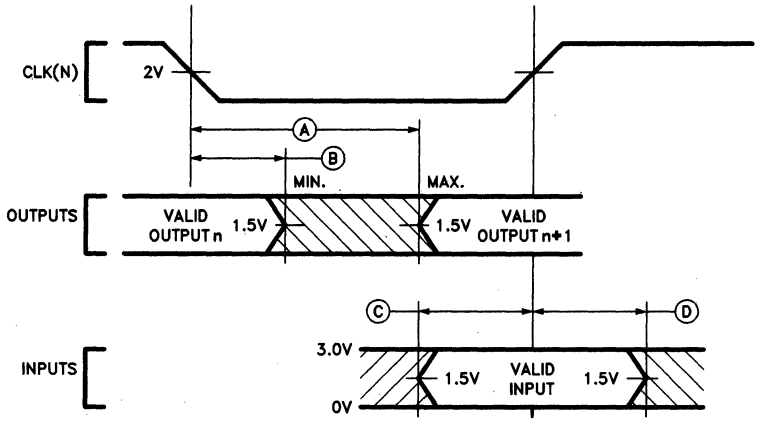
82307 DMA CONTROLLER A.C. SPECS

T_C = 0°C to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Kit 16 MHz		Kit 20 MHz		Kit 25 MHz		C_L (pF)	Notes
		Min	Max	Min	Max	Min	Max		
T1	CLK1, CLK2, CLK3 LOW TIME	15		15		14			
T2	CLK(N) NON-OVERLAP TIME	4		4		0			
T3	RESET(IN), NPRESET(OUT) PULSE WIDTH	500		500		500		50	
T4	TO# PULSE WIDTH	60		60		60		50	
T5	A23–A0, PM/IO#, BHE#, REFRESH# DELAY	4	35	4	35	4	32	75	
T6	A23–A0, PM/IO#, BHE#, STATUS FLOAT DELAY	4	40	4	40	4	40	75	
T7	WRITE DATA VALID DELAY	2	35	2	35	2	28	75	
T8	WRITE DATA FLOAT DELAY	2	40	2	35	2	35	75	
T9	READ DATA SETUP TIME	15		13		11			
T10	READ DATA HOLD TIME	4		4		4			
T11A	STATUS VALID DELAY (TPHL)	4	25	4	25	4	20	75	
T11B	STATUS VALID DELAY (TPLH)	2	35	2	35	2	30	75	
T12	ADDR-TO-STATUS SETUP	35		27		22		75	
T13A	PRDYI# SETUP TIME	25		20		20			2
T13B	PRDYI# SETUP TIME	8		8		8			2
T14	PRDYI# HOLD TIME	5		5		5			2
T15	A15–A0, PM/IO# SETUP TIME	35		35		35			
T16	A15–A0, PM/IO# HOLD TIME	10		10		10			
T17	STATUS SETUP TIME	26		26		24			
T18	STATUS HOLD TIME	10		10		10			
T19	WRITE DATA SETUP TIME	25		25		25			1
T20	WRITE DATA HOLD TIME	25		25		25			1
T21	READ DATA VALID DELAY	2	100	2	100	2	100	75	
T22	READ DATA FLOAT DEALY	8	40	8	35	8	35	75	
T23	HOLD DELAY	2	35	2	32	2	32	50	
T24	ARB/GNT# DELAY FROM EOT	30		30		30		50	
T25	ARB/GNT# PULSE WIDTH	300		300		300		50	
T26	OARB3–OARB0 DELAY		32		32		32	50	
T27	OPMT# INACTIVE DELAY		35		35		35	50	
T28	BSTO# DELAY	2	40	2	40	2	40	50	
T29	TC# DELAY	2	36	2	33	2	33	50	
T30	FDACK#, DMA# DELAY	2	40	2	40	2	40	50	
T32	LBEN# VALID DELAY	0	30	0	25	0	22	50	
T33	CDEN# VALID DELAY	2	35	2	35	2	35	25	
T34	NPCS# DELAY	2	65	2	65	2	50	50	
T35	59CS# VALID DELAY	8	42	8	42	8	42	50	
T36	PBA# DELAY	2	34	2	34	2	30	50	

NOTE:
1. Write data is sampled on different clock edges by different DMA internal registers. T19 is speced relative to the earliest sampling edge, while T20 is relative to the latest edge.
2. PRDYI# must be inactive and stable according to these specs at all DMA state boundaries except at the end of the TC boundary at which the cycle is to be terminated.

DRIVE LEVELS AND MEASUREMENT POINTS FOR A.C. SPECIFICATIONS

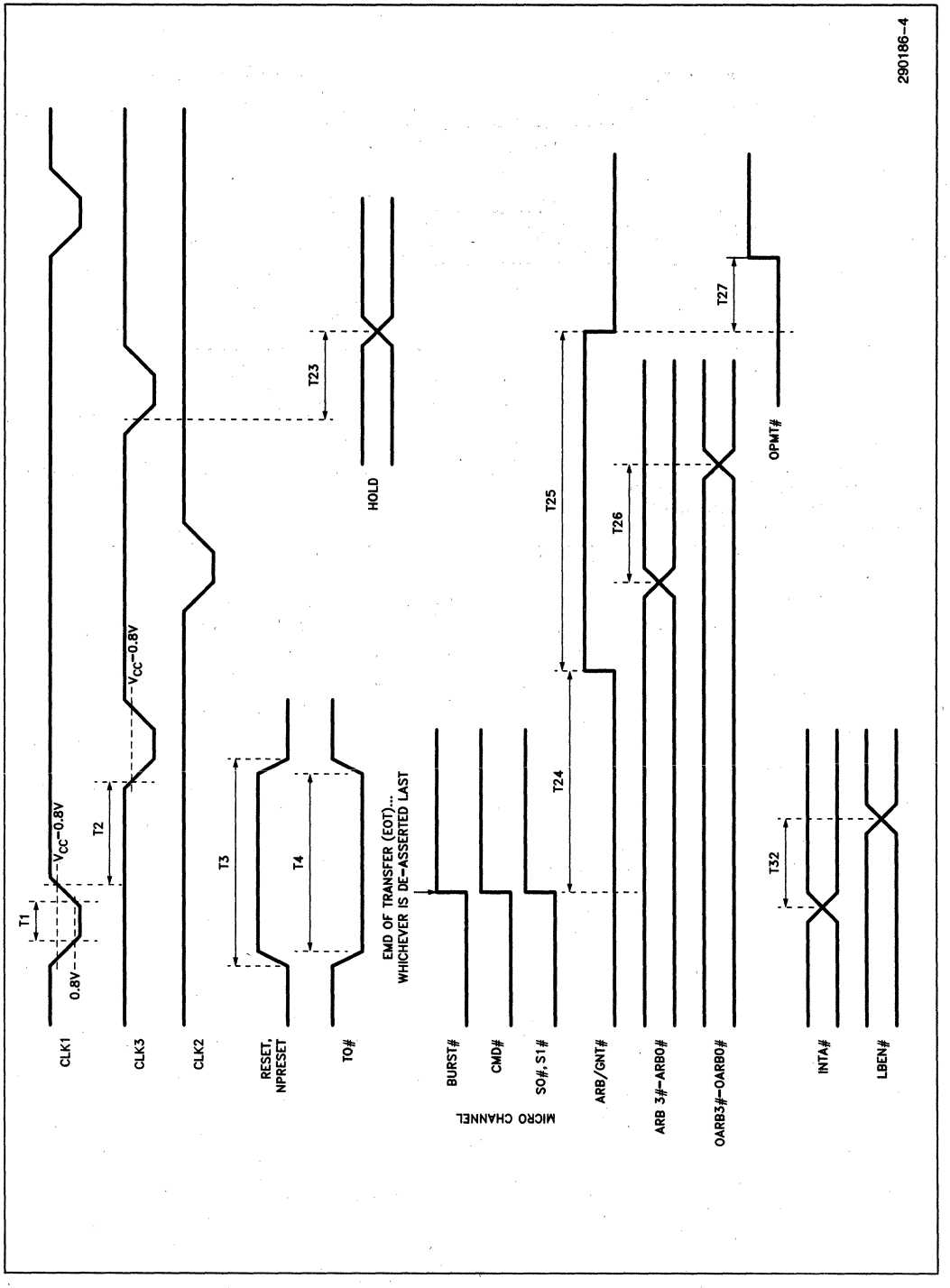


290186-3

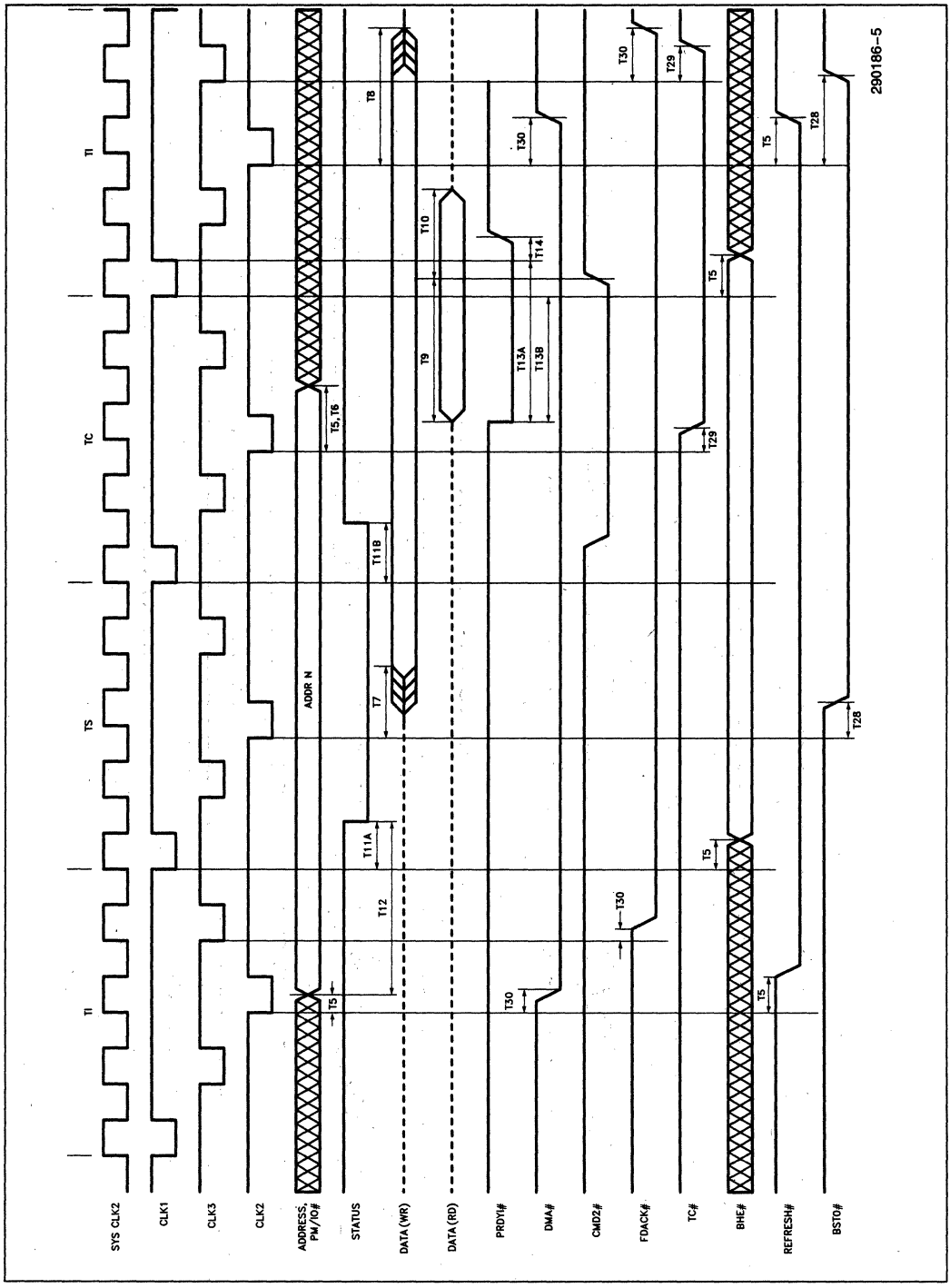
LEGEND:
A. Maximum Output Delay Specification.
B. Minimum Output Delay Specification.
C. Minimum Input Setup Specification.
D. Minimum Input Hold Specification.

Input waveforms have tr ≤ 2.0 ns from 0.8V to 2.0V.

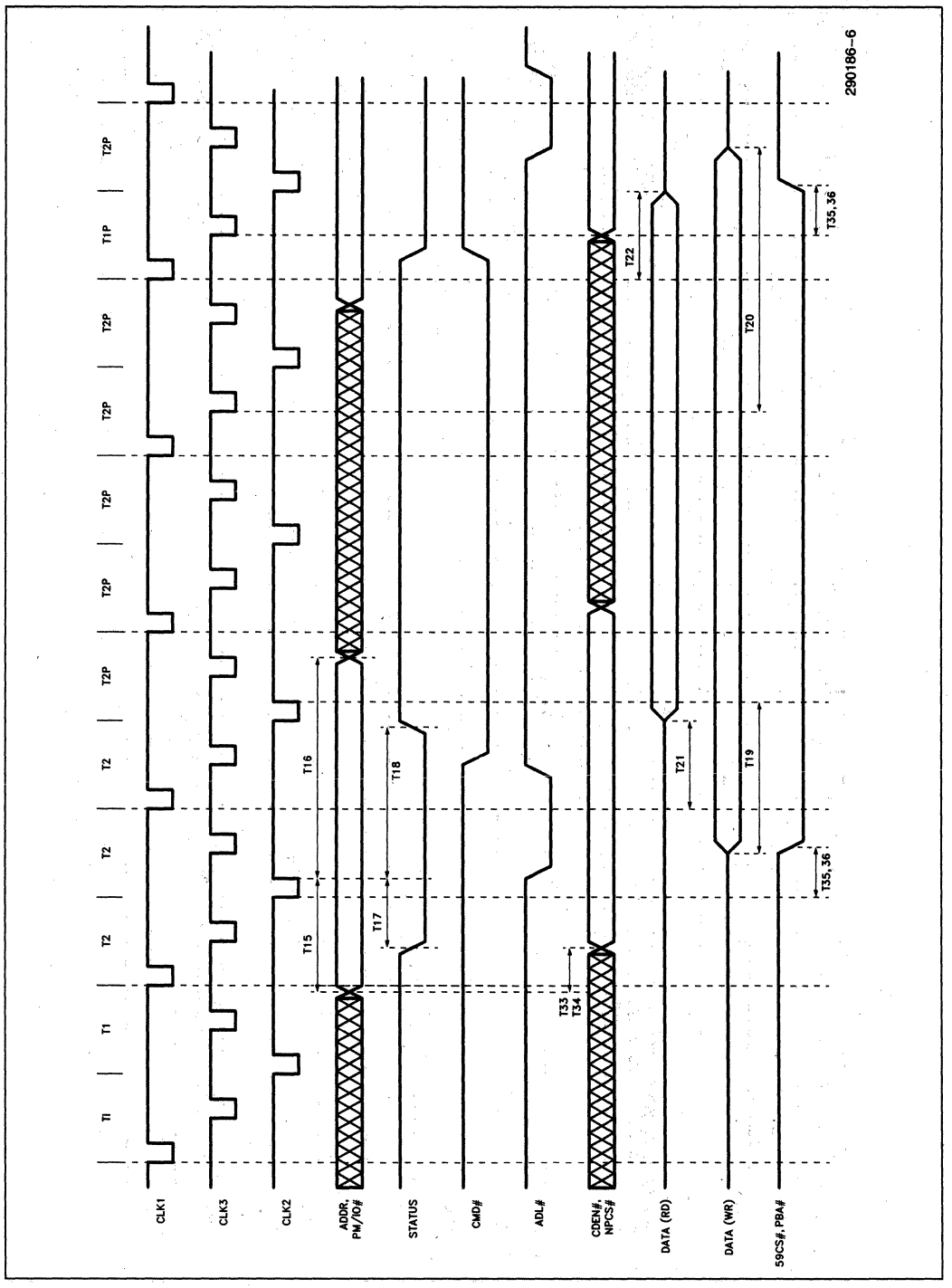
SYSTEM TIMINGS



DMA ... MASTER MODE



DMA . . . SLAVE MODE



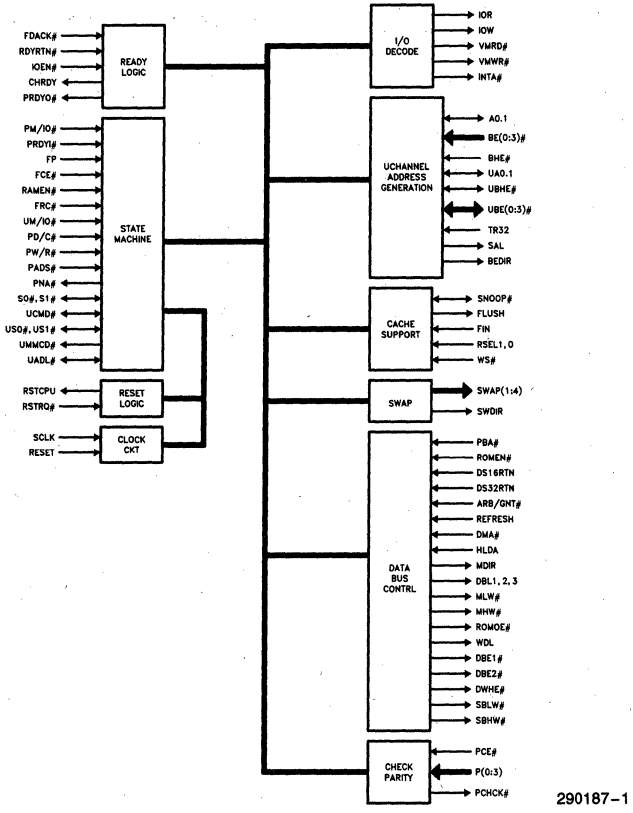
82308
Micro Channel BUS CONTROLLER

- **Micro Channel Compatible Bus Control**
- **Supports 8-, 16- or 32-Bit Data Transfers on the Micro Channel**
- **Optional Hardward Enforced I/O Recovery Mechanism**
- **Cache Controller (82385) Interface to Maximize Performance for 80386 Based Systems**
- **Low Power CHMOS Technology**
- **100-Pin Plastic Quad Flat Pack Packaging**
  (See Packaging Spec., Order # 231369)

The 82308 Micro Channel Bus Controller is the complementary device to the 82309 Address Bus Controller. It is designed to facilitate data transfers between the Microprocessor, DMA, Memory and Micro Channel bus. It generates the appropriate data conversion and alignment control signals to implement an external byte swap mechanism for transferring data of equal and different widths.

The 82308 Bus Controller generates all control signals necessary to run Micro Channel Memory and I/O Bus cycles for both 80386 and 80386SX processors.

To implement the highest performance 80386 based system with the 82385 cache controller, the Bus Controller features special cache hardware interface signals.



290187-1

October 1988
Order Number: 290187-001

STATE MACHINE

The primary purpose of the state machine is to generate the Micro Channel signals for processor and DMA cycles. These Micro Channel signals are: S0#, S1#, ADL#, MMCCMD# and CMD#. The state machine also generates the PNA# signal required by the 386 CPU and generates the DMA S0# and S1# based on the 386 processor status.

DATA TRANSFER

The 82308 Bus Controller directs the transfer of data between the 32-bit 80386 bus and 32-bit, 16-bit or 8-bit devices on the Micro Channel. For 16-bit transfers initiated by the 80386SX or DMA, the Bus Controller provides the control signals to facilitate transfers to 16-bit or 8-bit devices on the Micro Channel data bus and vice versa.

In addition to providing the transceiver direction, latch, and enable signals, the Bus Controller manipulates Address signals for the DMA and Micro Channel. For example, a 32-bit access to an 8-bit device is broken into four cycles, and the BC automatically sequences A1 and A0 in each cycle.

The 82308 Bus Controller also supports the ROM BIOS by providing the output enable for the BIOS based on the decoded BIOS address signal from the Address Bus Controller (ROMEN#).

RESET DETECT

The reset detect logic generates a synchronous CPU reset signal based on any of the following events:

— An active low-pulse on the RC# input to the Bus Controller

— Processor Shutdown Condition based on the Processors' status signals

— Power-up condition as determined by the RESET input.

I/O SUPPORT

The 82308 Bus Controller generates Memory and I/O Read and Write signals for devices on the motherboard. It also generates the Interrupt Acknowledge signal.

The Bus Controller extends motherboard device accesses by de-asserting CHRDY until a read or write strobe is generated. This gives the peripheral device an opportunity to extend the cycle even further if required by driving its own CHRDY inactive after it detects the read or write strobe. The motherboard device decode is performed by the 82309 Address Bus Controller.

CACHE SUPPORT

The 82308 Bus Controller supports 82385 Cache Controller interface signals to allow maximum system performance in cache based 386 systems.

CACHE FLUSH

The Bus Controller generates a synchronous flush output signal (FLUSH) to the cache controller whenever the flush request (FIN) is generated.

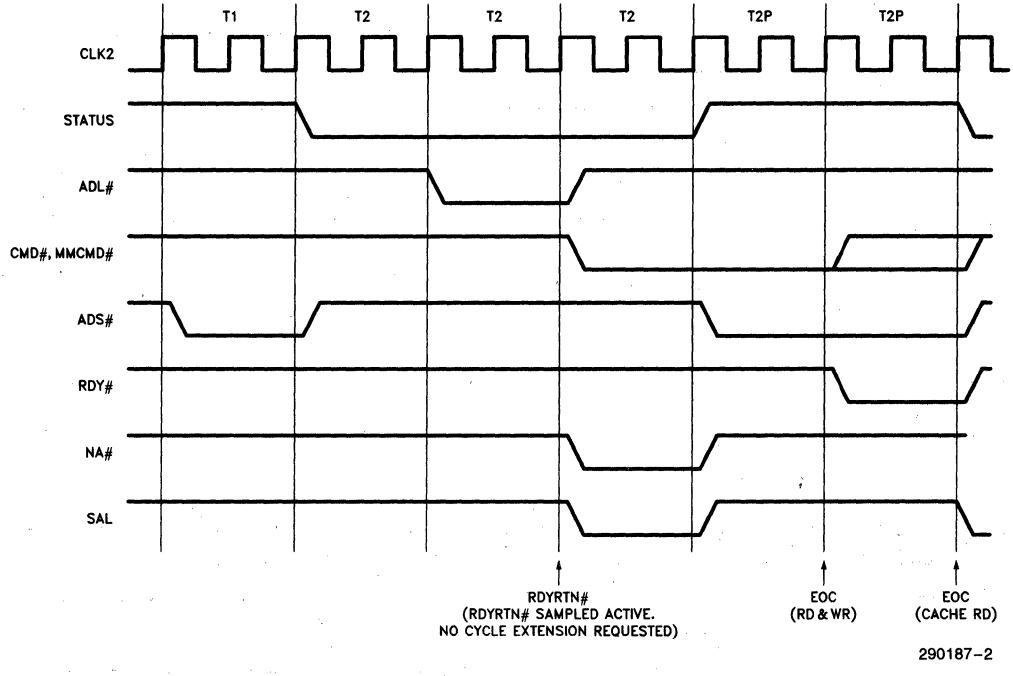
SNOOP STROBE

The Snoop Strobe output of the 82308 Bus Controller is a synchronized strobe indicating a valid address during non-processor write cycles. It is compatible with the 82385 cache controller's bus watching mechanism.

HARDWARE ENFORCED I/O RECOVERY

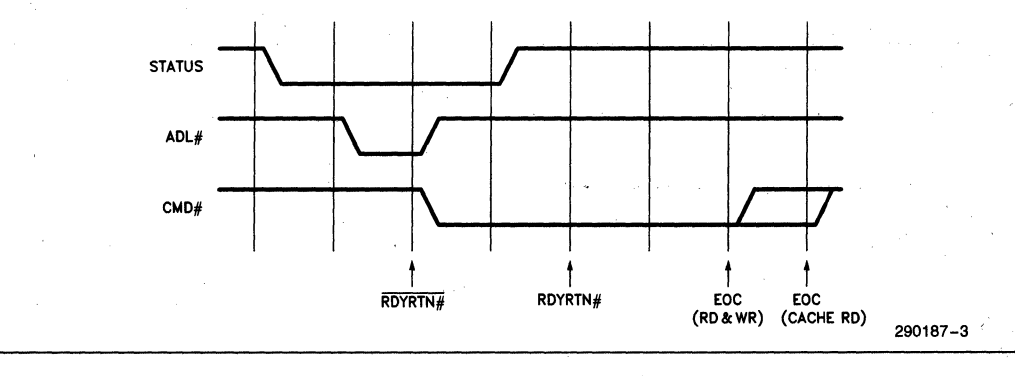
Certain I/O devices require a minimum delay between consecutive accesses. Typically, software loops are executed in the I/O routine to force the delay, but software loops cannot guarantee minimum delay times in all cases. The 82308 Bus Controller provides the option of enforcing I/O recovery in hardware. This mechanism is controlled by two inputs (RSEL1 and RSEL0), which select one of four possible minimum I/O recovery times. At the end of a CPU initiated I/O cycle, an internal timer is triggered, and the 82308 will not allow the next I/O access to proceed until the timer has timed out. The specific functioning of RSEL1 and RSEL0 is detailed in the pin definitions and A.C. Timing specifications.

16 MHz CPU DEFAULT CYCLE (FP = 0)

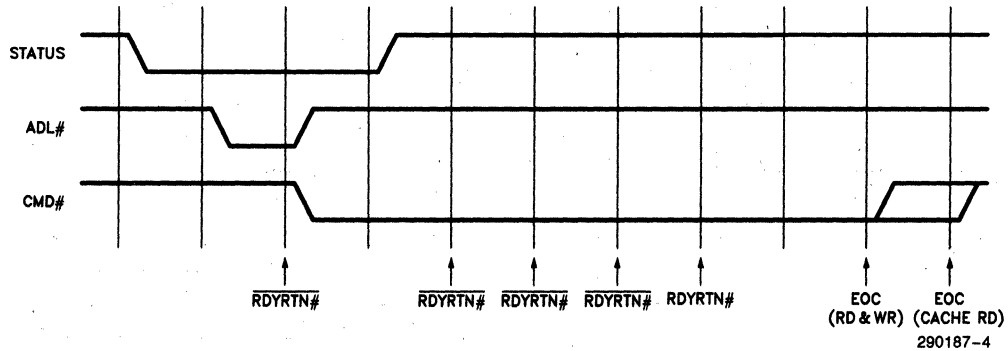


NOTE:
MMCCMD# only driven in CPU initiated cycles.
EOC = End Of Cycle.

16 MHz CPU SYNCHRONOUS EXTENDED

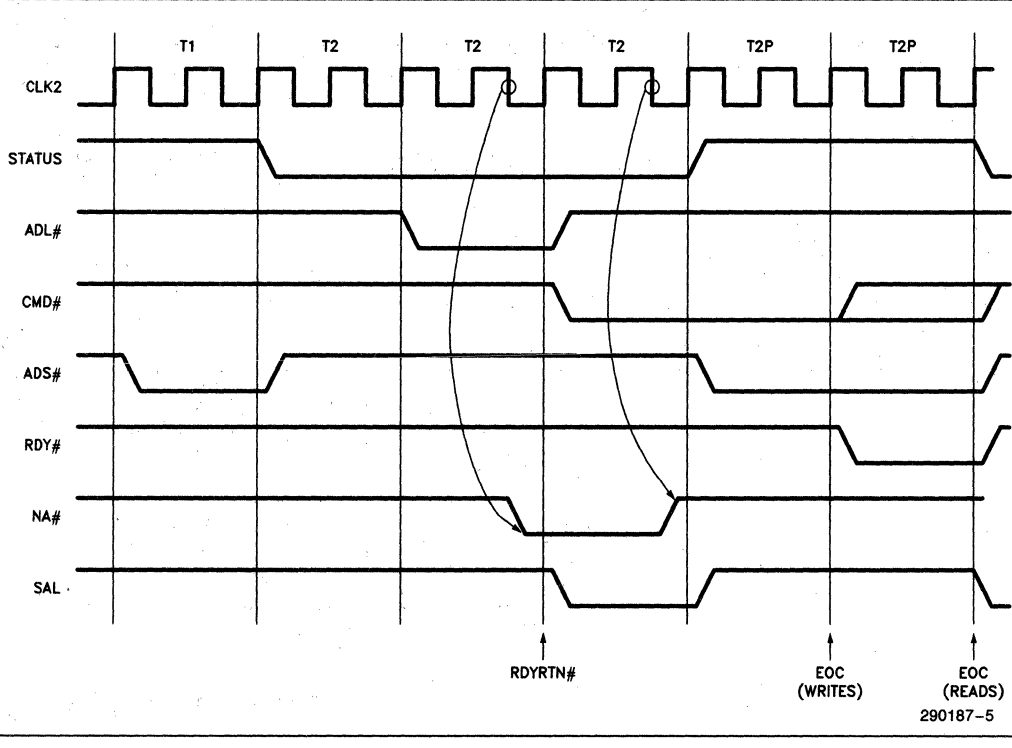


16 MHz CPU ASYNCHRONOUS EXTENDED

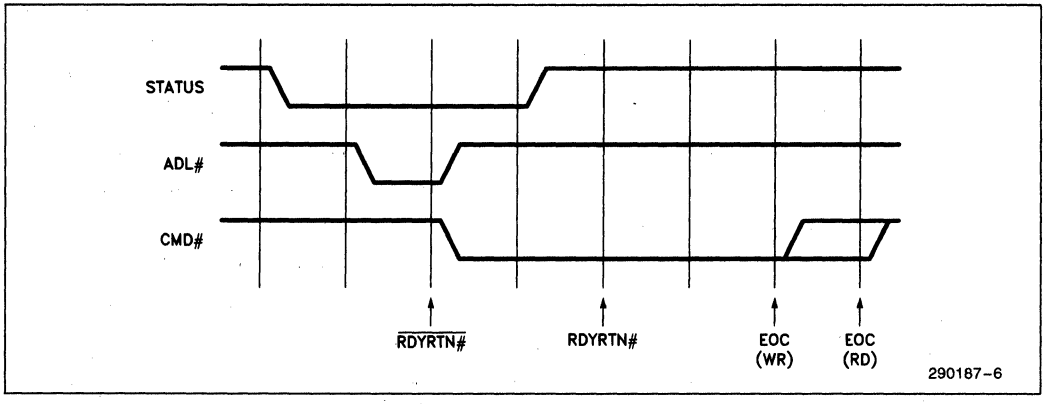


EOC = End Of Cycle

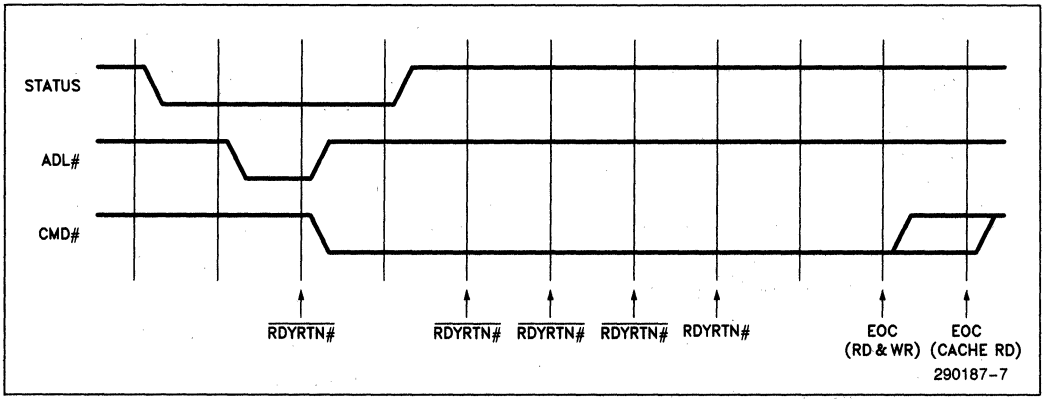
20 MHz CPU DEFAULT CYCLE (FP = 1)



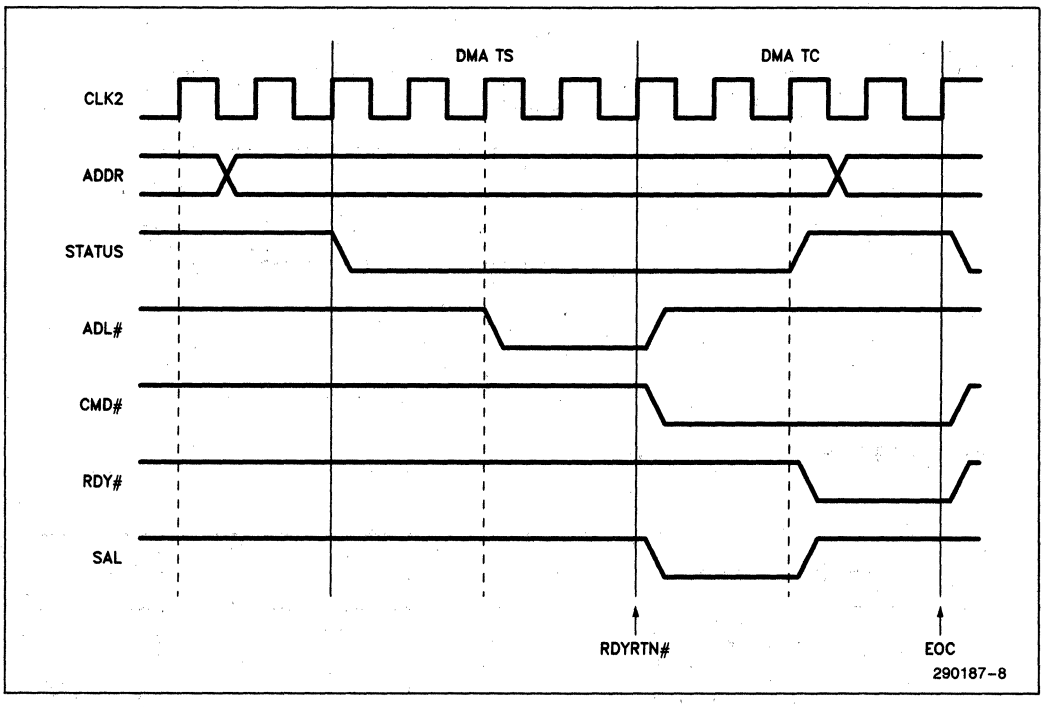
20 MHz CPU SYNCHRONOUS EXTENDED



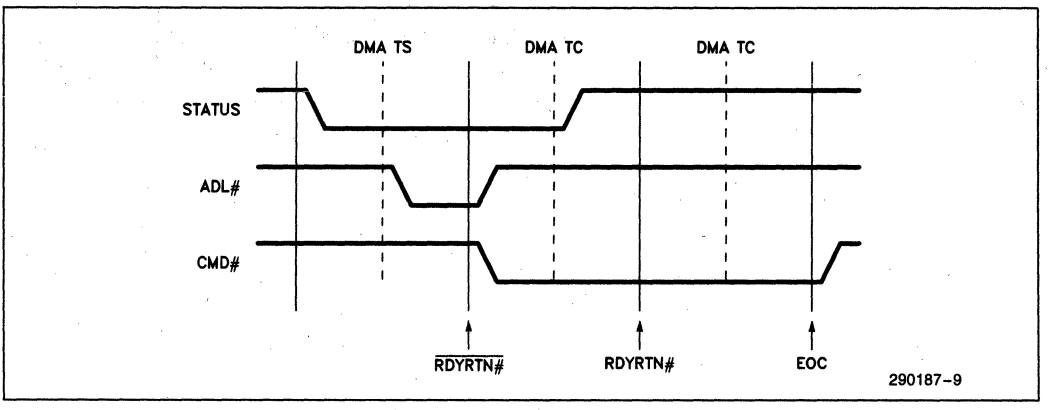
20 MHz CPU ASYNCHRONOUS EXTENDED



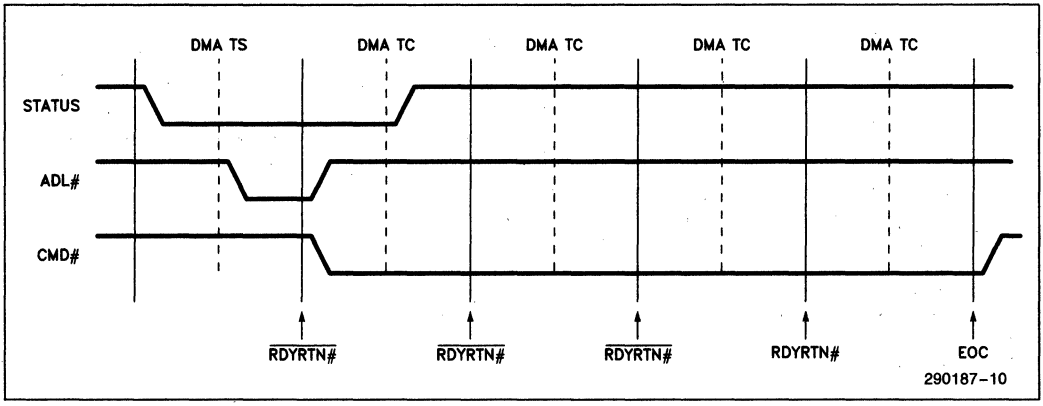
DMA DEFAULT CYCLE



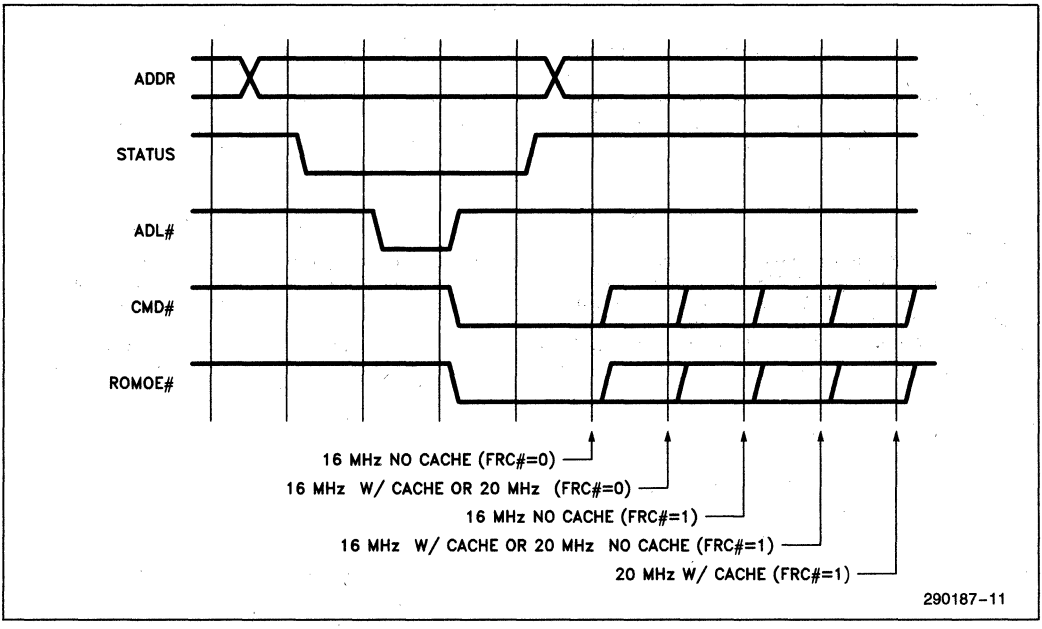
DMA SYNCHRONOUS EXTENDED



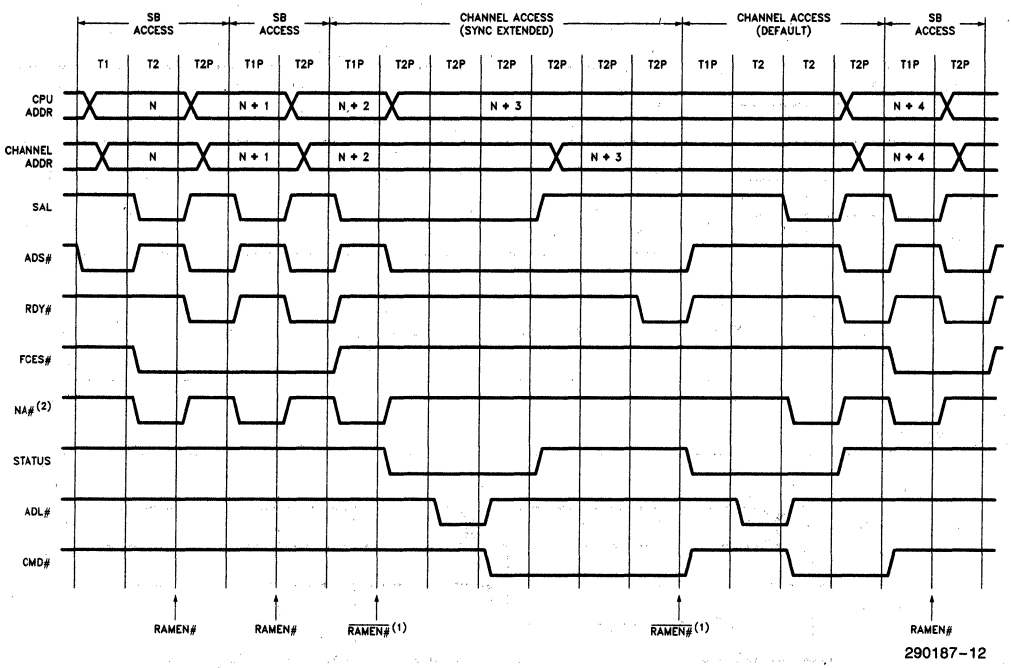
DMA ASYNCHRONOUS EXTENDED



ROM CYCLES



CPU SYSTEM BOARD MEMORY/Micro Channel MEMORY ACCESSES



290187-12

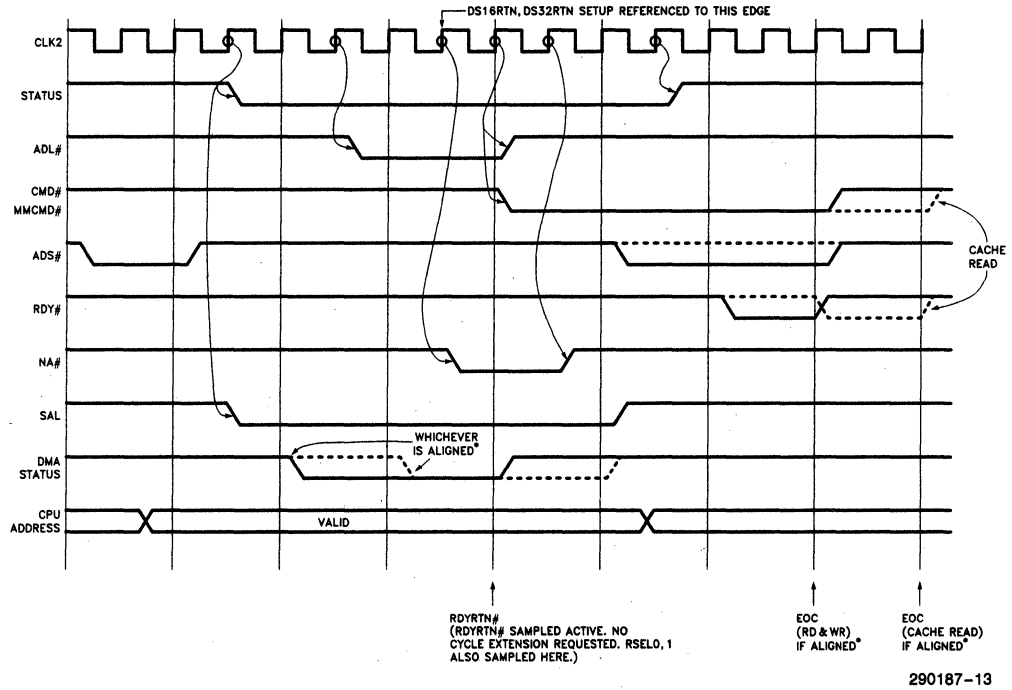
NOTES:

1. RAMEN# distinguishes between system board and channel memory accesses. The BC must wait for RAMEN# to resolve before driving STATUS in a channel access. Thus, in non-pipelined channel accesses or pipelined channel accesses in which the ABC sees only one state of pipelined address, the BC delays starting the channel access until the end of the T1P or first T2 state.
2. In memory cycles the BC must drive NA# before RAMEN# resolves in order to sustain 0WS pipelined page hits.

82308HS-25 MICROCHANNEL BUS CONTROLLER TIMING DIAGRAMS

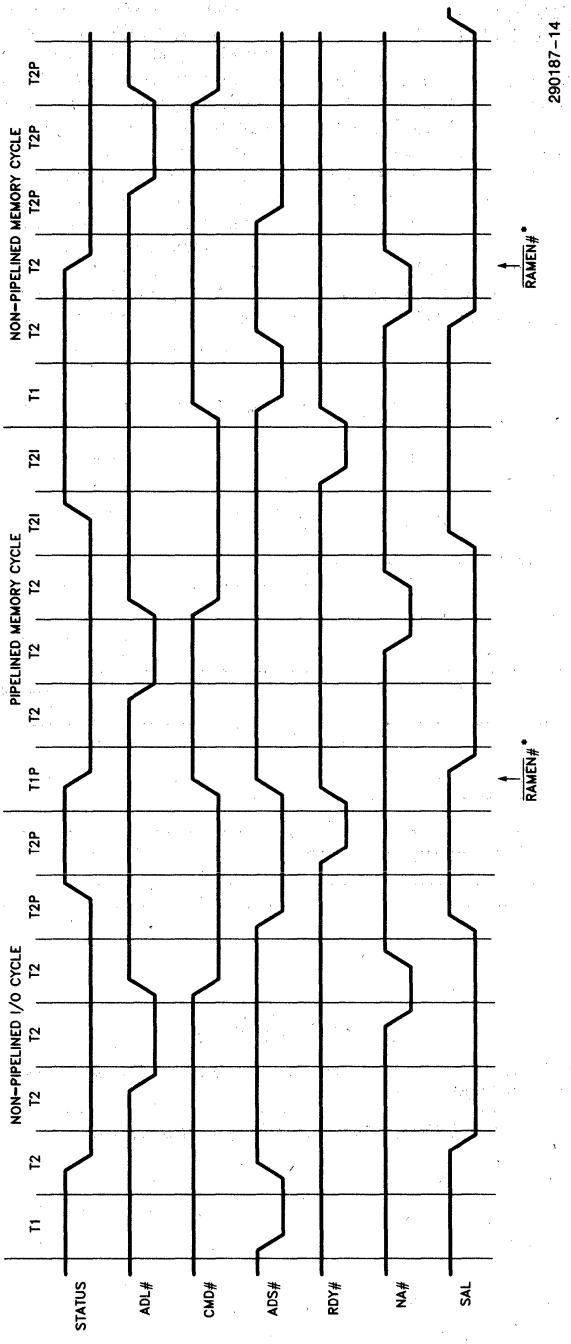
The 82308HS-25 provides Microchannel Bus Control for 25 MHz 80386 systems. It is 100% function and pin compatible with the 82308-16/20 Bus Controller, so minimal system re-design is required to upgrade current 16 MHz or 20 MHz systems to 25 MHz. (Note that the 82308HS-25 FP input must be tied high.)

Although the 82308HS-25 is functionally identical to the 82308-16/20, its internal state machine and external timing behavior are modified to insure full compatibility with published Microchannel timings at the increased CPU frequency, and to accommodate 25 MHz system and component specifications. This addendum to the 82308-16/20 data sheet provides the basic timing diagrams for the 82308HS-25, highlighting the specific clock edges that either sample specific inputs, or else trigger specific outputs. All AC specification output delays are referenced to the "causal" clock edge, and input setup/hold times are referenced to the sampling clock edge. Any signal not specifically addressed in these diagrams behaves just as it does in the 82308-16/20. (Note in the AC specifications that notes numbered 21 or greater apply only to the 82308HS-25.)



*If necessary, a wait state is added to CPU I/O cycles to align them with even DMA state boundaries. (Recall that DMA states are 4 clock phases wide while 80386 states are only 2 phases.)

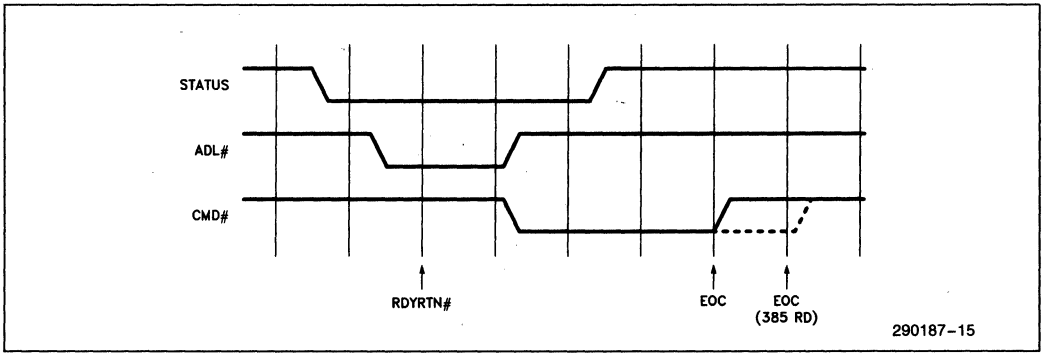
82308HS 25 MHz CPU Default I/O Cycle



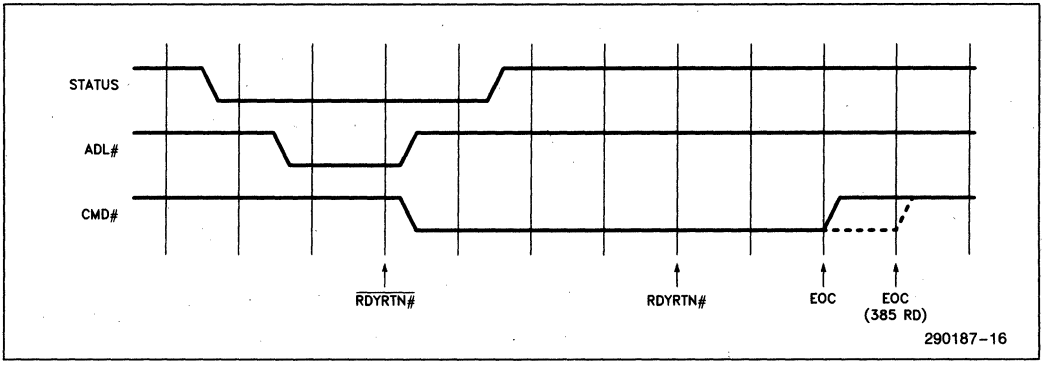
*Ramen# distinguishes between (non-broadcast) system board and channel memory accesses. (The 82308-25 samples Ramen# and Romen# on the phase 2 clock edge.) The BC must wait for Ramen# to resolve "not true" before driving status in a channel access. Thus, in a non-piplined channel memory access, the BC delays starting the channel cycle until after the first T2 state.

82308HS 25 MHz CPU Default Micro Channel Cycles

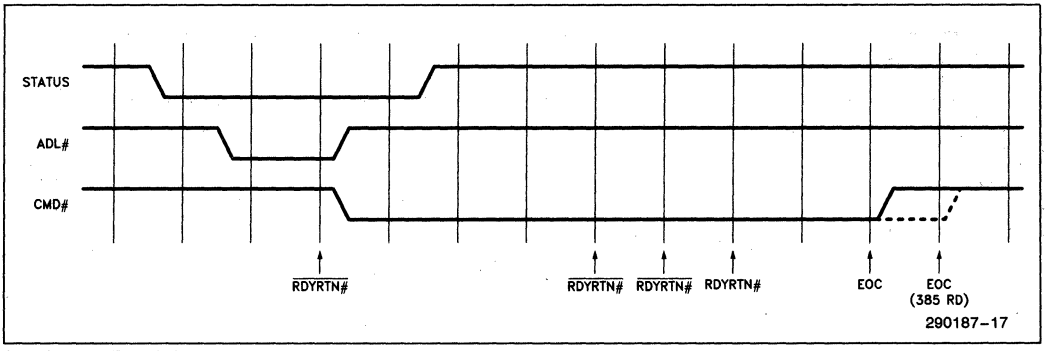
82308HS 25 MHz CPU EXTENDED CYCLES



Default

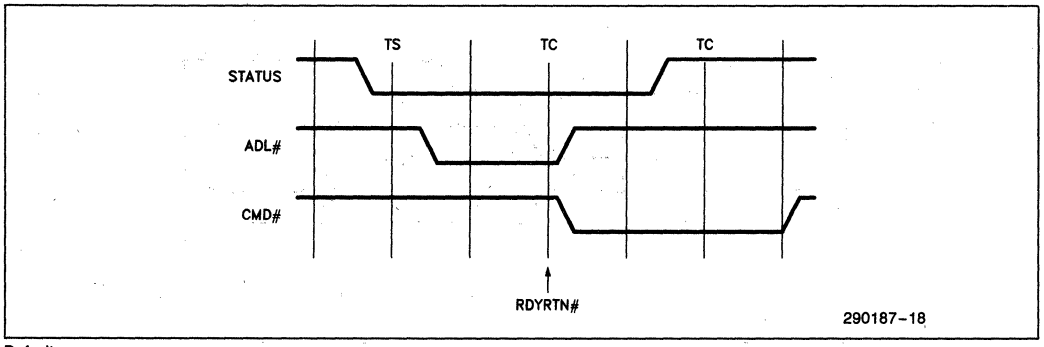


Synchronous Extended

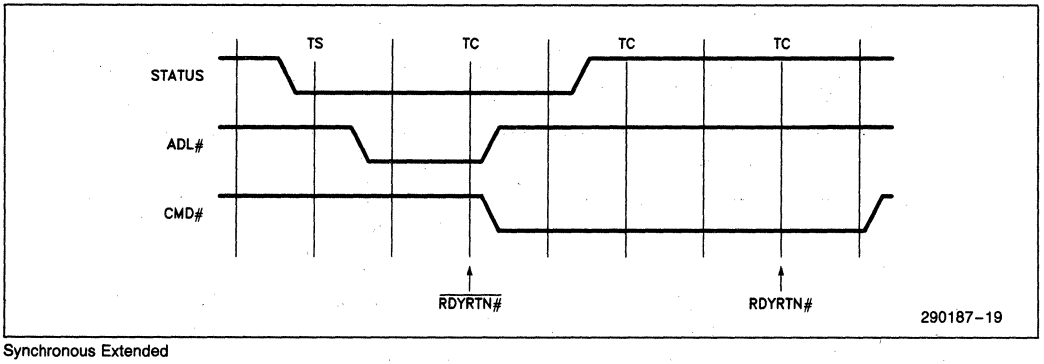


Asynchronous Extended

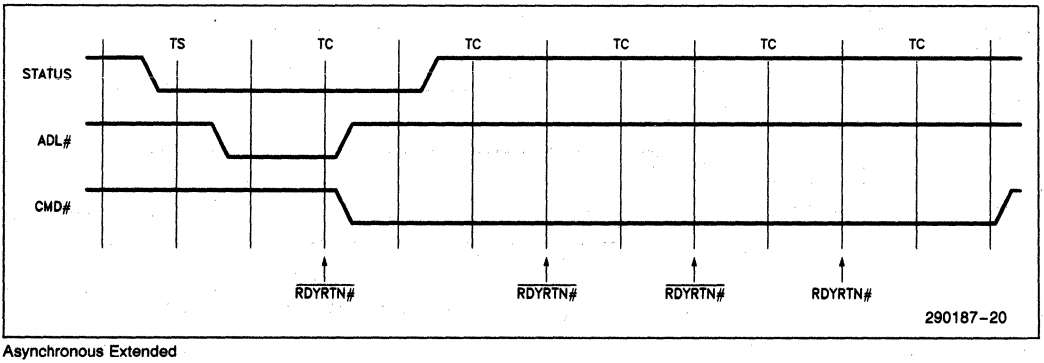
82308HS 25 MHz DMA EXTENDED CYCLES



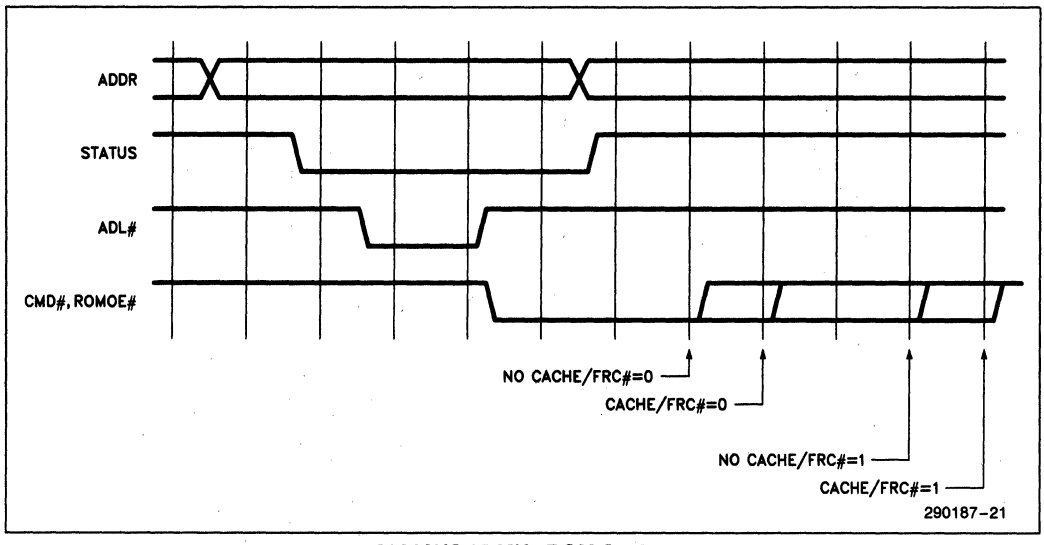
Default



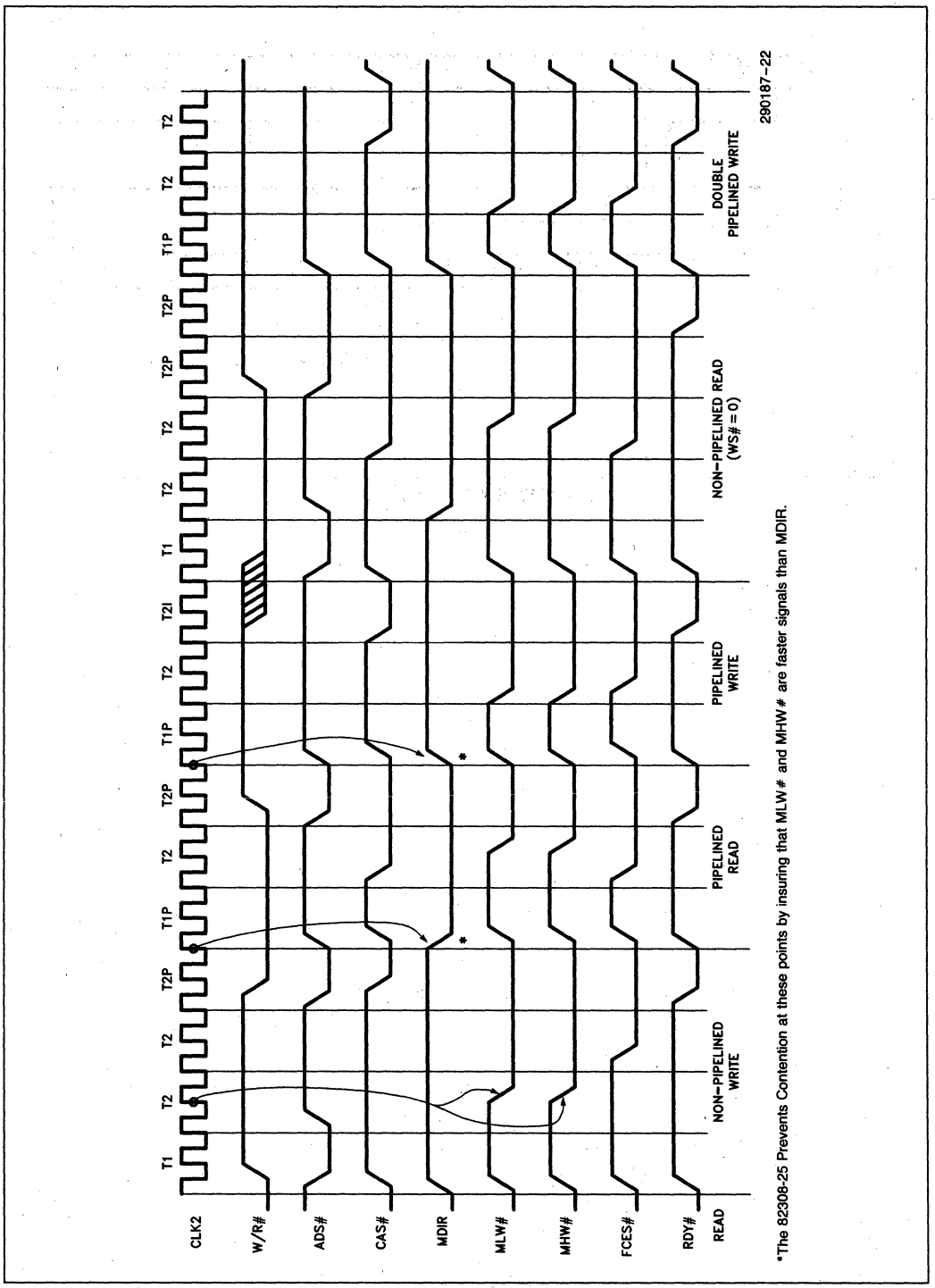
Synchronous Extended



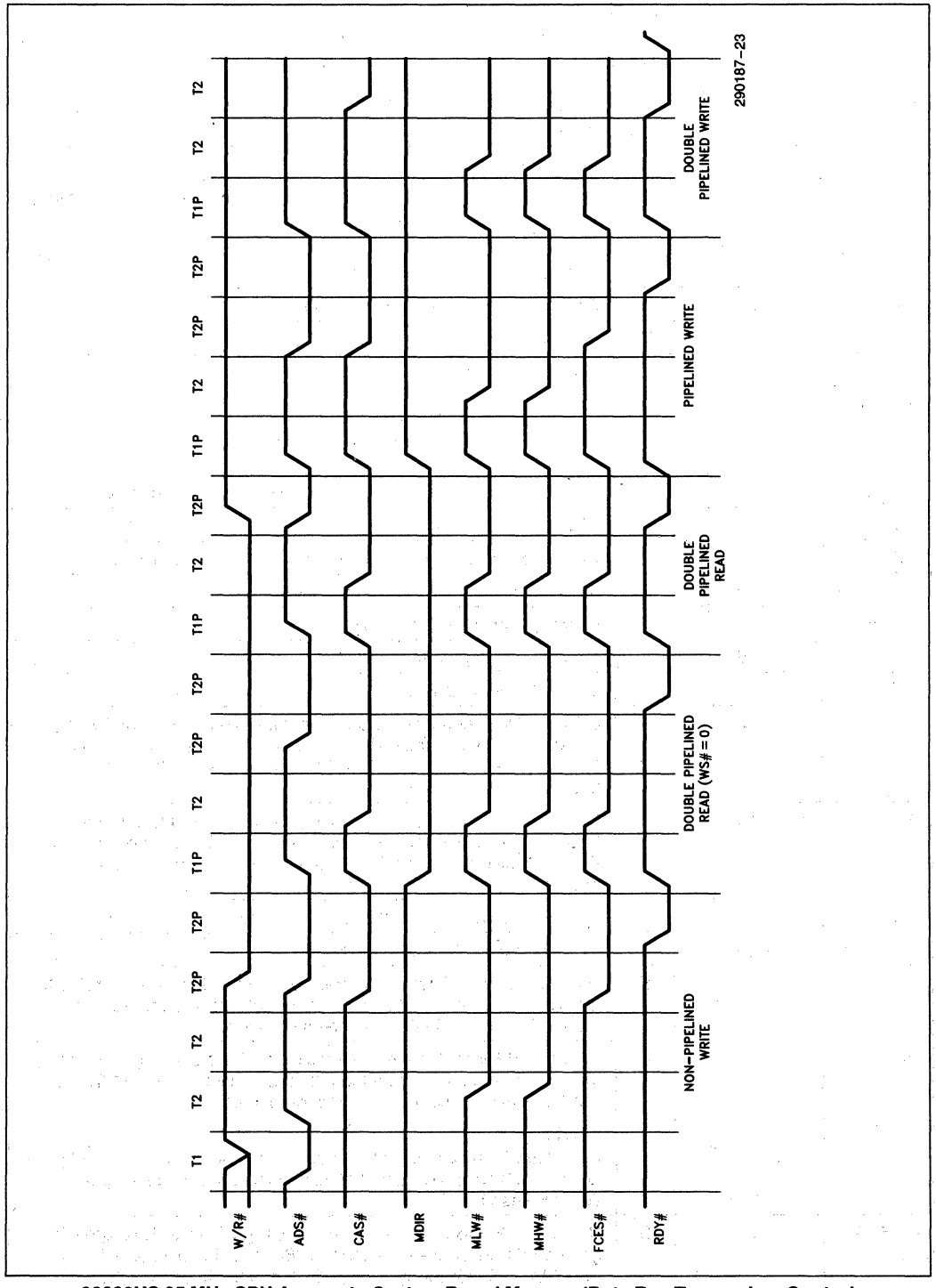
Asynchronous Extended



82308HS 25 MHz ROM Cycles



82308HS 25 MHz CPU Access to System Board Memory/Data Bus Transceiver Control



82308HS 25 MHz CPU Access to System Board Memory/Data Bus Transceiver Control

82308 Micro Channel Bus Controller Pin Definitions

Signal Name	Pin Number	I/O	Description
RSTRQ#	87	I	Logical NOR of 8042 pin 20 and LCS ALTRESET to initialize reset (Software reset).
RSTCPU	91	O	Microprocessor Reset.
PCE#	82	I	Enable Parity Check from Memory Encoding Register Bit 0. This input should be tied low for a model 60 system. (Parity check always enabled.)
PO	83	I	Parity Error from DRAM for bits 0–7.
P1	84	I	Parity Error from DRAM for bits 8–15.
P2	85	I	Parity Error from DRAM for bits 16–23.
P3	86	I	Parity Error from DRAM for bits 24–31.
PCHCK#	90	O	Parity Error Output.
IOEN#	79	I	Active low signal from the Address Bus controller indicating a motherboard I/O device address.
CHRDY	92	O	Input to the Channel Ready Return logic to extend the current cycle.
FDACK#	81	I	FDC DACK# signal.
IOR	97	O	I/O Read signal for motherboard devices (8042, 8259, etc.).
IOW	98	O	I/O Write signal for motherboard devices.
VMWR#	96	O	Memory write strobe to the VGA.
VMRD#	95	O	Memory read strobe to the VGA.
INTA#	94	O	INTA# Input to the 8259.
A0	9	B	CPU and DMA Address 0. A0 will be driven by the Bus controller based on the byte enable signals when the 386 owns the bus. It is a Bus controller input in an 80386SX system, or when the DMA is master.
A1	10	B	CPU and DMA Address 1. A1 will be driven by the Bus controller based on the byte enable signals when the 386 owns the bus. It is a Bus controller input in an 80386SX system, or when the DMA is master.
BHE#	100	I	Byte High Enable signal from the CPU and DMA.
BE0–3#	11–12, 14–15	I	Byte Enable bits 0–3 from the 80386.
UA0	3	B	Unbuffered Micro Channel Address bit 0. This signal is generated by the bus controller based on the byte enable signals and A0 from the DMA. It is also manipulated by the swap logic.
UA1	4	B	Unbuffered Micro Channel Address bit 1. This signal is generated by the bus controller based on the byte enable signals and A1 from the DMA. It is also manipulated by the swap logic. **NOTE:** For 80386SX systems, UA1 is unconnected, and should be lightly pulled up (10K). The channel A1 is latched along with the upper address lines.)
UBHE#	16	B	Unbuffered Micro Channel System Bus High Enable. This signal is generated from BE0–3# when the 386 owns the bus, or BHE# from the DMA. In 80386SX systems, UBHE# is a reflection of the 80386SX BHE# output.

82308 Micro Channel Bus Controller Pin Definitions (Continued)

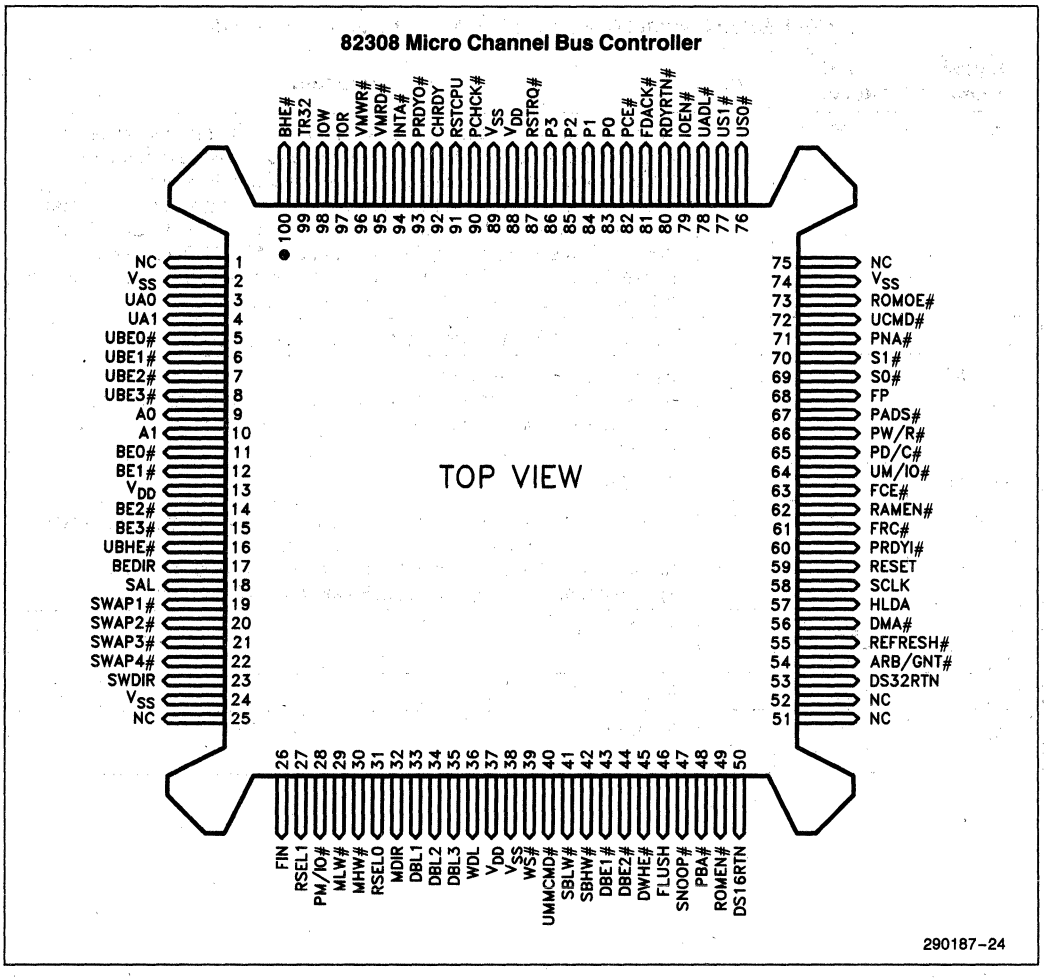
Signal Name	Pin Number	I/O	Description
UBE0–3#	5–8	B	Unbuffered Micro Channel Byte Enable bits 0–3. These byte enable signals are driven by the bus controller when the CPU or DMA is master. (An external PAL is required to generate the channel byte enables on behalf of a 16-bit channel master that requests translation.)
TR32	99	I	Translate 32 from the Micro Channel to indicate 32 bit masters driving BE0–3# (when inactive). (Tie high for 80386SX system.)
SAL	18	O	Latch enable for the system address bus. This signal controls the address latch between the CPU bus and channel.
BEDIR	17	O	Direction control for the UBE0–3# transceiver. It is high when the Bus controller is driving UBE0–3#.
PBA#	48	I	Indicates that the DMA or numeric coprocessor has been selected and is using the local data bus.
ROMEN#	49	I	Decode that inidicates that the BIOS ROM has been selected.
DS16RTN	50	I	Micro Channel Data Size 16 signal.
DS32RTN	53	I	Micro Channel Data Size 32 signal.
ARB/GNT#	54	I	Micro Channel ARB/-GNT status.
REFRESH#	55	I	Refresh Indicator.
DMA#	56	I	Indicates that the DMA owns the bus.
HLDA	57	I	CPU HLDA input. Indicates CPU controls local address and data bus if low.
WDL	36	O	Latch enable signal for latching data bus D0–31 for generating Micro Channel D0–31 signal. Insures Micro Channel write data hold time spec is met.
DBE1#	43	O	Output Enable for driving data on CPU D0–7 onto Micro Channel D0–7 during CPU or DMA writes or Micro Channel DRAM reads.
DBE2#	44	O	Output Enable for driving data on CPU D8–15 onto Micro Channel D8–15 during CPU or DMA writes or Micro Channel DRAM reads.
DWHE#	45	O	Output Enable for driving data on CPU D16–31 onto Micro Channel D16–31.
SBLW#	41	O	Output enable for driving data on Micro Channel D0–15 onto CPU D0–15.
SBHW#	42	O	Output enable for driving data on Micro Channel D16–31 onto CPU D16–31.
MDIR	32	O	Direction control for transferring data between the CPU Data Bus and the DRAM memory data bus.
MLW#	29	O	Output enable for the transceiver between the CPU data bus D0–15 and the DRAM memory data bus.
MHW#	30	O	Output enable for the transceiver between the CPU data bus D16–31 and the DRAM memory data bus.
ROMOE#	73	O	Output Enable signal for the BIOS ROMs.
SWAP1#	19	O	Transceiver enable for transferring data between Micro Channel Data Bus 0–7 and 8–15.

82308 Micro Channel Bus Controller Pin Definitions (Continued)

Signal Name	Pin Number	I/O	Description
SWAP2#	20	O	Transceiver enable for transferring data between Micro Channel Data Bus 0–7 and 16–23.
SWAP3#	21	O	Transceiver enable for transferring data between Micro Channel Data Bus 8–15 and 24–31.
SWAP4#	22	O	Transceiver enable for transferring data between Micro Channel Data Bus 0–7 and 24–31.
SWDIR	23	O	Direction control for Micro Channel Data Bus transceivers.
DBL1	33	O	Latch enable for latching Micro Channel Data Bus 0–7.
DBL2	34	O	Latch enable for latching Micro Channel Data Bus 8–15.
DBL3	35	O	Latch enable for latching Micro Channel Data Bus 16–23.
SCLK	58	I	Microprocessor Clock.
RESET	59	I	Synchronized reset input to synchronize the internal clock with the processor phase.
RDYRTN#	80	I	Channel Ready Return signal from Micro Channel (active low).
PRDYO#	93	O	Microprocessor ready signal.
FP	68	I	Processor Speed Select. (20 MHz = 1, 16 MHz = 0.)
PRDYI#	60	I	Synchronized microprocessor ready input.
FRC#	61	I	Fast ROM Cycle Select. When tied low, ROM cycles are run as Micro Channel default read cycles. When tied high, additional wait states are inserted to accommodate slower ROMs.
RAMEN#	62	I	Decode that indicates a system board DRAM access.
FCE#	63	I	Input that directs BC to terminate a CPU system board DRAM access.
UM/IO#	64	I	Micro Channel Memory/IO status.
PD/C#	65	I	CPU D/C# output.
PW/R#	66	I	CPU W/R# output.
PADS#	67	I	CPU ADS# output. (Indicates address valid.)
PNA#	71	O	Next Address Signal for address pipelining.
S0#, S1#	69, 70	B	DMA Status lines; input by the BC when DMA is master, and output by the BC when CPU is master.
UADL#	78	B	Micro Channel Address Latch Signal.
UMMCMD#	40	O	Micro Channel Matched Memory Command Signal.
UCMD#	72	B	Micro Channel Command Signal.
US0#, US1#	76, 77	B	Micro Channel Status.
RSEL1 RSEL0	27, 31	I I	These two signals are used for hardware enforced I/O recovery. They are sampled at the leading edge of UCMD# during CPU initiated I/O cycles, and are used to select one of four possible I/O recovery times. At the end of the I/O cycle, an internal timer is triggered, and then times out after the selected I/O recovery time. The next I/O cycle is not allowed to proceed into the active UCMD# phase until the internal timer times out. RSEL1,0 can be strapped for a particular time, or else driven from a combinatorial address decode.

82308 Micro Channel Bus Controller Pin Definitions (Continued)

Signal Name	Pin Number	I/O	Description
FIN	26	I	Asynchronous cache flush request input. A pulse on FIN causes a cache flush. Also, if FIN is left active for a long period of time, the 82385 will be kept in flush mode for as long as FIN is active. The exception to this is when the BC is directed to do a software initiated CPU reset when FIN is active. The BC will de-activate FLUSH for a period of time surrounding the falling edge of RSTCPU so as to prevent the 82385 from entering its self-test mode. If FIN is still active after the reset, then FLUSH will be re-activated.
FLUSH	46	O	Synchronous flush request to the 82385 Cache Controller.
SNOOP#	47	B	Synchronous strobe to the cache controller to indicate valid address during a non-processor memory write. SNOOP# is sampled at reset to indicate the presence of a cache. (1 = cache present, 0 = no cache.)
PM/IO#	28	I	CPU M/IO# output.
WS#	39	I	This input, if tied low, inserts an additional wait state into CPU reads from system board memory beyond the number of wait states requested via the FCE# input. It is primarily intended for cache applications, which typically require increased CPU data setup.
NC	1, 25, 51, 52, 75		No Connect
V_{DD}	13, 37, 88		Power
V_{SS}	2, 24, 38, 74, 89		Ground



NOTE:
NC = No Connect

82308 PARAMETRICS

ABSOLUTE MAXIMUM RATINGS*

Case Temperature under Bias .... −40°C to +85°C
Storage Temperature ..... −65°C to +150°C
Voltage to Any Pin with Respect to Ground ..... −0.3V to (V_{CC} + 0.3)V
DC Supply Voltage (V_{CC}) ..... −0.3V to +7.0V
DC Input Current ..... ±10 mA

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS

T_C = 0°C to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Min	Max	Units	Conditions
V_{IL}	Input Low Voltage		0.8	V	
V_{IH}	Input High Voltage	2.0		V	
V_{IL}	Input Low Voltage		0.8	V	SCLK
V_{IH}	Input High Voltage	V_{CC} − 0.8		V	SCLK
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 4 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 4 mA
I_{CC}	Power Supply Current		180	mA	No DC Loads
I_{LI}	Input Leakage Current		±10	μA	$V_{SS} < V_{IN} < V_{CC}$
I_{OZ}	Tri-State Output Leakage Current		±10	μA	$V_{SS} < V_{OUT} < V_{CC}$

82308 Micro Channel BUS CONTROLLER A.C. SPECS

T_C = 0°C to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	82308 Kit 16 MHz		82308 Kit 20 MHz		82308HS Kit 25 MHz		C_L (pF)	Notes	Conditions
		Min	Max	Min	Max	Min	Max			
T1	SCLK PERIOD	31.25		25		20				
T2A	SCLK HIGH/LOW TIME (50%)	12		10		8				
T2B	SCLK HIGH/LOW TIME (90%)	8		6.5		6				
T3	RESET SETUP	10		10		10				
T4	RESET HOLD	4		4		4				
T5A	RSTCPU DELAY	2	30	2	30	2	27	25		
T5B	RSTCPU PULSE WIDTH	1980		1580		1260		25	14, 20	
T6A	FLUSH DELAY	5	41	5	34	5	28	50		
T6B	FLUSH PULSE WIDTH	240		190		150		50	15, 20	
T6C	SNOOP# DELAY	5	41	5	34	5	34	50		
T7	FIN PULSE WIDTH	25		25		25				
T8A	COMMAND I/O RECOVERY PULSE WIDTH	600		600		600			20	RSEL1,0 = 01
T8B	COMMAND I/O RECOVERY PULSE WIDTH	2500		2500		2500			20	RSEL1,0 = 10
T8C	COMMAND I/O RECOVERY PULSE WIDTH	10000		10000		10000			20	RSEL1,0 = 00
T8D	COMMAND I/O RECOVERY PULSE WIDTH	0		0		0			20	RSEL1,0 = 11
T9	RSEL0,1 SETUP	15		15		15			4	
T10	RSEL0,1 HOLD	20		20		20			4	
T11A	PW/R#, PD/C#, PM/IO# SETUP	25		22		13				
T11B	PADS# SETUP	25		22		17				
T12	PADS#, PW/R#, PD/C#, PM/IO# HOLD	4		4		4				
T13A	UM/IO# SETUP TO UADL# ↓	20		20		20				
T13B	UM/IO# SETUP TO SCLK	20		20		20			8	
T14A	UM/IO# HOLD FROM UADL# ↑ OR UCMD# ↓	20		16		16			7	
T14B	UM/IO# HOLD FROM SCLK	20		16		16			8	
T15	PRDYO# DELAY	4	30	4	24	3	20	75		
T16	PRDYI# SETUP	18		18		15				
T17	PRDYI# HOLD	3		3		3				
T18	PNA# DELAY	0	25	0	25	3	28	25		
T19A	UADL# DELAY (TPHL)	2	24	2	24	2	22	25		
T19B	UADL# DELAY (TPLH)	2	24	2	24	2	22	25		
T20A	UCMD# DELAY (TPHL)	2	23	2	20	2	20	25	2	
T20B	UCMD# DELAY (TPLH)	2	25	2	23	2	20	25		
T21A	UMMCMD# DELAY (TPHL)	2	23	2	20	2	20	25	2, 5	
T21B	UMMCMD# DELAY (TPLH)	2	25	2	23	2	20	25		
T22A	US0#, US1# (TPHL) DELAY	2	27	2	26	2	24	25	2	
T22B	US0#, US1# (TPLH) DELAY	2	27	2	26	2	24	25		
T22C	US0#, US1# (TPHL) DELAY FROM S0#, S1#	0	27	0	30	0	30	25		
T23	S0#, S1# SETUP	20		20		15				
T24	S0#, S1# DELAY	2	35	2	30	2	22	50		
T25	A0, A1 DELAY FROM BE0-3#	2	35	2	30	2	25	25		
T26	UBE0-3#, UA0-1#, UBHE# DELAY FROM BE0-3#, A0-1, BHE#	2	30	2	28	2	28	25		
T28	SAL DELAY	2	36	2	30	2	27	100		
T30A	SBHW#, SBLW# INACTIVE DELAY FROM SCLK	2	30	2	25	2	32	50	11, 21	
T30B	SBHW#, SBLW# DELAY FROM UADL#	2	40	2	40	2	40	50	25	

82308 Micro Channel BUS CONTROLLER A.C. SPECS

T_C = 0°C to +70°C, V_{CC} = 5V ±10% (Continued)

Symbol	Parameter	82308 Kit 16 MHz		82308 Kit 20 MHz		82308HS Kit 25 MHz		C_L (pF)	Notes	Conditions
		Min	Max	Min	Max	Min	Max			
T30C	SBHW#, SBLW# DELAY FROM UCMD#	5	35	5	35	5	35	50	10, 25	
T30D	SWDIR DELAY FROM UADL#	2	40	2	40	2	40	100	12	
T30E	DWHE# DELAY FROM UADL#	2	40	2	40	2	40	50		
T30F	DWHE# DELAY FROM UCMD#	5	45	5	45	5	45	50	25	
T30G	DBE1#, DBE2# DELAY FROM UADL#	2	40	2	40	2	40	25		
T30H	DBE1#, DBE2# DELAY FROM UCMD#	5	45	5	45	5	45	25	25	
T30I	SWAP1-4# DELAY FROM UADL#	2	40	2	40	2	40	25	25	
T30J	SWAP1-4# DELAY FROM UCMD#	5	35	5	35	5	35	25	13, 25	
T30K	SBHW#, SBLW# ACTIVE DELAY FROM SCLK	2	40	2	40	2	40	50	11	
T30L	SWDIR DELAY FROM UCMD#	5	40	5	40	5	40	100	13, 25	
T30M	SWDIR DELAY FROM DS16RTN, DS32RTN	2	35	2	35	2	35	100	12, 25	
T30N	SWAP1-4# DELAY FROM UCMD#	2	35	2	35	2	35	25	11, 25	
T31	DBL1-3 SETUP TO UCMD#, UMMCMD#	−4		−4		−4		25		
T32	WDL DELAY	2	32	2	26	2	21	50		
T33	DBL1-3 DELAY ↑	2	27	2	25	2	22	25		
T34A	MHW#, MLW# ACTIVE DELAY FROM SCLK ↑	2	30	2	25	2	25	50	16	
T34B	MHW#, MLW# INACTIVE DELAY FROM SCLK ↑	2	30	2	25	2	32	50	17, 21, 24	
T34C	MDIR DELAY FROM SCLK	2	40	2	40	2	40	125	22, 24	
T34D	MHW#, MLW# DELAY FROM UADL#	2	45	2	45	2	45	50	25	
T34E	MHW#, MLW# DELAY FROM UCMD#	5	40	5	40	5	40	50	25	
T34F	MDIR DELAY FROM UADL#	2	45	2	45	2	45	125	25	
T34H	ROMOE# DELAY FROM SCLK	2	28	2	28	2	28	50		
T34I	MHW#, MLW# ACTIVE DELAY FROM SCLK ↓	2	35	2	35	2	35	50	18, 23, 25	
T34J	MHW#, MLW# INACTIVE DELAY FROM SCLK ↑	2	35	2	35	2	35	50	19, 24, 25	
T35	FCE# SETUP	15		15		10				
T36	FCE# HOLD	3		3		3				
T37	ROMEN#, RAMEN# SETUP	20		19		19				
T38A	DS16RTN, DS32RTN SETUP TO SCLK	30		30		25			6	
T38B	DS16RTN, DS32RTN SETUP TO UADL# ↑	15		15		15			3	
T41	RDYRTN# SETUP TO SCLK	10		10		10			9	
T42	RDYRTN# HOLD FROM SCLK	4		4		4				
T43	IOEN#, FDACK# SETUP TO UADL# ACTIVE	10		10		10			25	
T44A	P0-P3 SETUP TO SCLK	0		0		0			11	
T44B	P0-P3 SETUP TO UCMD#	3		3		3			13, 25	
T45	P0-P3 HOLD	12		12		12				
T46A	CHRDY DELAY FROM IOEN#	2	30	2	30	2	30	25		
T46B	CHRDY DELAY FROM STATUS	2	25	2	25	2	25	25	25	
T46C	CHRDY ACTIVE FROM MB COMMAND	100		90		70		25	1, 20	
T47A	MB COMMAND DELAY FROM UCMD# ACTIVE	75		75		75		100	1, 20	
T47C	MB COMMAND DELAY FROM UCMD# INACTIVE	3	40	3	40	3	40	100	1, 25	
T48	MB COMMAND PULSE WIDTH	250		225		190		100	1, 12, 20	

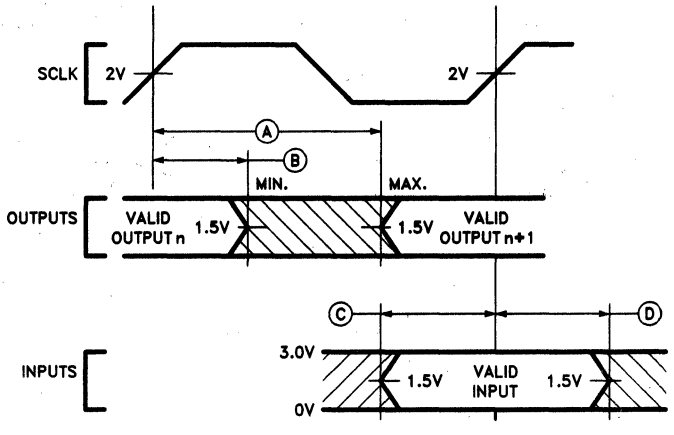
NOTES:

1. MB Commands include IOR, IOW, INTA#, VMRD# and VMWR#.
2. These specs and cycle edge definitions support a worst case "effective" data setup of 40 ns for a 385 system at 20 MHz. (Effective setup means setup to the "386-like" front end created by the 385.)
3. Spec applies only when master resides on Micro Channel.
4. RSEL0,1 should be tied high or low, or else driven from a combinatorial address decode.
5. UMMCMD# is only driven when the CPU is master.
6. Applies only when CPU is master.
7. T14A applies to the later of ADL# ↑ or CMD# ↓.
8. T13B, T14B apply to CPU or DMA master.
9. RDYRTN# is an asychronous input. Meeting T41 simply guarantees recognition at a particular clock edge.

NOTES:
10. Applies when DMA is master.
11. Applies when CPU is master.
12. Applies when CPU or DMA is master.
13. Applies when DMA or channel master is master.
14. Functional Spec . . . Not Tested (= 64 SCLK Periods).
15. Functional Spec . . . Not Tested (= 8 SCLK Periods).
16. READS AND PIPELINED WRITES
17. READS
18. NON-PIPELINED WRITES
19. WRITES
20. Functional Spec . . . Not Tested

NOTES (82308HS-25 ONLY):
21. Contention will not occur when a write immediately follows a read because the 82385 insures at least one BTI state between a read followed by write sequence during which the data bus remains tri-stated.
22. MDIR is generated and speced from SCLK ↑ instead of from SCLK ↓ at it is in the 82308-16/20.
23. In non-piped writes, MHW# and MLW# are generated from the phase 2 SCLK ↑ edge rather than from SCLK ↓ as in the 82308-16/20.
24. Since MDIR toggles from the same clock edge that MHW# and MLW# are de-asserted from, contention is prevented by 82308-25 internal design such that MDIR is guaranteed to be slower than MHW# or MLW#.
25. Only tested when UADL#, UCMD#, US0#, and US1# are inputs; i.e., when Microchannel is master.

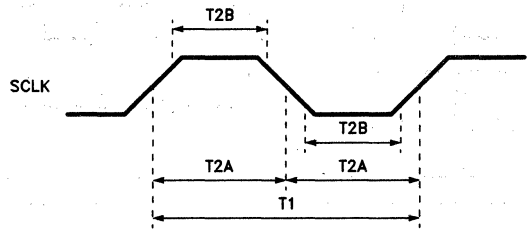
DRIVE LEVELS AND MEASUREMENT POINTS FOR A.C. SPECIFICATIONS



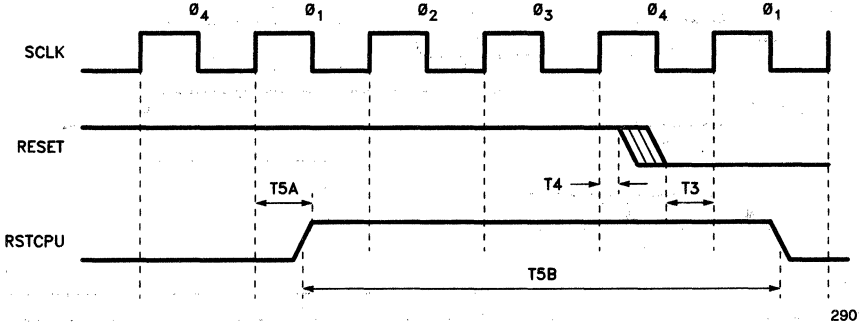
290187-25

LEGEND:
A. Maximum Output Delay Specification.
B. Minimum Output Delay Specification.
C. Minimum Input Setup Specification.
D. Minimum Input Hold Specification.

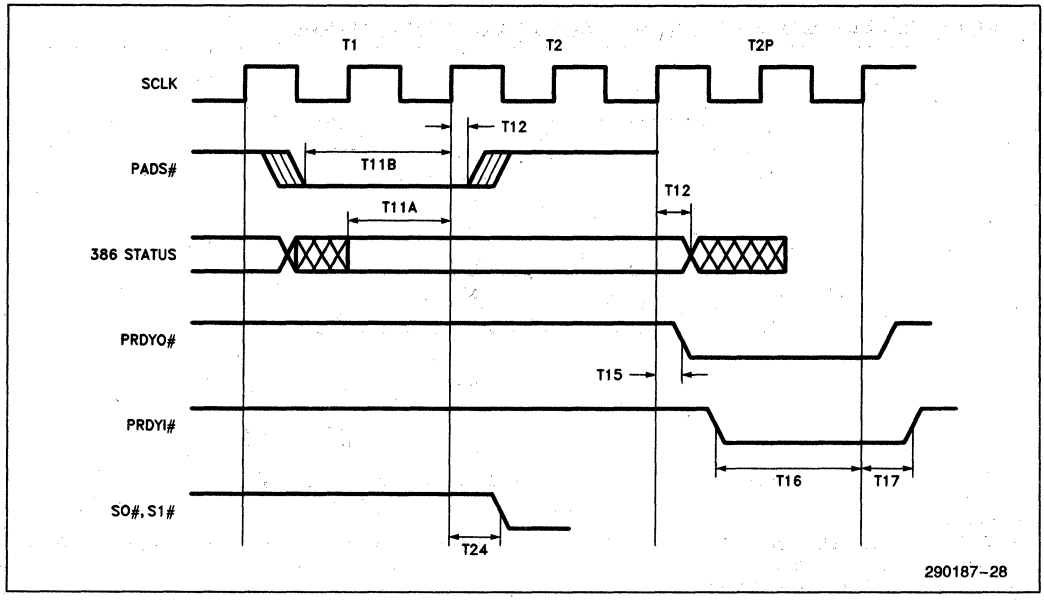
Input waveforms have tr ≤ 2.0 ns from 0.8V to 2.0V.

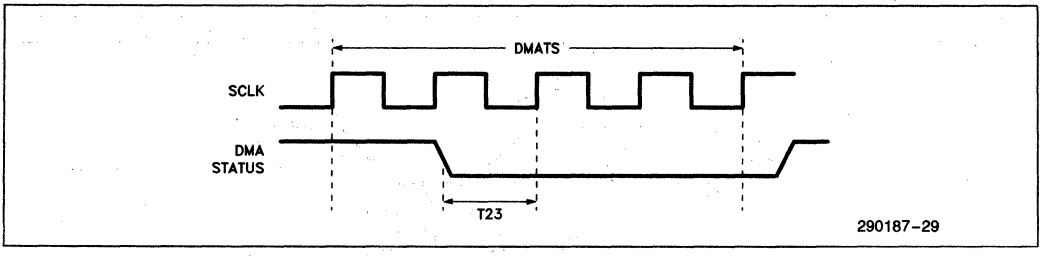


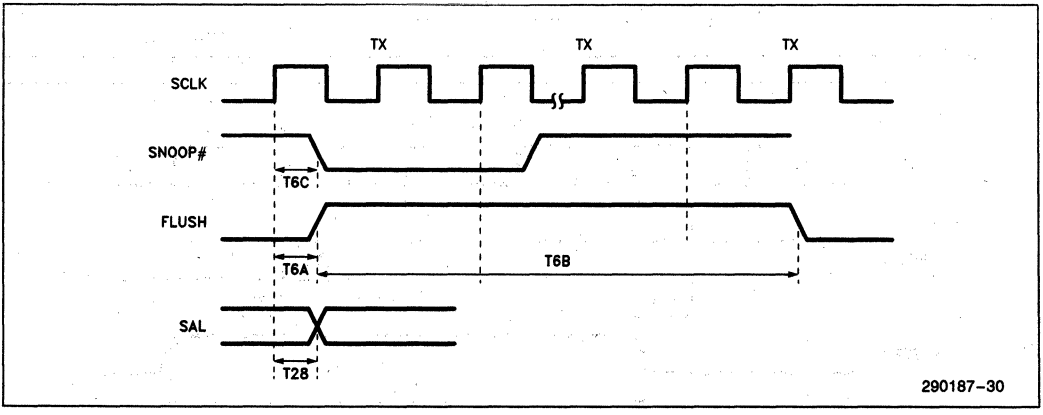
290187-26

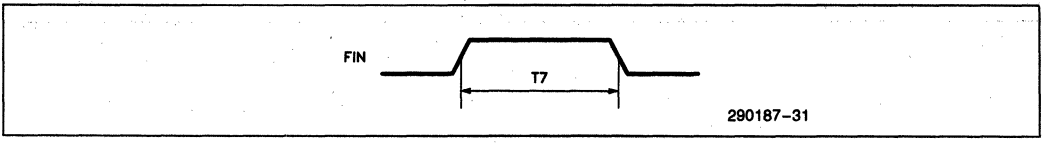


290187-27

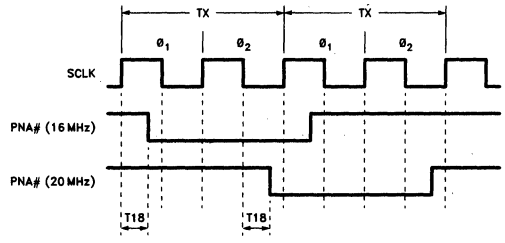

T1
T2
T2P
SCLK
T12
PADS#
T11B
T12
T11A
386 STATUS
PRDYO#
T15
PRDYI#
T16
T17
SO#, S1#
T24
290187-28


DMATS
SCLK
DMA STATUS
T23
290187-29

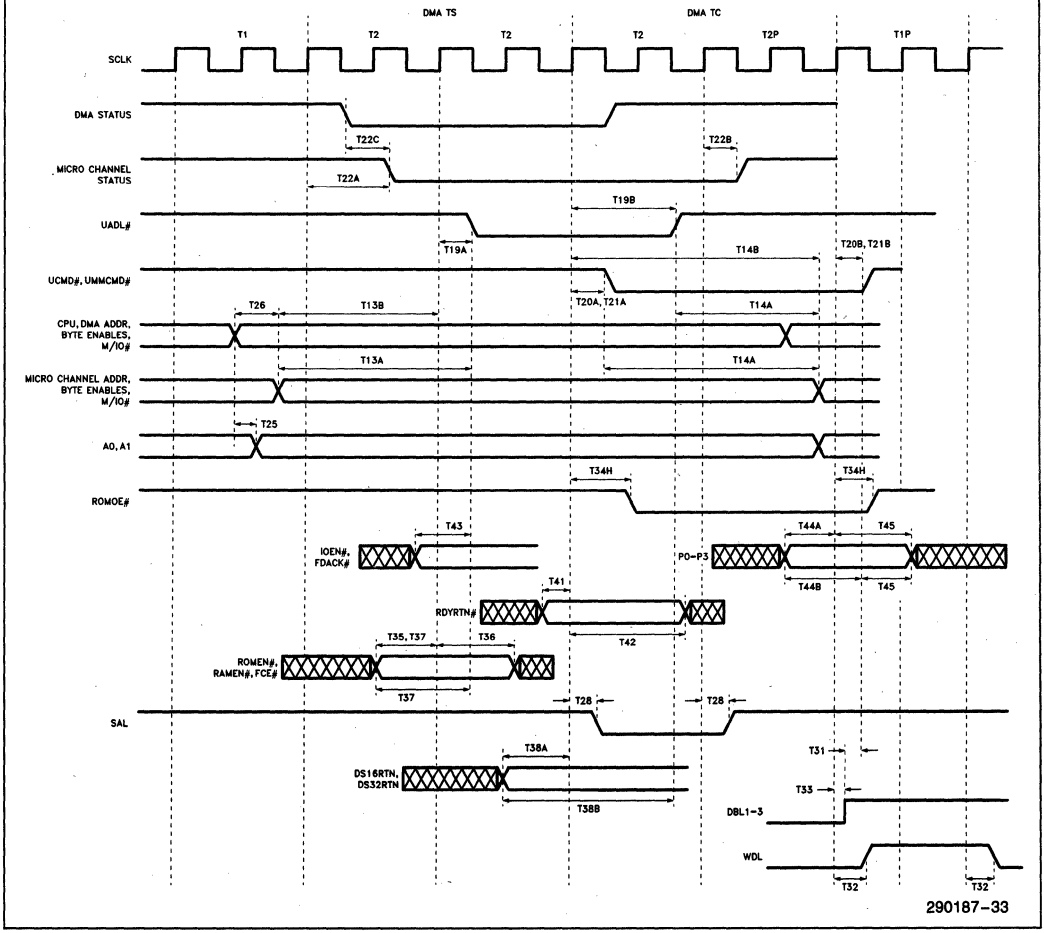

TX
TX
TX
SCLK
SNOOP#
T6C
FLUSH
T6A
T6B
SAL
T28
290187-30


FIN
T7
290187-31

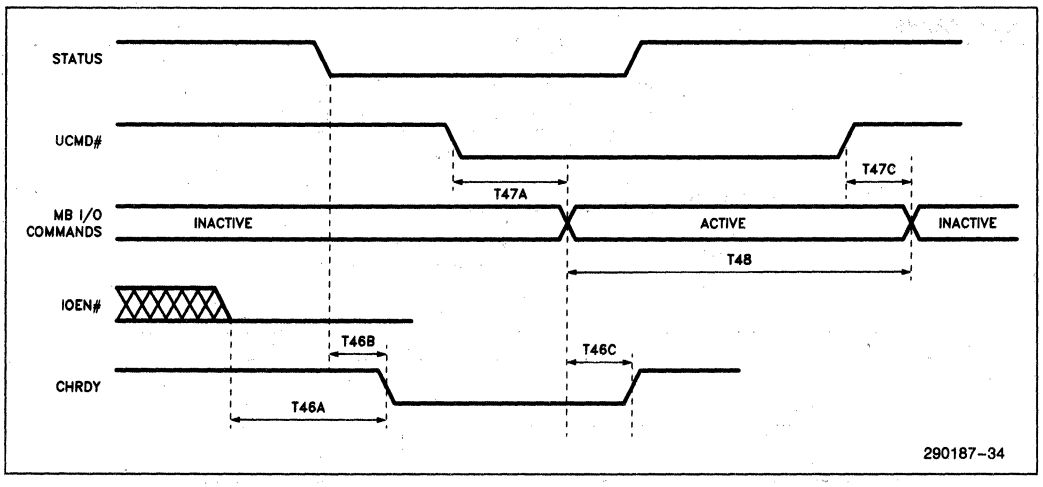
PNA# TIMING



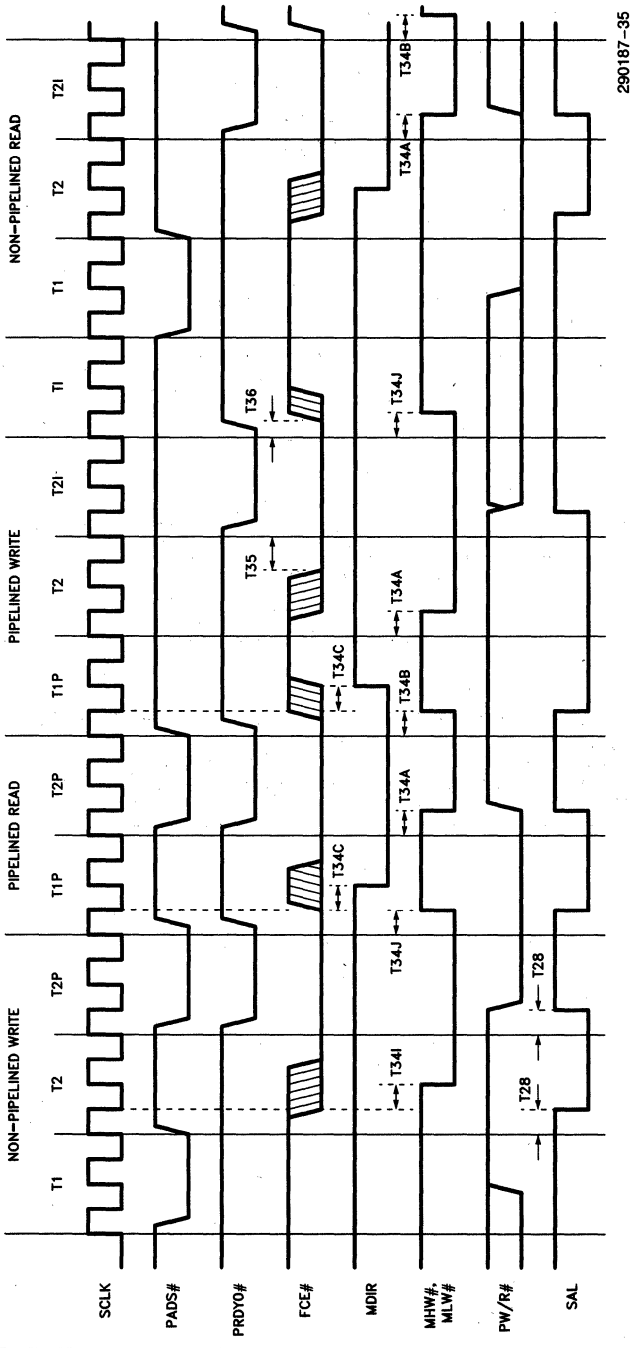
290187-32



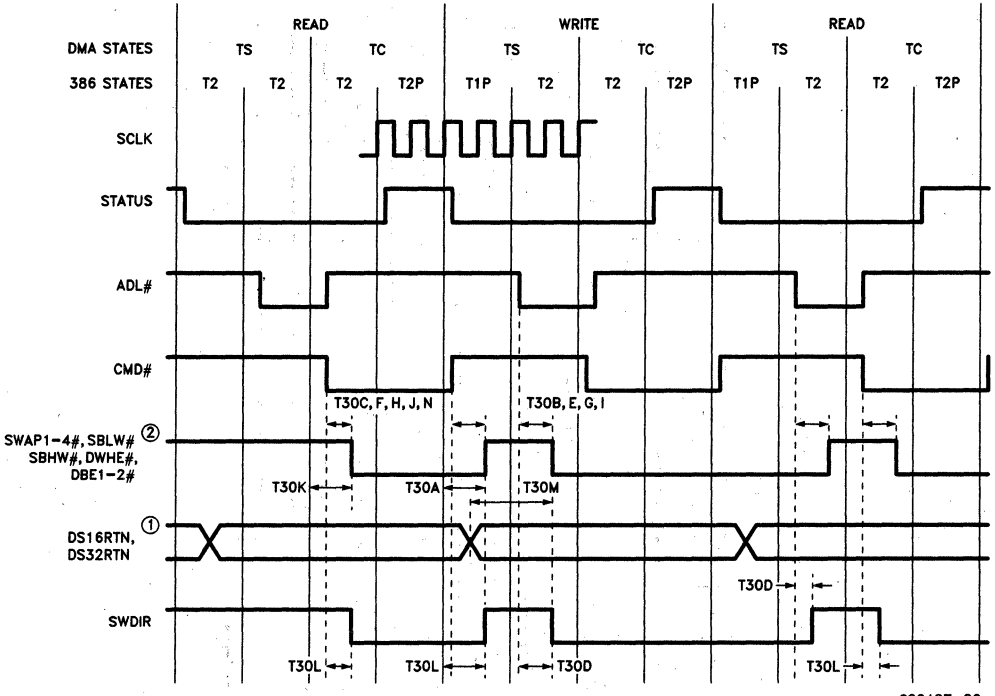
290187-33


STATUS
UCMD#
MB I/O COMMANDS
INACTIVE
ACTIVE
INACTIVE
T47A
T47C
T48
IOEN#
CHRDY
T46B
T46C
T46A
290187-34

MEMORY DATA BUFFER CONTROL
80386 ACCESSES TO MOTHERBOARD DRAM



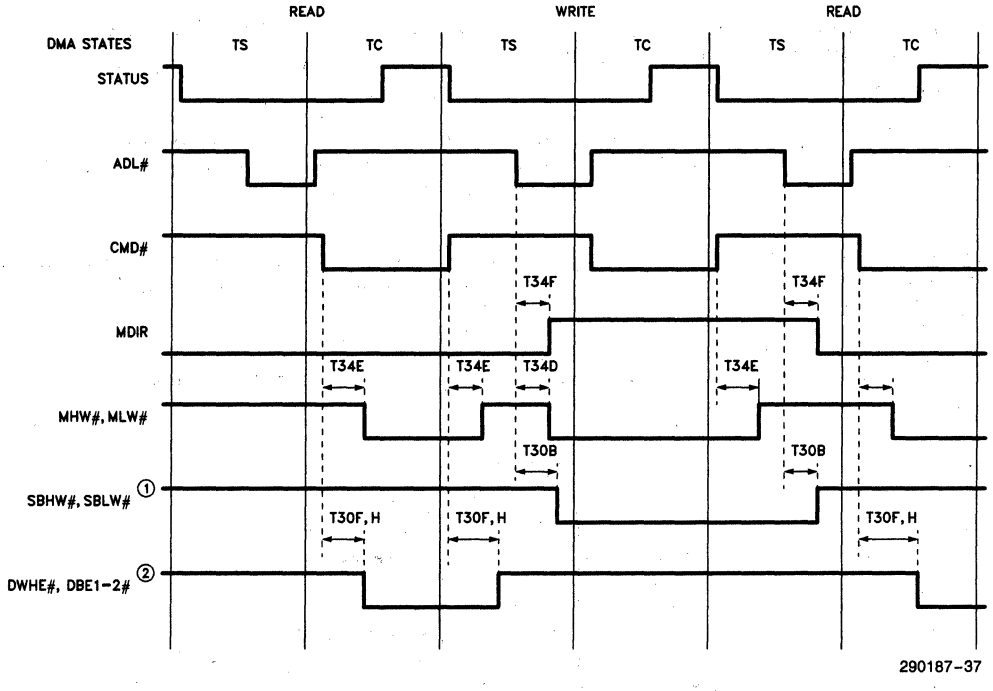
Micro Channel DATA BUFFER CONTROL 386/DMA MASTER, DATA STEERING FOR CHANNEL MASTER TO CHANNEL SLAVE



290187-36

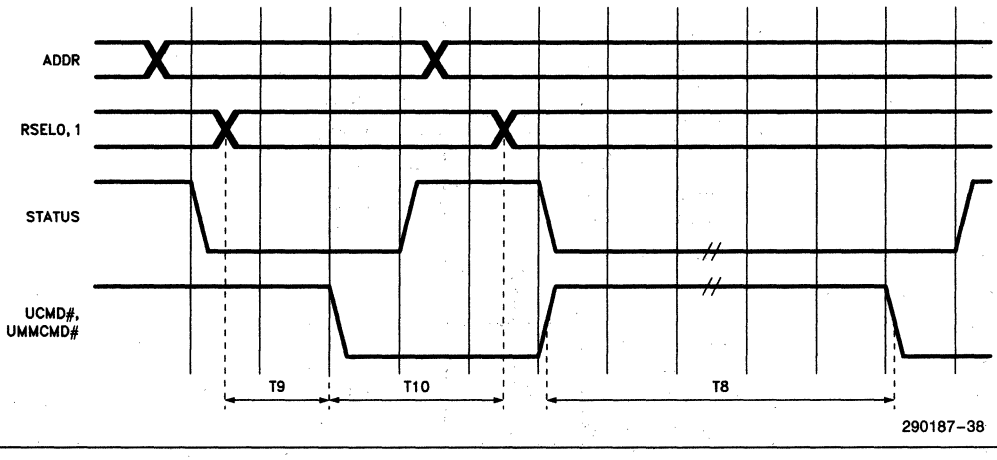
①T30D applies to SWDIR only when all involved combinatorial data size inputs (DS16RTN, DS32RTN) are available early enough not to delay SWDIR.
②T30A and T30K apply only to SBHW# and SBLW#, and only when the 386 is master.
③T30B applies here when DMA data is routed to system board memory from the lower half of the data bus (D0–D15) to the upper half (D16–D31).

MEMORY/Micro Channel DATA BUFFER CONTROL DMA/CHANNEL MASTER ACCESSES TO MB DRAM



①Channel Master writes to MB DRAM.
②Channel Master reads from MB DRAM.

(HARDWARE ENFORCED) I/O RECOVERY TIMING



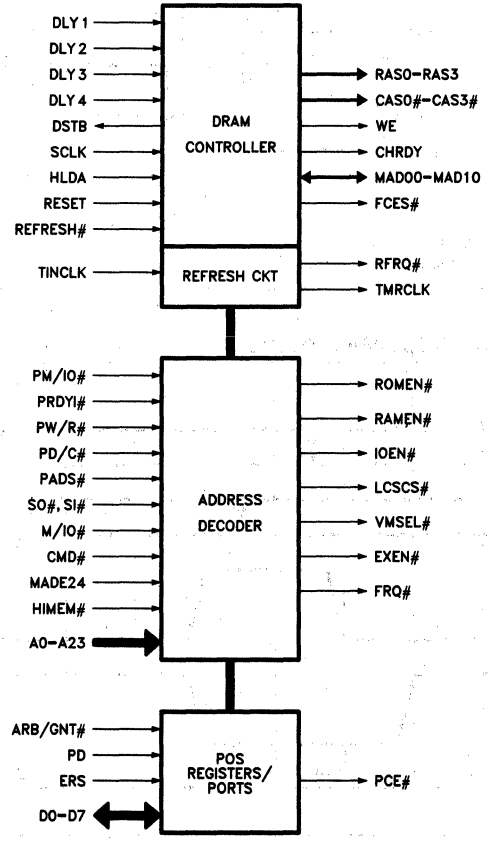
82309
ADDRESS BUS CONTROLLER

- **Address Decoder**
- **DRAM Controller ... Up to Four Banks of Page Interleaved Memory (Max 16M)**
- **Refresh Timer**
- **Integrated I/O Ports and Registers**
- **Low Power CHMOS Technology**
- **100-Pin Plastic Quad Flat Packaging**
  (See Packaging Spec., Order # 231369)

The 82309 Address Bus Controller provides Address decoding for devices on the motherboard, including the shadowed DRAM address of the ROM BIOS. The Address Bus Controller also has integrated DRAM controller, Refresh Timer and miscellaneous registers for memory control and error recovery, specifically ports E0, E1, E3, E4, E5, E7 and 103.

The 82309 Address Bus Controller provides the designer several price/performance choices for the configuration of up to 16 MBytes of Page Interleave DRAM memory on the motherboard. Up to four banks of 256K, 1M and 4M DRAMs are supported.

The 82309 Address Bus Controller generates periodic refresh requests to the 82307 DMA controller to run refresh cycles. The 82309 does not use the Refresh Address generated by the DMA controller but provides its own refresh address to the 256K, 1M and 4M DRAMs.



November 1988
Order Number: 290188-001

PORTS AND REGISTERS

Configuration bits SS1 and SS2 control the function of the Ports and Register Block. The Ports and Register Block, in turn, control the function of the Refresh Timer and the address mapping of the motherboard DRAMs and the BIOS EPROMs.

SS1 and SS2 essentially select one of four definitions of the memory encoding registers (E0, E1), error trace registers (E3, E4, E5, E7), and motherboard POS setup port (103). These defintions are depicted in Table 0, and go by the names System A, System B, System C and System D.

System A presents a Model 50/60 compatible definition of these ports. Specifically, Port 103 is defined as it is in the IBM PS/2 Model 50/60 Technical Reference and ports E0–E7 are non-existent. System B presents a Model 80 compatible definition of these ports, as detailed in the IBM PS/2 Model 80 Technical Reference.

System B has a limitation in that due to the definition of the card enable bits in ports E0 and E1 (described later), it is limited to 4 Mbytes of system board memory. System C overcomes this by making the card enable bits "free form"; i.e., accessible as read/write bits, but otherwise meaningless in terms of their effect on the system. System C allows a Model 80 type system to provide up to 16 Mbytes of system board memory.

System D provides a Model 50/60 compatible definition of port 103 and a Model 80 compatible definition of ports E0–E7. This system is targeted for designs that wish to present a Model 50/60 port definition, but wish to make use of features provided in the Model 80 register set, specifically the ability to copy ROM into RAM for increased performance. This system requires external logic (approx. ½ of a 16L8 PAL) that essentially makes E0–E7 disappear from a software point of view once the ROM has been copied into RAM. (Details will be provided in the forth coming Intel *Designers Guide for Micro Channel Compatible Implementation.*)

In systems A and D, bit 0 (the Memory Enable Bit) is the only accessible bit in Port 103. This bit can be accessed via channel I/O Read and/or Write operations. When Port 103 is read only bit 0 is driven, all other data bus bits remain tristated. This bit is set to a 1 by RESET. When the Memory Enable Bit = 0 all of the motherboard DRAM is disabled (but still refreshed). In both systems A and D, the refresh timer produces a 400 ns pulse every 15.12 μs. In system D, mapping is controlled by ports E0 and E1, as de-

Table 0. Configuration Bits SS1, SS2 Definition

Config Bits		System	Description
SS1	SS2		
0	0	A	Model 50/60 Compatible Port 103(1) Registers E0–E7 Non-Accessible
1	0	B	Model 80 Compatible Port 103 Error Trace Registers E3, E4, E5 and E7 Accessible Memory Encoding Registers E0 and E1 Accessible Compatible Card Enable Bits in E0 and E1
1	1	C	Model 80 Compatible Port 103 Error Trace Registers E3, E4, E5 and E7 Accessible Memory Encoding Registers E0 and E1 Accessible Free Form Card Enable Bits In E0 and E1
0	1	D	Model 50/60 Compatible Port 103 Error Trace Registers E3, E4, E5 and E7 Accessible (But Not Typically Used) Memory Encoding Registers E0 and E1 Accessible Free Form Card Enable Bits In E0 and E1

NOTES:
1. Port 103 is a motherboard POS port; i.e., accessible only when the motherboard is in Setup Mode.

scribed in a moment. In system A, the other functions of the ports and register block are as follows:

— The Split in the first megabyte is located at 640 Kbytes.

— If the motherboard DRAM space equals 16 Mbytes then the remaining DRAM is disabled, otherwise the remaining 384 Kbytes are remapped to the first 384 Kbytes past the end of the motherboard DRAM address space. (i.e., if there are 4 Mbytes of DRAM then the split is remapped to address 00400000 → 0045FFFF.)

— The BIOS EPROMs are mapped to both 000E0000 → 000FFFFF and FFFE0000 → FFFFFFFF.

In systems B and C, port 103 is defined as follows:

— Port 103 bit 0 (the Memory Enable Bit) is not accessible.

— Port 103 bit 1 (the Refresh Rate Bit) is accessible for write operations only. If this bit is a 1 then the Refresh Timer produces an approximately 400 ns long pulse once every 15.12 μs. If this bit is a 0 then the Refresh Timer produces a continuous stream of 400 ns pulses with a period of approximately 800 ns. This bit is set to a 1 by RESET.

In systems B, C and D, ports E0, E1, E3, E4, E5 and E7 are defined as follows:

— Four of the Read only Micro Channel Error Trace Registers (Ports 00E3, 00E4, 00E5 and 00E7) are accessible. (Typically, a system D design will not utilize these registers and will thus not require any external logic to implement any error register support.) These registers sample SA<02:23>, M/IO#, D/C# and ARB/GNT# on every rising edge of the ERS input pin. The bit assignments for these registers are as follows:

Bit	00E3	00E4	00E5	00E7
7	SA23	SA15	SA07	—
6	SA22	SA14	SA06	—
5	SA21	SA13	SA05	—
4	SA20	SA12	SA04	—
3	SA19	SA11	SA03	—
2	SA18	SA10	SA02	—
1	SA17	SA09	M/IO#	—
0	SA16	SA08	ARB/GNT#	D/C#

These four registers are all set to 00 by RESET. When Register E7 is read, only data bus bit 0 is driven by the ABC, data bus bit 1–7 remain tristated.

— Registers E0 and E1 are accessible via the channel for both I/O read and I/O write operations. These two registers control the address mapping of both the motherboard DRAMs and the BIOS EPROMs. (The reset state of E0 and E1 is FF.) The two most significant bits of both of these registers are free form register bits and have no effect on the functioning of the ABC. The functioning of the two next most significant bits (bits 5 & 4, the card enable bits) of both of these registers are controlled by configuration bits SS2 and SS1 as discussed in a moment.

— The four least significant bits (bits 3, 2, 1 & 0) of register E1 are defined as follows:

Bit 3	2	1	0	
0	.	.	.	- Memory beyond split Enabled
1	.	.	.	- Memory beyond split Disabled
.	0	.	.	- Split is at 640K (000A0000)
.	1	.	.	- Split is at 512K (00080000)
.	.	0	.	- BIOS ROMs deactivated in 000E0000 to 000FFFFF BIOS ROMs active in FFFE0000 to FFFFFFFF Shadow RAM Write Protected
.	.	1	.	- BIOS ROMs active in 000E0000 to 000FFFFF BIOS ROMs active in FFFE0000 to FFFFFFFF Shadow RAM Writeable.
.	.	.	0	- Parity Checking enabled
.	.	.	1	- Parity Checking disabled

If the memory beyond the Split is enabled by bit 3 then the four least significant bits (bits 3, 2, 1 & 0) of register 00E0 define the address range in memory where the portion of the first megabyte of system RAM beyond the split will be remapped. Bits 3, 2, 1 & 0 of this register correspond to address bits 23, 22, 21 & 20 of the remap location for this memory.

Bit 2 of register E1 defines the partitioning of the first megabyte of the motherboard DRAM. Figure 0 details the effect of this bit. The S in the remap addresses represents the value of the four least significant bits of register E0.

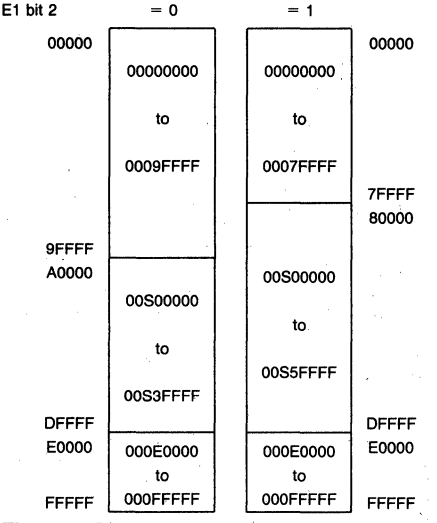


Figure 0. Partition of First Megabyte of DRAM

There is DRAM mapped in the address range 000E0000 to 000FFFFF. The function of this DRAM is controlled by bit 1 of register E1. If bit 1 = 1 then this RAM is writeable but not readable (thus the BIOS EPROMs can be Shadowed by Reading and Writing to the same address). If bit 1 = 0 then the BIOS EPROMs are disabed and this area of RAM is read enabled but write protected.

When bit 1 of register E1 is a 1 both the ROMEN# and the RAMEN# signals will respond to accesses in the range 000E0000 to 000FFFFF. In this way, the 82308 Bus Controller knows to direct reads to ROM and writes to RAM to allow shadowing. (In system D, if memory is disabled via bit 0 of port 103, then ROM is enabled in 000E0000 to 000FFFFF, regardless of the status of bit 1 in E1.)

Bit 0 of register E1 is output to the Bus Controller on the PCE# pin for use as an (active low) Parity Check Enable control bit.

— In system B, the amount of the physical motherboard DRAM that is accessable is controlled by the card enable bits (bits 5 and 4 of registers E0 and E1). These four bits act as enables (active low) for each of the first four megabytes of the physical motherboard DRAM space. Any additional DRAM controlled by the ABC will be refreshed but is otherwise disabled.

Register				Function
E0		E1		
Bit 5	Bit 4	Bit 5	Bit 4	
0	X	X	X	Megabyte #3 Enabled
1	X	X	X	Megabyte #3 Disabled
X	0	X	X	Megabyte #2 Enabled
X	1	X	X	Megabyte #2 Disabled
X	X	0	X	Megabyte #1 Enabled
X	X	1	X	Megabyte #1 Disabled
X	X	X	0	Megabyte #0 Enabled
X	X	X	1	Megabyte #0 Disabled

All megabytes that are enabled by these bits are mapped into one continuous block (with the exception of the Split from the first active megabyte) starting at address 00000000. (Thus if megabyte #0 is disabled, then the rest of the megabytes are remapped down to the range 00000000 to 002FFFFF, etc.)

In systems C and D, bits 5 and 4 of both registers E0 and E1 are free form register bits and have no effect on the functioning of the ABC.

DRAM CONTROLLER

The DRAM controller supports page interleaved memory designs in the configurations shown in Table 1. This table also details which channel address bits map to which DRAM address bits. Note that even though options D and G are two-bank options, the ABC thinks of these banks as 0 and 2, not banks 0 and 1, i.e., use RAS0, RAS2, CAS0# and CAS2#.

Table 1 describes the basic memory configurations A through N. However, a wide variety of additional options can be easily realized by building on A through N with minimal external address decode logic. These additional options include the ability to mix DRAM types (for example 256K and 1M DRAMs in the same system), and allow for a great deal of flexibility in memory upgrade paths. Examples of how to do this are included in the *Designer's Guide for Micro Channel Compatible Implementation.*

Table 1. Memory Configuration Options and Channel Address-To-DRAM Address Mapping

Opt	Size	Memory Configuration Options	Page Size
A	1M	1 Bank of 256K DRAMs (x 32) Page Mode	512
B	1M	2 Banks of 256K DRAMs (x 16) Page Mode	512
C	2M	1 Bank of 1M DRAMs (x 16) Page Mode	1024
D	2M	2 Banks of 256K DRAMs (x 32) Page Mode	512
E	2M	4 Banks of 256K DRAMs (x 16) Page Mode	512
F	4M	1 Bank of 1M DRAMs (x 32) Page Mode	1024
G	4M	2 Banks of 1M DRAMs (x 16) Page Mode	1024
H	4M	4 Banks of 256K DRAMs (x 32) Page Mode	512
I	8M	1 Bank of 4M DRAMs (x 16) Page Mode	2048
J	8M	2 Banks of 1M DRAMs (x 32) Page Mode	1024
K	8M	4 Banks of 1M DRAMs (x 16) Page Mode	1024
L	16M	1 Bank of 4M DRAMs (x 32) Page Mode	2048
M	16M	2 Banks of 4M DRAMs (x 16) Page Mode	2048
N	16M	4 Banks of 1M DRAMs (x 32) Page Mode	1024

Opt	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
A					Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws		
B					Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Bs	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	
C				Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	
D				Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Bs	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws		
E				Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Bs	Bs	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	
F			Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws		
G			Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Bs	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	
H			Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Bs	Bs	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws		
I		Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	
J		Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Bs	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws		
K		Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Bs	Bs	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	
L	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws		
M	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Bs	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	
N	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Ps	Bs	Bs	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws	Ws		

Ps ≥ Page Select Ws ≥ Word Select Bs ≥ Bank Select

NOTE:
Options A, C, F, I & L use Bank 0
Options D & G use Bank 0 & 2
Options B, J & M use Bank 0 & 1
Options E, H, K & N use all Banks

Opt	Ps											Ws										Bs	
	10	09	08	07	06	05	04	03	02	01	00	10	09	08	07	06	05	04	03	02	01	00	01	00
A	><	><	11	12	13	19	18	17	16	15	14	><	><	02	10	03	09	08	07	06	05	04	><	><
B	><	><	11	12	13	19	18	17	16	15	14	><	><	02	01	03	09	08	07	06	05	04	><	10
C	><	11	20	12	13	19	18	17	16	15	14	><	01	02	10	03	09	08	07	06	05	04	><	><
D	><	><	20	12	13	19	18	17	16	15	14	><	><	02	10	03	09	08	07	06	05	04	11	><
E	><	><	20	12	13	19	18	17	16	15	14	><	><	02	01	03	09	08	07	06	05	04	11	10
F	><	21	20	12	13	19	18	17	16	15	14	><	11	02	10	03	09	08	07	06	05	04	><	><
G	><	21	20	12	13	19	18	17	16	15	14	><	01	02	10	03	09	08	07	06	05	04	11	><
H	><	><	20	21	13	19	18	17	16	15	14	><	><	02	10	03	09	08	07	06	05	04	11	12
I	22	21	20	12	13	19	18	17	16	15	14	01	11	02	10	03	09	08	07	06	05	04	><	><
J	><	21	20	22	13	19	18	17	16	15	14	><	11	02	10	03	09	08	07	06	05	04	><	12
K	><	21	20	22	13	19	18	17	16	15	14	><	01	02	10	03	09	08	07	06	05	04	11	12
L	23	21	20	22	13	19	18	17	16	15	14	12	11	02	10	03	09	08	07	06	05	04	><	><
M	23	21	20	22	13	19	18	17	16	15	14	01	11	02	10	03	09	08	07	06	05	04	><	12
N	><	21	20	22	23	19	18	17	16	15	14	><	11	02	10	03	09	08	07	06	05	04	13	12

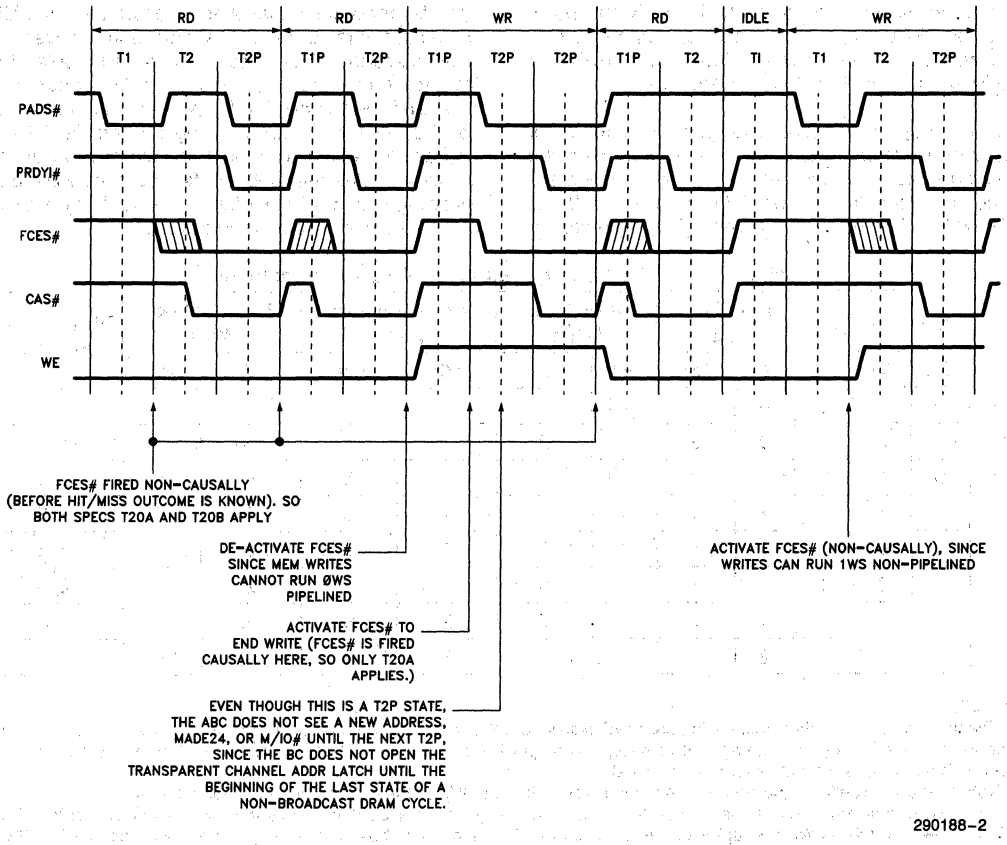
Typically, zero wait state pipelined page hit performance can be achieved at 16 MHz using 100 ns or 120 ns DRAMs, resulting in an aggregate of 0.5 to 0.8 wait states on average. The same DRAMs at 20 MHz will yield 1 wait state page hits.

At power-up, the 82309 Address Bus Controller samples its memory address bus to determine the desired system configuration. (This operation is described in detail later in the data sheet under "MAD BUS RESET CONFIGURATION".) The three configuration switches, C0, C1 and C2 are used to select a specific performance level as measured in page hit/page miss wait states. DRAM selection involves not only selecting a DRAM, but also choosing delay line taps to control the sequence of DRAM control signals, and then choosing the performance level that can be reliably supported using a particular DRAM and set of delay line taps. The next several pages describe all the available configuration options, and following this is a DRAM/Delay Tap selection guide along with some sample calculations.

Table 2. 82309 ABC Configuration and CPU Performance(3)

	Config Inputs			Pipelined Read		Pipelined Write		Non-Pipelined Read		Non-Pipelined Write		Reference Figures
	C0	C1	C2	Hit	Miss	Hit	Miss	Hit	Miss	Hit	Miss	
(1)	0	0	0	0	2	1(2)	2	1	3	1(2)	3	1, 4
(1)	0	0	1	0	3	1(2)	3	1	4	1(2)	4	1, 4
(1)	0	1	0	0	4	1(2)	4	1	5	1(2)	5	1, 4
	0	1	1	1	4	1	4	2	5	2	5	2, 5
	1	0	0	1	5	1	5	2	6	2	6	2, 5
	1	0	1	1	6	1	6	2	7	2	7	2, 5
	1	1	0	1	7	1	7	2	8	2	8	2, 5
	1	1	1	2	7	2	7	3	8	3	8	3, 5

NOTES:
1. These three configuration options feature 0WS pipelined page read hits. Strapping for one of these directs the ABC to determine whether a cycle is a page hit or miss, and to generate CAS# one SCLK phase earlier then the other options. Hence, these options are only supported at 16 MHz.
2. Note that both pipelined and non-pipelined write page hits run 1WS in these three configuration options.
3. The ABC completely controls the wait state counts in memory cycles according to this table via its FCES# output. The BC can however, (via its WS# strap) insert an additional wait state beyond those stated above in memory reads. (The WS# strap is intended for cache systems, which typically require additional data setup.)



NOTE:
Once the ABC fires FCES# to terminate a DRAM cycle, it remains active until the ABC detects the actual CPU end of cycle via the PRDYI# input. This allows the BC or any other external device to extend DRAM cycles if desired.

Figure 1. 0/2, 0/3, 0/4 Page Hits
(Cycles Named According to Pipelined Read Performance)

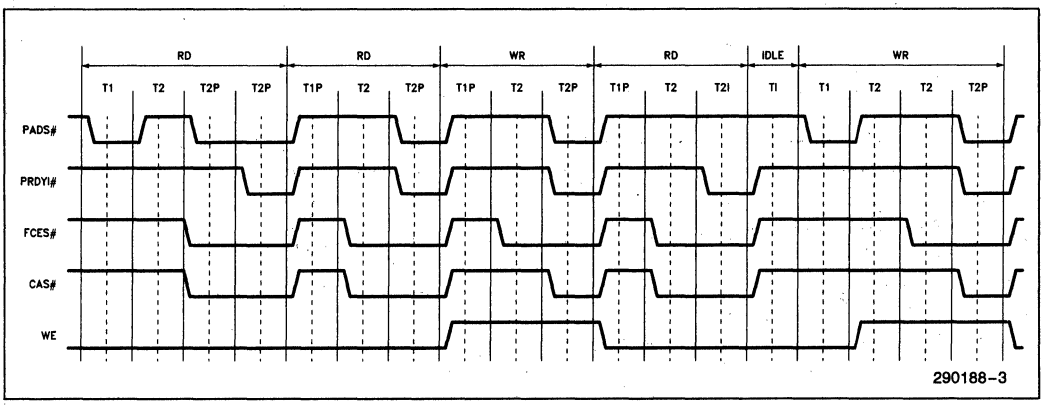


Figure 2. 1/4, 1/5, 1/6, 1/7 Page Hits (Cycles Named According to Pipelined Read Performance)

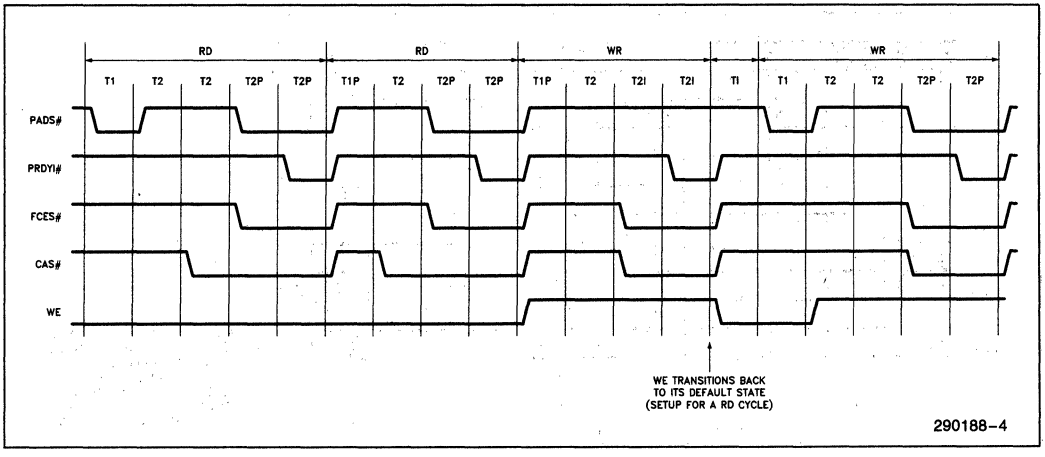


Figure 3. 2/7 Page Hit

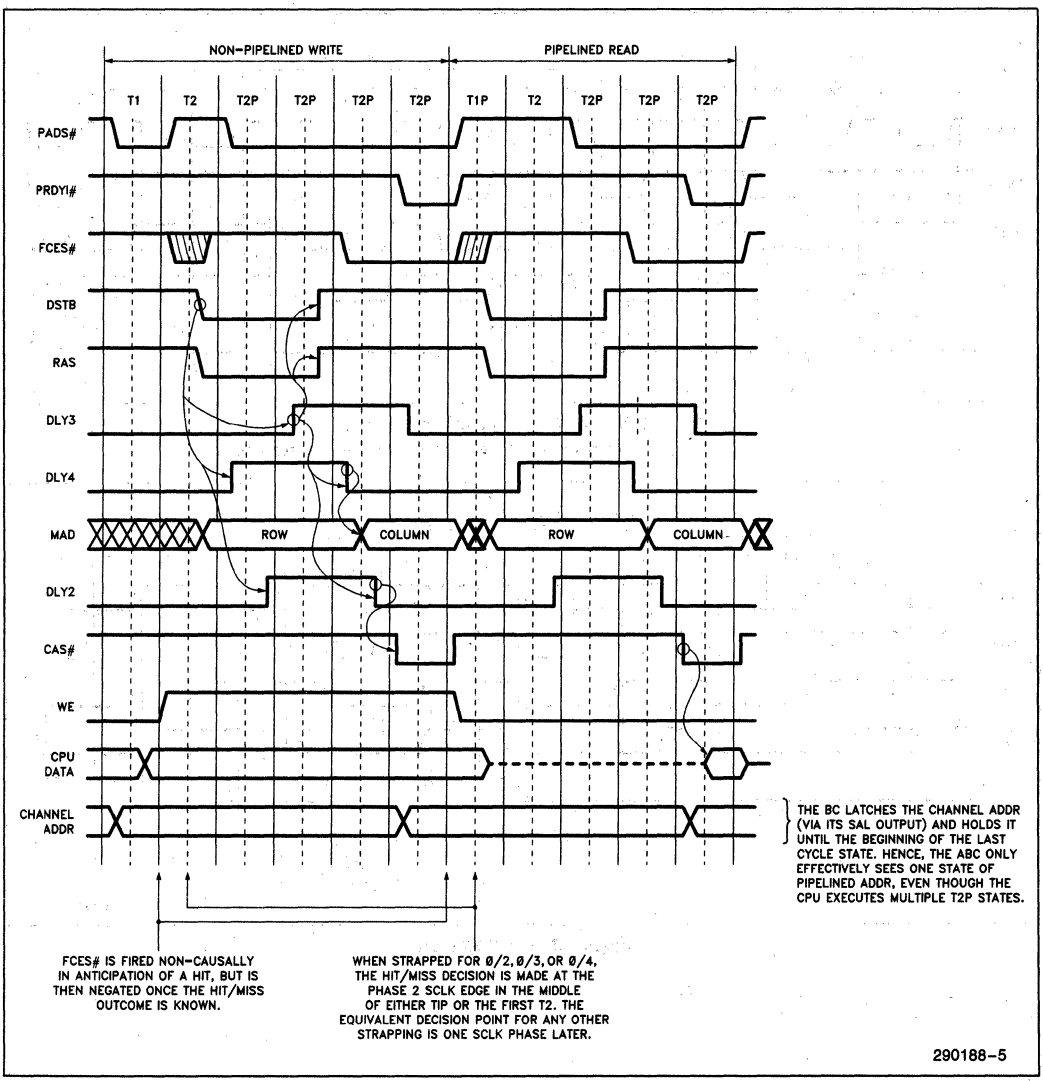


Figure 4. 0/2, 0/3, 0/4 Page Misses (Diagram Depicts 0/3 Operation. In 0/2, FCES# Fired One State Earlier. In 0/4, FCES# Fired One State Later.)

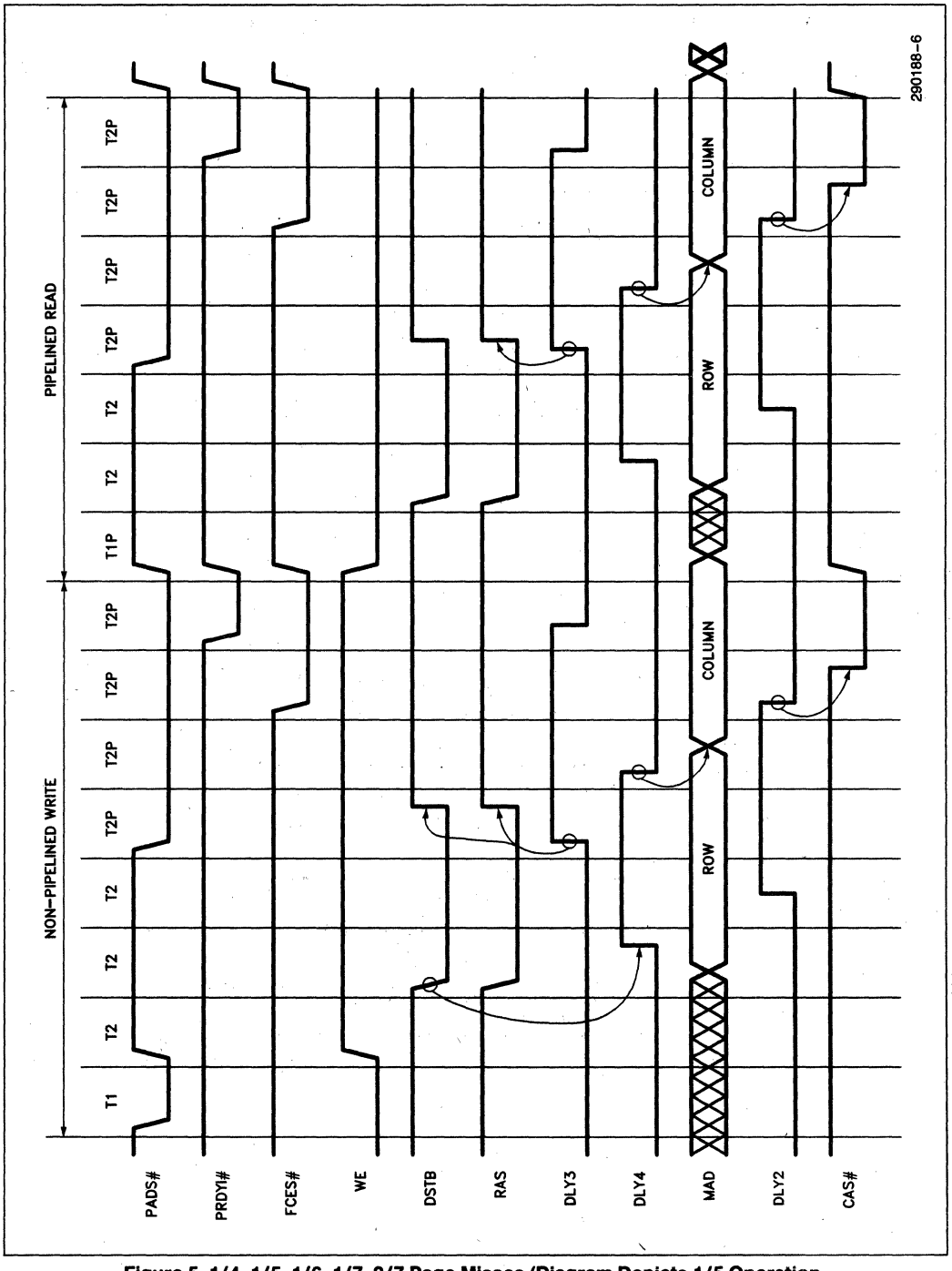
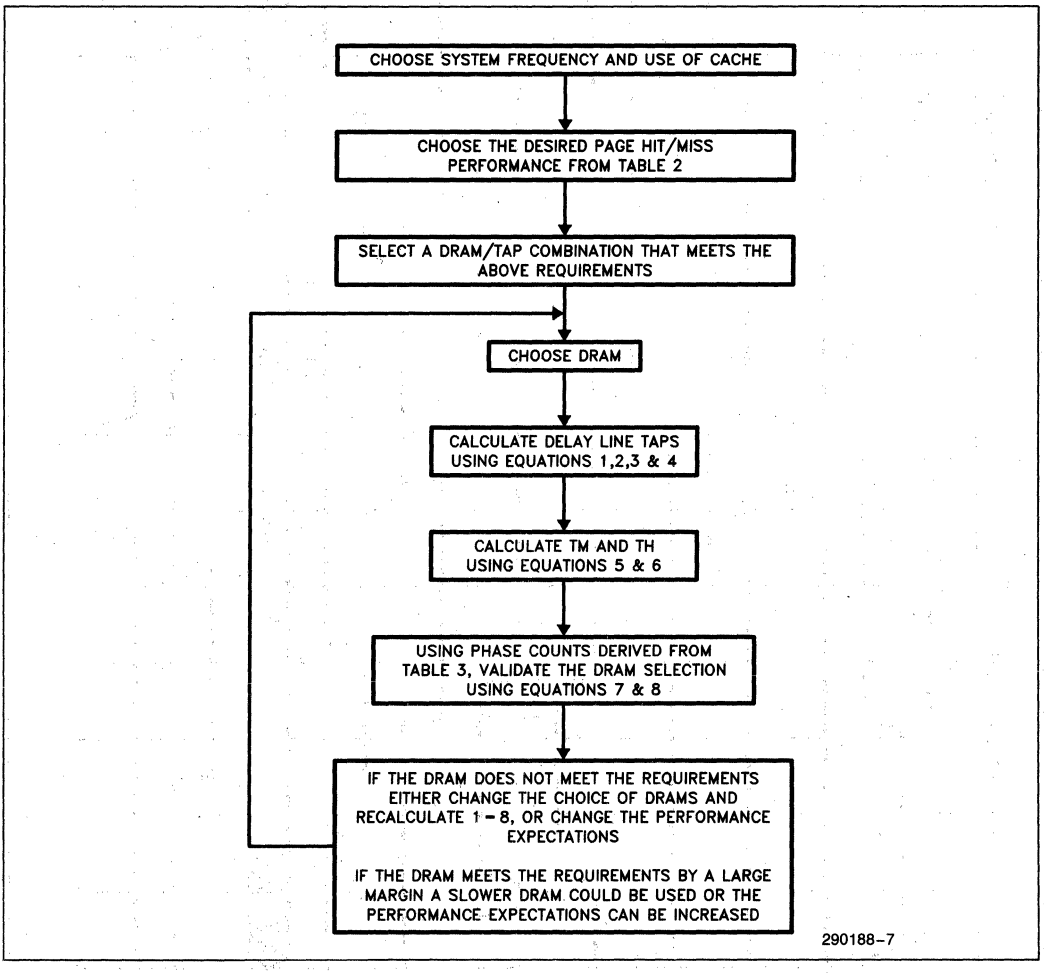


Figure 5. 1/4, 1/5, 1/6, 1/7, 2/7 Page Misses (Diagram Depicts 1/5 Operation. FCES# is Fired One State Earlier in 1/4 Operation, One State Later in 1/6 Operation, and Two States Later in Either 1/7 or 2/7 Operation.)

DRAM AND DELAY LINE TAP SELECTION

This chapter illustrates the methods that should be used to determine the delay line taps for a given DRAM, and the number of wait states a given DRAM will require.

The Flow Chart below should be used to select a DRAM/Delay line tap combination that meets the performance requirements of a system.



DELAY LINE TAP SELECTION

Function of the 4 Delay Line Taps

DLY1— Guarantees max. DRAM data from CHRDY on Micro Channel

DLY2— Guarantees the minimum RAS to CAS delay and DRAM address setup to CAS

DLY3— Guarantees min. RAS# precharge time

DLY4— Guarantees min. address hold to RAS#

Simplified DRAM Tap Selection Equations

DLY1 is the maximum of the following:

80386 System—

$$\text{Trac (Max)} - 10 \quad (1a)$$

$$\text{Trcd (Min)} + \text{Tcac (Max)} \quad (1b)$$

$$\text{Tasc (Min)} + \text{Trah (Min)} + \text{Tcac (Max)} + 20 \quad (1c)$$

80386SX System—

$$\text{Trac (Max)} - 25 \quad (1a)$$

$$\text{Trcd (Min)} + \text{Tcac (Max)} - 15 \quad (1b)$$

$$\text{Tasc (Min)} + \text{Trah (Min)} + \text{Tcac (Max)} + 5 \quad (1c)$$

DLY2 is the maximum of the following:

$$\text{Trcd (Min)} + 10 \quad (2a)$$

$$\text{Tasc (Min)} + \text{Trah (Min)} + 30 \quad (2b)$$

$$\text{DLY3} = \text{Trp (Min)} \quad (3)$$

$$\text{DLY4} = \text{Trah (Min)} + 10 \quad (4)$$

DRAM Access Time Calculations

Two access time parameters have been derived, one for hits (Th) and one for misses (Tm). These are the time from the decision to start a DRAM access to the time that data is available to the motherboard CPU.

$$\text{Th} = \text{Tcac (Max)} + \text{K1} \quad (5)$$

Tm is the maximum of the following:

$$\text{DLY3} + \text{Trac (Max)} + \text{K2} \quad \text{(RAS Path Limited)} \quad (6a)$$

$$\text{DLY3} + \text{DLY2} + \text{Tcac (Max)} + \text{K3} \quad \text{(CAS Path Limited)} \quad (6b)$$

The constants K1, K2 and K3 in equations 5 and 6 are simply a sum of all the propagation delay elements in the appropriate data access path including capacitive load derating:

$$\text{K1} = \underset{\text{(T41A)}}{\text{ABC CAS\# DLY}} + \underset{\text{(INCLUDE DERATE)}}{\text{CAS\# BUFFER DLY}} + \underset{\text{('F657)}}{\text{DATA BUFFER DLY}} = 44.5$$

$$\text{K2} = \underset{\text{(T32E)}}{\text{ABC DSTB DLY}} + \underset{\text{('AS02)}}{\text{NOR GATE DLY}} + \underset{\text{(T32G)}}{\text{ABC RAS DLY}} + \underset{\text{(INCLUDE DERATE)}}{\text{RAS BUFFER DLY}} + \underset{\text{('F657)}}{\text{DATA BUFFER DLY}} = 79$$

$$\text{K3} = \underset{\text{(T32E)}}{\text{ABC DSTB DLY}} + \underset{\text{('AS02)}}{\text{2X NOR GATE DLY}} + \underset{\text{(T34)}}{\text{ABC CAS\# DLY}} + \underset{\text{(INCLUDE DERATE)}}{\text{CAS\# BUFFER DLY}} + \underset{\text{('F657)}}{\text{DATA BUFFER DLY}} = 81.5$$

(See Figure 6 for a diagram of the timing model used.)

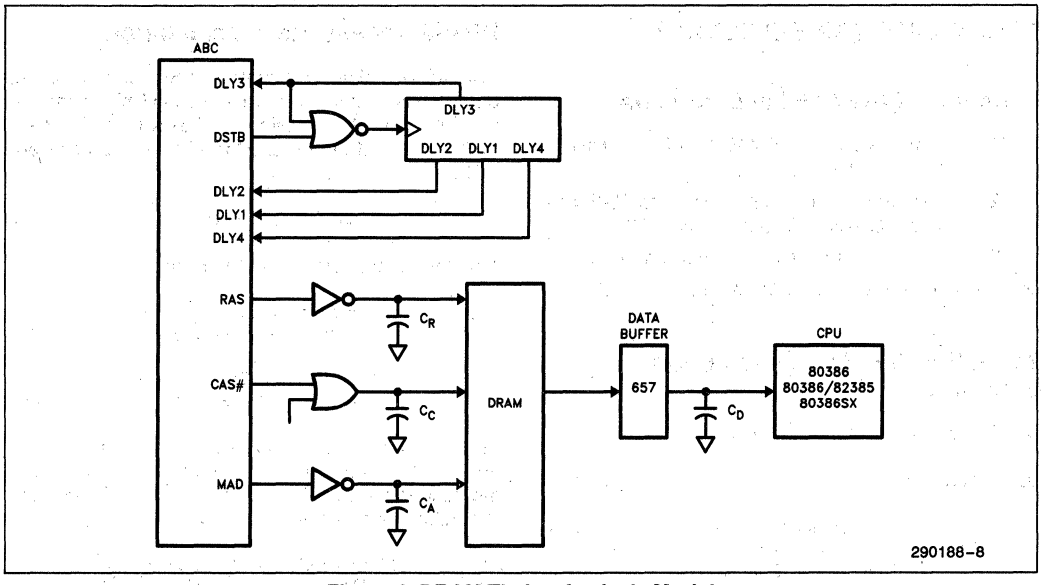


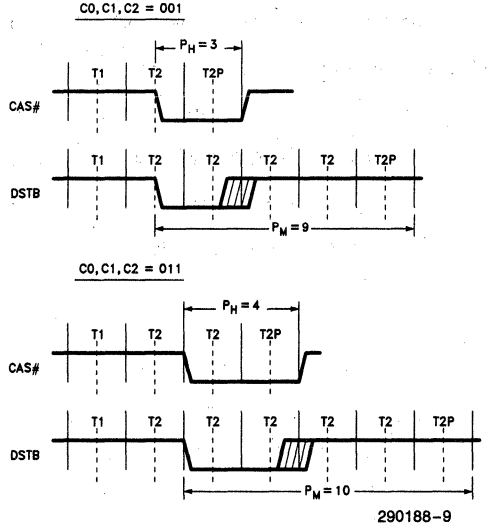
Figure 6. DRAM Timing Analysis Model

Tables 2 and 3 define the number of DRAM wait states that the motherboard CPU will see for all combinations of the configuration bits C0, C1 and C2.

Table 3. Configuration and DRAM Calculation Clock Phase Counts

Config Inputs			SCLK Phases	
C0	C1	C2	PH	PM
0	0	0	3	7
0	0	1	3	9
0	1	0	3	11
0	1	1	4	10
1	0	0	4	12
1	0	1	4	14
1	1	0	4	16
1	1	1	6	16

PH, PM Definition Examples



NOTES:
1. The phase counts are from the clock edge that either fires CAS# (Hit) or fires DSTB (Miss) to the end of cycle, as shown above.
2. Cache systems typically require additional read data setup. The BC WS# (Wait State) strap inserts an additional wait state into system board memory reads, and can be used to accommodate this increased setup if required. If WS# is tied low, then the phase counts above all increase by two.

Validation of DRAM Selection

After Th and Tm have been calculated the performance expectations of the DRAM can be checked.

First the number of clock phases both hit and miss DRAM cycles are allowed are calculated. For Configurations 0, 1 & 2 this is 2 × the number of non-pipelined waitstates + 1. For other Configurations this is just 2 × the number of non-pipelined waitstates. The phases for hits are called Ph, Pm for misses.

The following equations must then be satisfied:

$$Ph/(2*Clk\ Freq.) - Th - CPU\ Data\ setup \geq 0 \quad (7)$$

$$Pm/(2*Clk\ Freq.) - Tm - CPU\ Data\ setup \geq 0 \quad (8)$$

Two examples of Delay Line Tap Selection and DRAM Performance Verification are given below, one for an 80386 system and one for an 80386SX system.

Sample Calculation—80386 20 MHz 100 ns DRAMs 1/5 Performance

Target Dram

Key Specs (ns)

Trac	100
Trp	80
Trah	15
Trcd	25
Tasc	0
Tcac	35

Delay Line Calculations

DLY3 = Trp = 80 ns

DLY4 = Trah + 10 = 15 + 10 = 25 ns

DLY2 = Trcd + 10 = 25 + 10 = 35 ns
or
= Tasc + Trah + 30 = 0 + 15 + 30 = 45 ns

DLY1 = Trac − 10 = 100 − 10 = 90 ns
or
= Trcd + Tcac = 25 + 35 = 60 ns
or
= Tasc + Trah + Tcac + 20
= 0 + 15 + 35 + 20 = 70 ns

Delay Line Summary

DLY1 90
DLY2 45
DLY3 80
DLY4 25

Page Hit Access Time & Performance

Th = Tcac + 44.5
= 35 + 44.5 = 79.5 ns

80386 Data Setup Time = 10 ns

1 Waitstate Margin (Pipelined)

Ph/(2*CLK Freq.) − Th − 386 Data Setup = 100 − 79.5 − 10
= 10.5 ns

Page Miss Access Time & Performance

Tm is the maximum of EQN 6a and 6b.

Tm = DLY3 + Trac + 79
= 80 + 100 + 79 = 259

or

Tm = DLY3 + DLY2 + Tcac + 81.5
= 80 + 45 + 35 + 81.5 = 241.5

5 Waitstate (Pipelined) Margin

Pm/(2* CLK Freq.) − Tm − 80386 Data Setup = 300 − 259 − 10
= 31 ns

Sample Calculation—80386SX 16 MHz 100 ns DRAMs

0/3 Performance

Only DLY1 changes

DLY1 = Trac − 25 = 100 − 25 = 75 ns

Delay Line Summary

DLY1 75
DLY2 45
DLY3 80
DLY4 25

Page Hit Access Time & Performance

Th = Tcac + 44.5
= 35 + 44.5 = 79.5

80386SX Data Setup = 5 ns

0 Waitstate Margin (Pipelined)

Ph/(2* CLK Freq.) − Th − 80386SX Data Setup > = 0
93.75 − 79.5 − 5 = 9.25 ns

Page Miss Access Time & Performance

Tm = 259

(Same as for 20 MHz 386 Case)

3 Waitstate Margin (Pipelined)

Pm/(2* CLK Freq.) − Tm − 80386SX Data Setup > = 0
281.25 − 259 − 5 = 17.5 ns

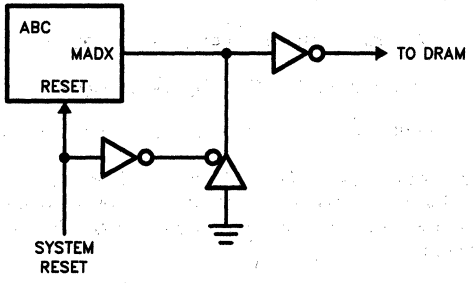
MAD BUS RESET CONFIGURATION

The ABC samples the MAD bus at the falling edge of RESET to determine system configuration as shown:

Table 4

MAD Bus Bits											Options
10	9	8	7	6	5	4	3	2	1	0	
0	0										256K DRAMs
0	1										1M DRAMs
1	1										4M DRAMs
										0	32 Bit Memory
										1	16 Bit Memory
									0		SS1 = 0
									1		SS1 = 1
							0	0			Invalid
							0	1			Single Bank
							1	0			Two Banks
							1	1			Four Banks
						0					Reserved
						1					Normal Mode
					0						C2 = 0
					1						C2 = 1
				0							C1 = 0
				1							C1 = 1
			0								C0 = 0
			1								C0 = 1
		0									SS2 = 0
		1									SS2 = 1

NOTES:
1. When either MAD09 or MAD10 is sensed as a zero, it's output driver is tri-stated, thus allowing these two pins to be tied directly to ground. For example, if 1M DRAMs are used, MAD10 should be tied to ground, since 1M DRAMs only require use of bits 0–9. MD09 should be lightly pulled up (~ 10K).
2. For MAD bits 0–8, any bit that is to be sensed as a one should be lightly pulled up. Any bit that is to be sensed as a zero must be driven low by a tri-state driver that is active while the ABC RESET input is active, and then tri-stated from the falling edge of RESET, as depicted in the figure:



290188-10

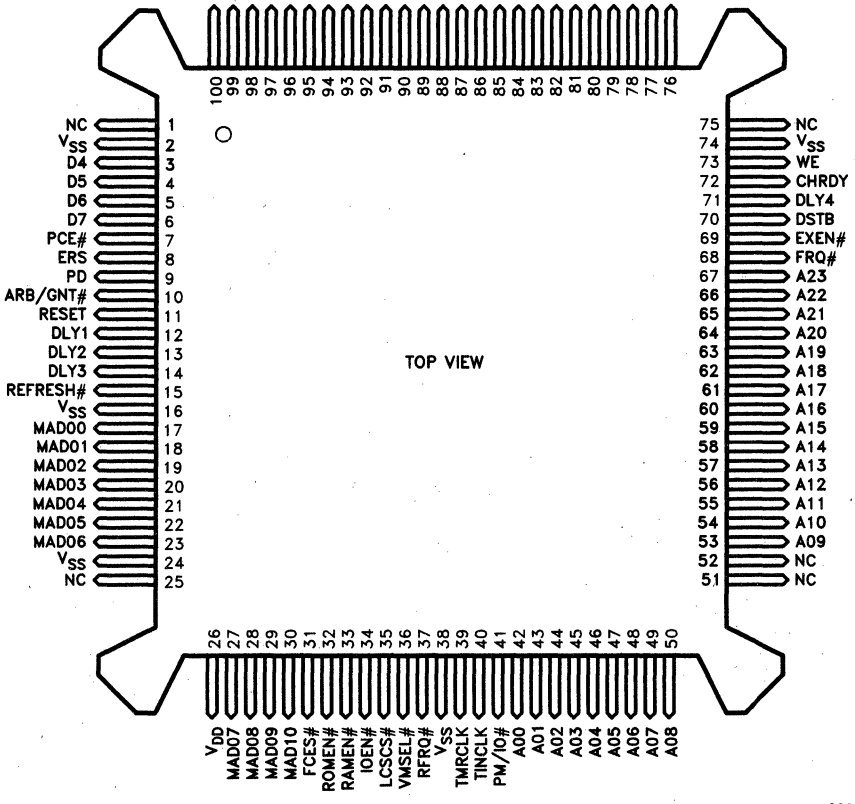
3. MAD4 sensed as a 0 is a reserved state. This bit should be lightly pulled up.
4. MAD0 is typically configured low for an 80386 system, and high for an 80386SX system.
5. MAD1 and MAD8 are respectively the system select bits SS1 and SS2. These bits determine the definition of ABC ports E0, E1 and 103, as described in the section on ABC ports and registers.
6. MAD5, MAD6 and MAD7 are respectively the DRAM performance select bits C2, C1 and C0. The effect of these bits is described in the DRAM control section.

82309 Address Bus Controller Pin Definitions

Signal Name	Pin Number	I/O	Description
A<00:23>	42–50, 53–67	I	Micro Channel Address 0 to 23
HIMEM#	84	I	Micro Channel Address 24 to 31 = FF (Active Low). Used in decoding the top-of-memory mapping of the BIOS EPROMs.
MADE24	85	I	Micro Channel Address 24 to 31 = 00 (Active High)
ROMEN#	32	O	EPROM Decode. In systems that support shadow RAM, if ROM is enabled (bit 1 in port E1), accesses to ROM space actually generate both ROMEN# and RAMEN#. In this mode, reads are from ROM, and writes are to RAM.
RAMEN#	33	O	DRAM Decode
IOEN#	34	O	Motherboard I/O devices decode (Active Low). Decode also includes memory decode of video RAM.
LCSCS#	35	O	Chip Select for the LCS (82306) Chip (Active Low). Decodes address range 0–3FFH when CPU master, or 100–3FFFH when CPU is not master.
VMSEL#	36	O	VGA Memory Space Selected (Active Low) (000A0000–000BFFFF)
S0#	89	I	Micro Channel S0# Signal
S1#	90	I	Micro Channel S1# Signal
PM/IO#	41	I	Microprocessor M/IO# Signal
PW/R#	91	I	Microprocessor W/R# Signal
PD/C#	92	I	Microprocessor D/C# Signal
PADS#	93	I	Microprocessor ADS# Signal
SCLK	94	I	Microprocessor CLK2
HLDA	95	I	HLDA Signal from the Processor
PRDYI#	96	I	READY# Signal from the Processor
M/IO#	86	I	Micro Channel M/IO# Signal
CMD#	87	I	Micro Channel CMD Signal

82309 Address Bus Controller Pin Definitions (Continued)

Signal Name	Pin Number	I/O	Description
WE	73	O	DRAM Write Enable Signal (Active High)
RAS<0:3>	76–79	O	DRAM RAS Strobes (Active High)
CAS# <0:3>	80–83	O	DRAM CAS Strobe Enables (Active Low)
MAD<00:10>	17–23, 27–30	B	DRAM Muxed Address bus. These signals are sampled at reset to determine ABC configuration.
DSTB	70	O	Output to Delay Line. A pulse put into the delay line controls page miss timing.
DLY<1:4>	12–14, 71	I	Inputs from the Delay Line. DLY1 controls CHRDY timing in non-CPU cycles. DLY2 controls RAS active to CAS active timing. DLY3 controls RAS precharge, and DLY4 controls row-to-column address multiplex.
CHRDY	72	O	DRAM Ready Signal (Active High)
REFRESH#	15	I	Refresh Operation in Progress (Active Low)
FCES#	31	O	Request to BC to terminate CPU accesses to system board memory.
TINCLK	40	I	14.3 MHz Clock for Refresh Timer
RFRQ#	37	O	Refresh Request (Active Low)
TMRCLK	39	O	14.3 MHz Clock divided by 12 to get 1.19 MHz.
FRQ#	68	O	Asynchronous Cache Flush Request. Activated in I/O writes to ports E0, E1, or 100–107 (POS Address Space).
EXEN#	69	O	Read/Write Strobe for Ports 00E0–00E7 (Active Low)
ERS	8	I	Sampling Strobe for Ports 00E2–00E7
PD	9	I	Select Signal for POS Register 10X
ARB/GNT#	10	I	Micro Channel ARB/GNT# Signal
D<0:7>	97–100, 3–6	I/O	Data Bus
PCE#	7	O	Enable Parity Checking (MER<0>)
RESET	11	I	Synchronous reset input. RESET falling edge used to synchronize ABC internal clock to CPU phase.
NC	1, 25, 51, 52, 75		No Connect
V_{DD}	26, 28		Power
V_{SS}	2, 16, 24, 38, 74		Ground



290188-11

NOTE:
NC = No Connect

82309 PARAMETRICS

ABSOLUTE MAXIMUM RATINGS*

Case Temperature under Bias .... −40°C to +85°C
Storage Temperature ..... −65°C to +150°C
Voltage to Any Pin with
Respect to Ground ..... −0.3V to (V_{CC}+0.3)V
DC Supply Voltage (V_{CC}) ..... −0.3V to +7.0V
DC Input Current ..... ±10 mA

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS

T_C = 0°C to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Min	Max	Units	Conditions
V_{IL}	Input Low Voltage		0.8	V	
V_{IH}	Input High Voltage	2.0		V	
V_{IL}	Input Low Voltage		0.8	V	SCLK
V_{IH}	Input High Voltage	V_{CC} − 0.8		V	SCLK
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 4 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 4 mA
I_{CC}	Power Supply Current		180	mA	No DC Loads
I_{LI}	Input Leakage Current		±10	μA	$V_{SS} < V_{IN} < V_{CC}$
I_{OZ}	Tri-State Output Leakage Current		±10	μA	$V_{SS} < V_{OUT} < V_{CC}$

82309 ADDRESS BUS CONTROLLER A.C. SPECS

T_C = 0°C to +70°C, V_{CC} = 5V ± 10%

Symbol	Parameter	Kit 16 MHz		Kit 20 MHz		Kit 25 MHz		CL(PF)	Notes
		Min	Max	Min	Max	Min	Max		
T1	SCLK PERIOD	31.25		25		20			
T2A	SCLK HIGH/LOW TIME (50%)	12		10		8			
T2B	SCLK HIGH/LOW TIME (90%)	8		6.5		6			
T3	RESET SETUP	10		10		10			
T4	RESET HOLD	4		4		4			
T5A	STATUS SETUP TO SCLK	11		11		8			1
T5B	CMD# SETUP TO SCLK	11		11		8			1
T6	PADS#,PW/R#,PD/C#,PM/IO# SETUP	25		22		13			
T7	PADS#,PW/R#,PD/C#,PM/IO# HOLD	4		4		4			
T8	ADDRESS,M/IO#,MADE24,REFRESH# SETUP	10		10		10			
T9	ADDRESS,M/IO#,MADE24,REFRESH# HOLD	10		10		10			
T10	ADDRESS, M/IO# SETUP	40		50		36			3
T11	ADDRESS, M/10# HOLD	30		30		30			3, 12
T12	MADE24 SETUP	32		40		28			3
T13	MADE24 HOLD	30		30		30			3, 12
T14	M/IO#,ARB/GNT# SETUP TO ERS	20		20		20			
T15	M/IO#,ARB/GNT# HOLD FROM ERS	10		10		10			
T16	PRDYI# SETUP	18		18		15			
T17	PRDYI# HOLD	3		3		3			
T18A	ROMEN#,RAMEN#,IOEN#,VMSEL# DLY FRM MADE24 HIMEM#	2	30	2	30	2	30	75	
T18B	ROMEN#,RAMEN#,IOEN#,VMSEL# DLY FRM ADDR	2	38	2	38	2	38	75	15
T18C	ROMEN#,RAMEN#,IOEN#,VMSEL# DLY FRM A20	2	35	2	35	2	35	75	15
T19	LCSCS# DELAY	2	45	2	45	2	45	75	
T20A	FCES# DELAY FROM SCLK	3	45	3	35	3	30	25	5
T20B	FCES# DLY FRM ADDR, M/IO#, MADE24		50					25	2, 5
T21	PD SETUP TO CMD# ↑	100		100		100			
T22	WRITE DATA SETUP	30		30		30			
T23	WRITE DATA HOLD	5		5		5			
T24	READ DATA VALID DELAY		200		200		200	75	
T24A	CMD# ↓ TO READ DATA LOW-Z	25		25		25		75	
T25	READ DATA FLOAT DELAY	2	35	2	35	2	35	75	
T26	EXEN# DELAY (INACTIVE)	2	50	2	50	2	50	50	
T26A	EXEN# DELAY (ACTIVE)	25	150	25	150	25	150	50	
T27A	CHRDY DELAY (FROM ADDR)	2	50	2	50	2	50	50	4, 5, 7
T27B	CHRDY DLY FROM STATUS OR CMD#	0	33	0	33	0	33	50	4, 5, 7
T29	TMRCLK HIGH/LOW TIME	300		300		300		50	
T30	RFRQ# PULSE WIDTH	300		300		300		50	
T31	TINCLK HIGH/LOW TIME	21		21		21			
T32B	CMD# ↑-RAS ↓ (REFRESH CYCLE ONLY)	0	55	0	55	0	55	75	10
T32C	CMD# ↑-CAS# ↑	0	38	0	38	0	38	75	
T32E	SCLK-DSTB ↓	4	27	4	27	4	27	50	8
T32F	DLY3 ↑-DSTB ↑	0	50	0	50	0	50	50	
T32G	DLY3 ↑-RAS ↑	4	26	4	26	4	26	75	
T32I	SCLK-RAS ↓	8	40	8	40	8	40	75	8
T33	DLY1 ↓ TO CHRDY ↑	5	30	5	30	5	30	50	
T34	DLY2 ↓ TO CAS# ↓	3	27	3	27	3	27	75	

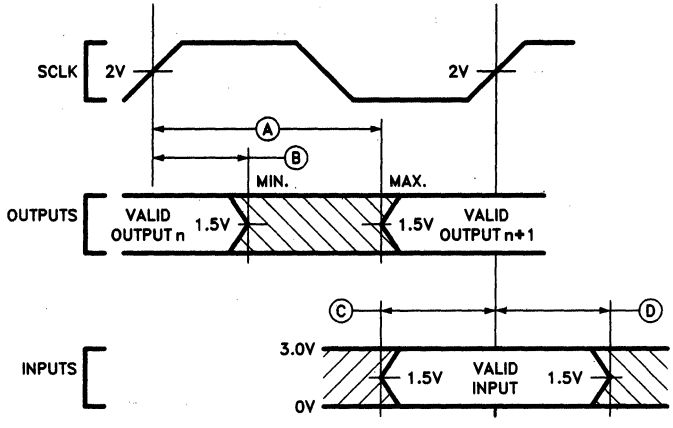
82309 ADDRESS BUS CONTROLLER A.C. SPECS (Continued)

Symbol	Parameter	Kit 16 MHz		Kit 20 MHz		Kit 25 MHz		CL(PF)	Notes
		Min	Max	Min	Max	Min	Max		
T35	CMD# ↓ TO CAS# ↓ (WRITE CYCLES ONLY)	25	115	25	115	25	115	75	6
T39A	WE DLY FROM STATUS	2	30	2	30	2	30	75	14
T39B	WE DLY FROM CMD# ↑	2	30	2	30	2	30	75	14
T41A	CAS# ↓ DELAY FROM SCLK (READS)	2	30	2	30	2	30	75	
T41B	CAS# ↑ DELAY FROM SCLK	5	34	5	34	5	34	75	9
T41C	CAS# ↓ DELAY FROM SCLK (WRITES)	2	38	2	38	2	38	75	
T43A	ADDR TO MAD DELAY (COLUMN ADDR)		45		45		40	75	13
T43B	SCLK TO MAD DELAY (COLUMN ADDR)		36		36		31	75	13
T43C	CMD# TO MAD DELAY (COLUMN ADDR)		38		38		38	75	13
T43D	SCLK TO MAD DELAY (ROW ADDR)		50		50		50	75	13
T44	WE DELAY FROM SCLK	2	42	2	42	2	42	75	
T45	DLY4 ↓ TO MAD	6	32	6	32	6	32	75	
T46	MAX PAGE MODE RAS ACTIVE	15 μs		15 μs		15 μs			10, 11

NOTES:

1. Status and CMD# are asychronous inputs. T5 simply guarantees that they are recognized at a particular clock edge.
2. FCES# is speced from address only in 0WS pipelined/1WS non-pipelined memory cycles, which are only supported at 16 MHz.
3. Address, M/IO# and MADE24 setup and hold times are speced relative to the Phase 2 SCLK edge only in 0WS pipelined/1WS non-pipelined memory cycles, which are only supported at 16 MHz. (This Phase 2 edge is in the middle of the first T2 state, or the middle of the T1P state.)
4. The 82309 de-activates CHRDY for motherboard I/O and VGA Memory cycles (as decoded by IOEN#), and then re-activates it when CMD# is activated. The 82309 also de-activates CHRDY for non-CPU (DMA or channel master) accesses to motherboard DRAM that are decoded as page misses. CHRDY is then re-activated according to the appropriate external DRAM control delay line tap (DLY1), or else when CMD# is activated, whichever comes later.
5. FCES# is used to terminate CPU accesses to motherboard DRAM, as these cycles are not broadcast on the Micro Channel. CHRDY is used to terminate DMA and channel master accesses to motherboard DRAM, which are broadcast.
6. The large value for T35 (Min) guarantees that data being written into motherboard DRAM by a channel master or DMA controller has adequate time to propogate through the data buffers between the channel or DMA and memory. T35 (Min) is guaranteed on any non-CPU write, both page hit and page miss. (The Micro Channel specs 0NS of data setup to CMD# active.) T35 (Max) applies only when CMD# ↓-to-CAS# ↓ is indeed the limiting spec; specifically, when neither T34 (Max) nor T41A (Max) limits CAS# activation.
7. The 82309 guarantees that any time it de-asserts CHRDY, it will not re-assert it until after CMD# is activated.
8. These specs are referenced with respect to the causal SCLK edge, which differs in different frequency systems. At 16 MHz, the appropriate edge is one clock phase after the edge that recognizes status in non-CPU cycles, or one phase after the edge that samples PADS# active in CPU cycles. At 20 MHz, the appropriate edge is two clock phases after these events. The 82309 distinguishes 16 MHz from 20 MHz via the memory performance configuration inputs C0, C1 and C2. 16 MHz is assumed anytime these inputs indicate a zero wait state pipelined read page hit. (C0, C1, C2 = 000,001,010).
9. This spec insures a minimum CAS# high time of 25 ns at 16 MHz. (16 MHz is the worst case since the CAS# inactive and CAS# active SCLK edges are only one phase apart. At 20 MHz, these edges are always at least two phases apart.)
10. Refresh cycles are RAS only, and are forced to be page misses. Thus, the refresh interval (typically 15 μs) defines the required page mode RAS active time. RAS is de-activated at the end of a refresh cycle since typical page mode DRAMs spec a maximum RAS active time less than 15 μs for refresh cycles. (Note that the first access to any bank following a refresh cycle is also a forced page miss.)
11. Functional spec only . . . Not tested. Max page mode RAS active is governed by refresh interval.
12. T11 and T13 are speced relative to the clock edge that samples the page hit/miss outcome. This edge also is used by the 82309 to internally latch the channel address. At 20 MHz, T11 & T13 are speced relative to the end of the first T2 state or the end of T1P. Note, however, the large T11 & T13 values in the spec, which seem like they would be difficult to meet going directly into a T2P, where the CPU puts out a new address. This is not a problem, however, since the minimum memory cycles at 20 MHz run T1•T2•T2P•T2P•T1P•T2P•T2P, and the 82308 does not open the transparent channel address latch until the beginning of the last T2P, i.e., the 82309 effectively only sees the last T2P in terms of pipelined addressing regardless of how many T2P states the CPU actually executes.
13. T43A, T43B, and T43C all refer to column address, as the 82309 assumes a page hit as the default case until proven otherwise. In case of a page miss, the row address is muxed onto the MAD lines from the same clock edge that de-activates RAS and fires a pulse (DSTB) into the delay line. The column address is then muxed onto the MAD lines by delay tap DLY4.
14. In Non-CPU cycles, WE (active high from the 82309) is simply an inverted version of channel S0#, which indicates a write cycle when low. This signal is internally latched (transparent latch) by the leading edge of CMD#, and then released by the trailing edge of CMD#, hence the need for T39B.
15. A20 typically has more logic in its path than the other address bits, hence the tighter spec. T18B applies to all bits except A20.

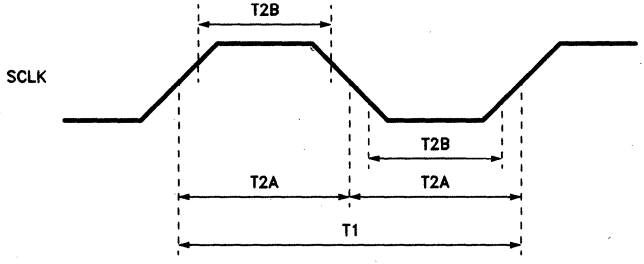
DRIVE LEVELS AND MEASUREMENT POINTS FOR A.C. SPECIFICATIONS



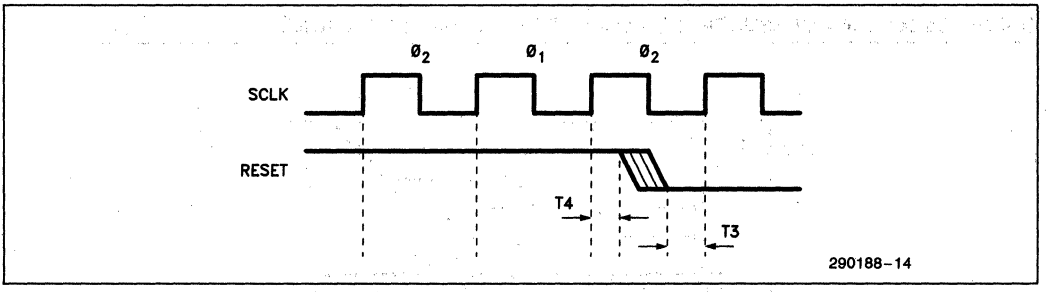
290188-12

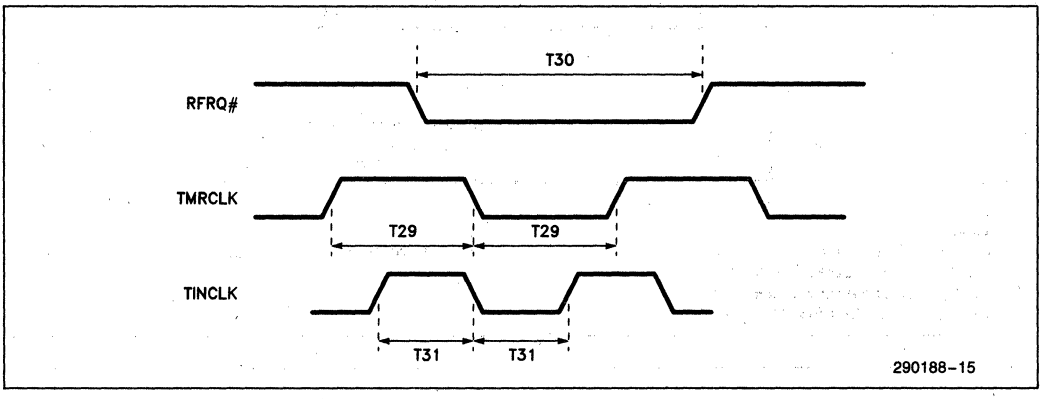
LEGEND:
A. Maximum Output Delay Specification.
B. Minimum Output Delay Specification.
C. Minimum Input Setup Specification.
D. Minimum Input Hold Specification.

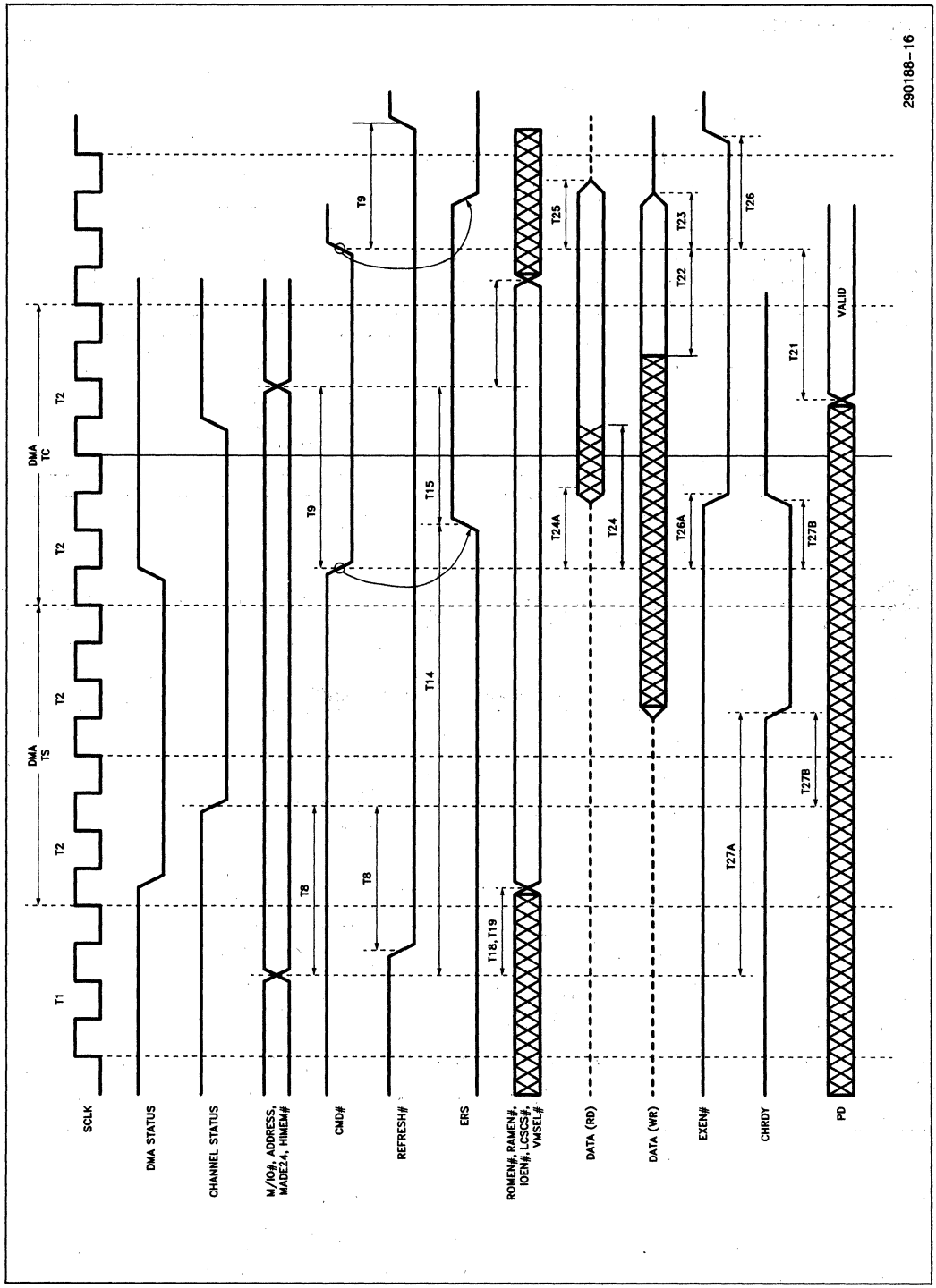
Input waveforms have tr ≤ 2.0 ns from 0.8V to 2.0V.



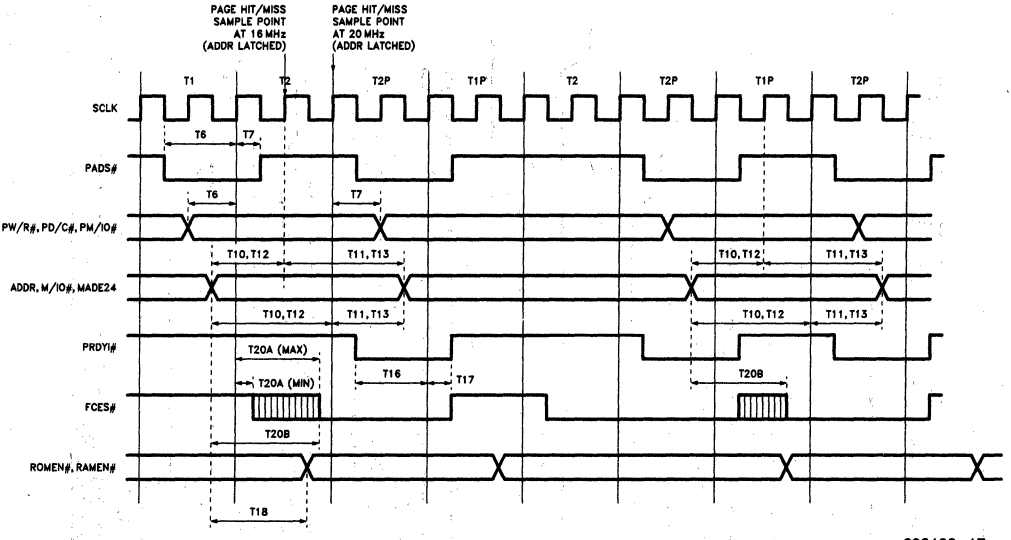
290188-13


Ø2
Ø1
Ø2
SCLK
RESET
T4
T3
290188-14


T30
RFRQ#
TMRCLK
T29
T29
TINCLK
T31
T31
290188-15

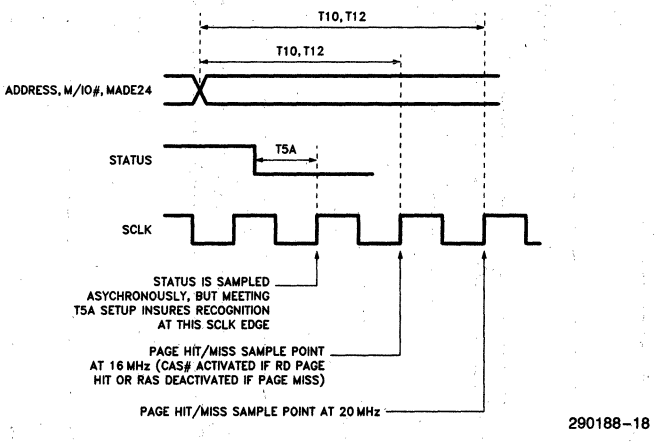

290188-16
SCLK
DMA STATUS
CHANNEL STATUS
M/IO#, ADDRESS, MADE24, HIMEM#
CMD#
REFRESH#
ERS
ROMEN#, RAMEN#, IOEN#, LCSCS#, VMSEL#
DATA (RD)
DATA (WR)
EXEN#
CHRDY
PD
VALID
T1
T2
DMA TS
DMA TC
T8
T9
T14
T15
T18, T19
T21
T22
T23
T24
T24A
T25
T26
T26A
T27A
T27B

82309/80386 INTERFACE TIMINGS

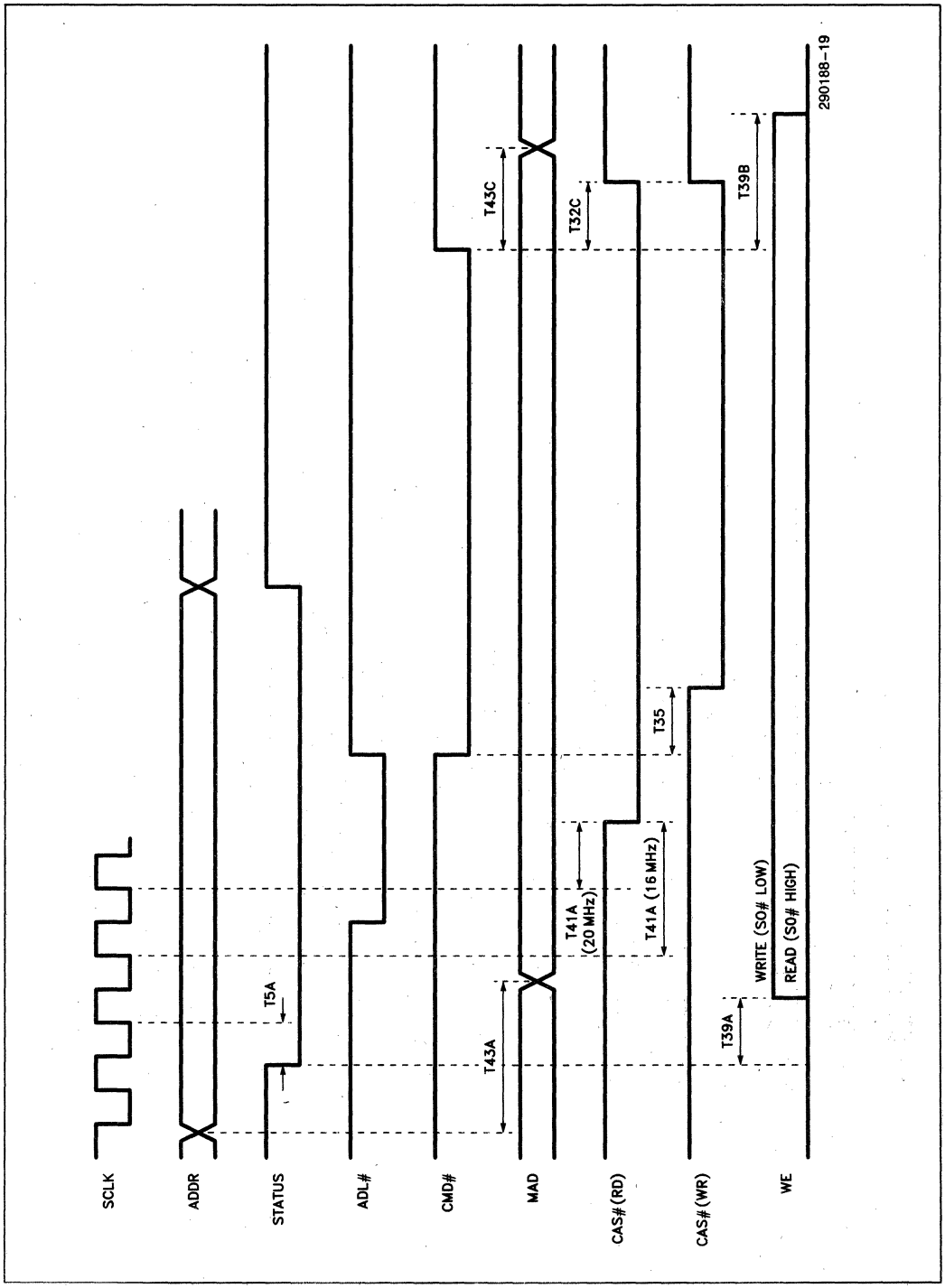


NOTE:
Once the ABC fires FCES#, it remains active until the ABC detects and end of cycle via its PRDYI# input. This allows the BC or other external devices to extend DRAM cycles if desired.

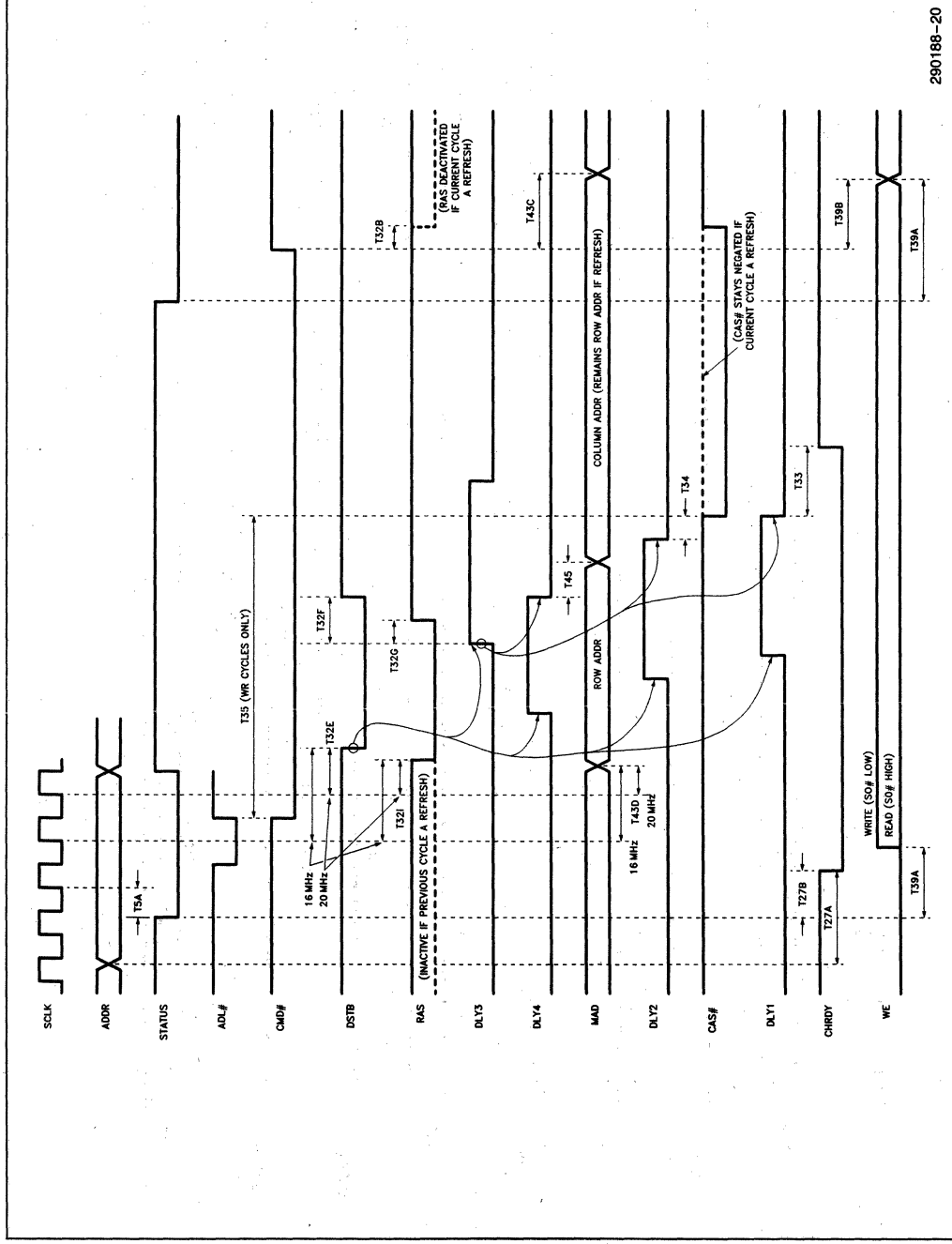
ADDRESS, M/IO#, MADE 24 SETUP FOR DMA/Micro Channel MASTER



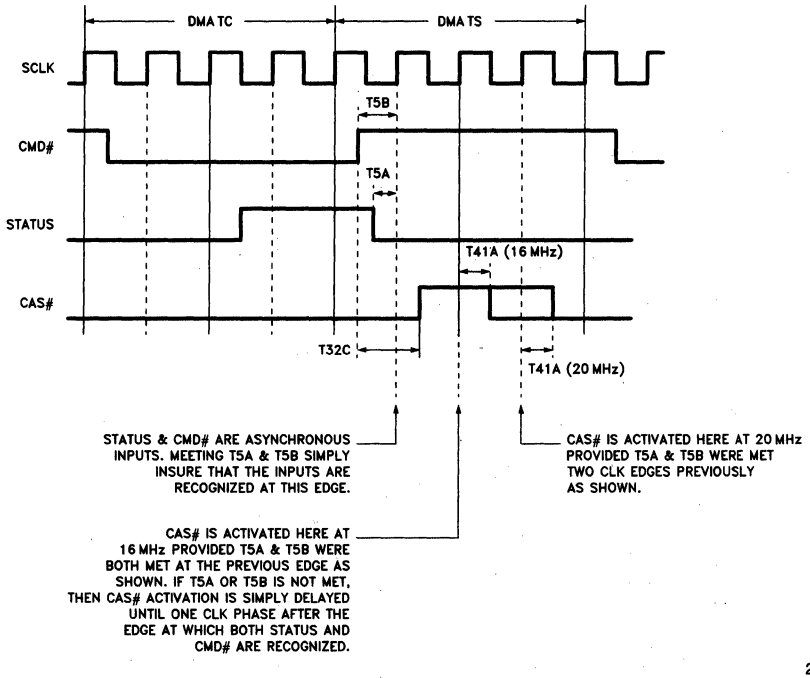
DMA/CHANNEL MASTER ACCESSES (PAGE HIT)



DMA/CHANNEL MASTER ACCESSES TO DRAM (PAGE MISS)



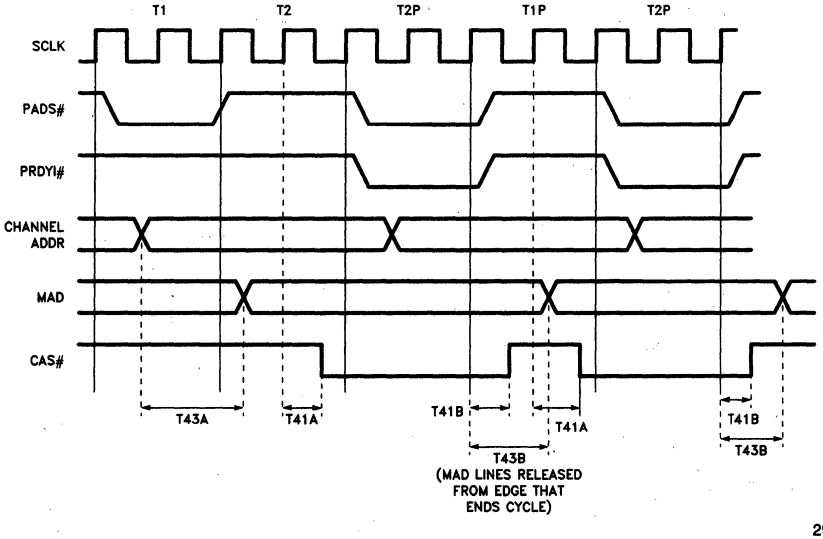
DMA MASTER ... BACK-TO-BACK RD PAGE HITS



290188-21

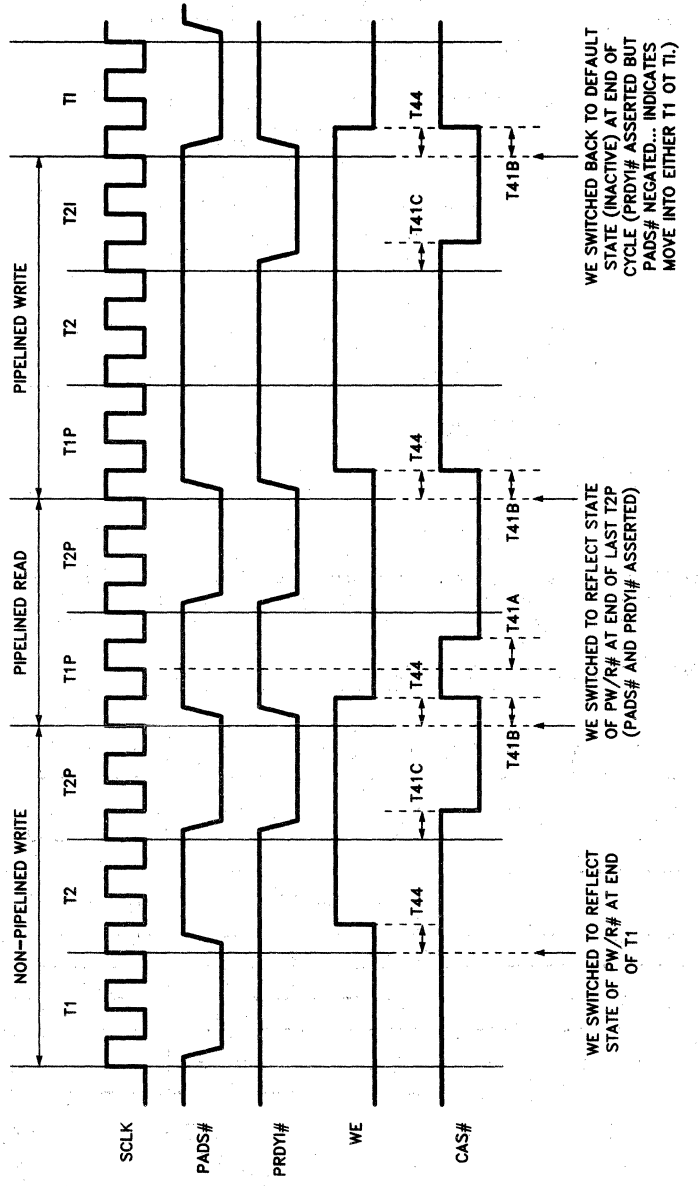
The requirement that T5B be met before CAS# is allowed (one or two phases later) insures adequate CAS# high time in back-to-back cycles. (One SCLK phase at 16 MHz; Two SCLK phases at 20 MHz.)

0 WAIT STATE (PIPELINED) READ PAGE HITS



290188-22

WE TIMING IN CPU CYCLES

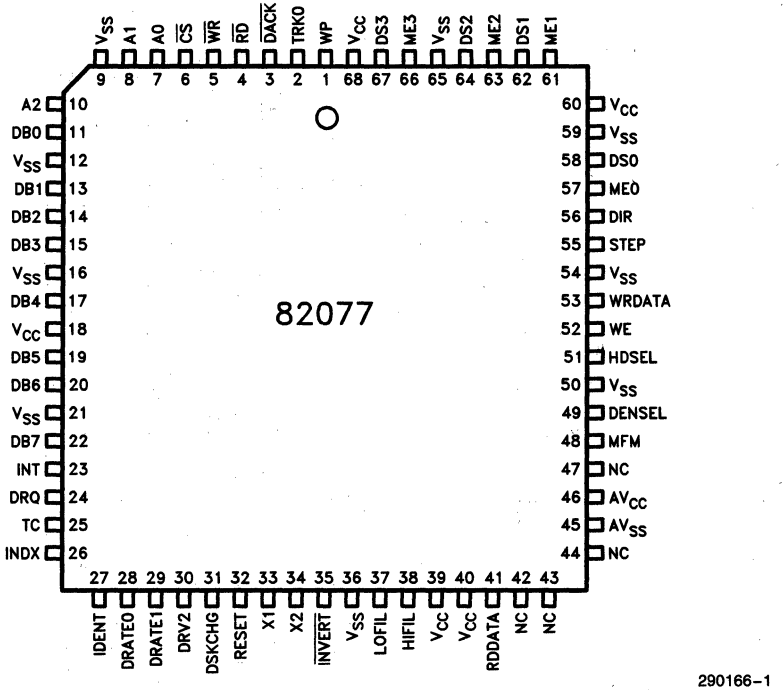


82077
CHMOS SINGLE-CHIP FLOPPY DISK CONTROLLER

- **Single-Chip Floppy Disk Solution**
  - **— 100% PC-AT Hardware Compatible**
  - **— 100% PS/2™ Hardware Compatible**
  - **— Integrated Drive and Data Bus Buffers**
- **Integrated Analog Data Separator**
  - **— 250 Kbits/sec**
  - **— 300 Kbits/sec**
  - **— 500 Kbits/sec**
  - **— 1 Mbits/sec (82077-1 only)**
- **High Speed Processor Interface**
- **Vertical Recording Support**
- **12 mA Data Bus Drivers, 40 mA Disk Drivers**
- **Four Fully Decoded Drive Select and Motor Signals**
- **Programmable Write Precompensation Delays**
- **Addresses 256 Tracks Directly, Supports Unlimited Tracks**
- **16 Byte FIFO**
- **68-Pin PLCC**

The 82077 floppy disk controller has completely integrated all of the logic required for floppy disk control. The 82077, a 24 MHz crystal, a resistor package and a device chip select implements a PC-AT or PS/2™ solution. All programmable options default to compatible values. The dual PLL data separator has better performance than most board level/discrete PLL implementations. The FIFO allows better system performance in multi-master systems (e.g. PS/2™).

The 82077 is fabricated with Intel's CHMOS III technology and is available in a 68-lead PLCC (plastic) package.



290166-1

Figure 1. 82077 Pinout

"For a complete data sheet, please refer to the Floppy Disk Controller section of the Intel Microprocessor and Peripherials Handbook (Volume II. Peripherials)".

PS/2™ is a trademark of IBM.

October 1988
Order Number: 290166-001

82706
INTEL VIDEO GRAPHICS ARRAY

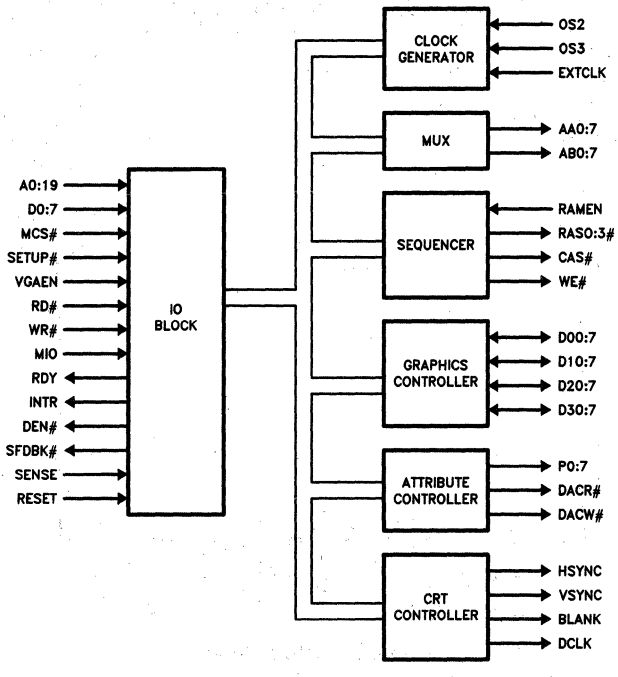
- **Single Chip Video Graphics Array for IBM PC/XT/AT*, Personal System/2* and Compatible Systems**
- **100% Gate, Register, and BIOS Level Compatibility with IBM VGA**
- **EGA/CGA/MDA BIOS Compatibility**
- **Inmos IMSG 171 Palette/DAC Interface**
- **4 mA Drive Capability on Output Pins**
- **Implemented in High Speed CHMOS III Technology**
- **Available in 132-Pin Plastic Quad Flat Pack Package**
  (See Packaging Spec. Order #231369)

The 82706 is the Intel VGA compatible display controller. It is 100% register compatible with all IBM VGA modes and provides software compatibility at the BIOS level with EGA, CGA, and MDA. All video monitors designed for IBM PS/2* systems are supported by the Intel VGA controller. The 82706 provides an 8-bit video data path to any Inmos IMSG 171 compatible palette/DAC. It also acts as a CRT controller and video memory controller. The 82706 supports 256 Kbytes of video memory.

The 82706 is designed for compatibility with the Intel 80286 and 80386 microprocessors and other microprocessors.

Implemented in low power CHMOS technology, the 82706 VGA Controller is packaged in a fine pitch (25 mil) surface mount gull wing package. It can be enabled or disabled under software control via the 82306 Peripheral Bus Controller.

*IBM PC, XT, AT, Personal System/2, PS/2, and MicroChannel are trademarks of International Business Machines.

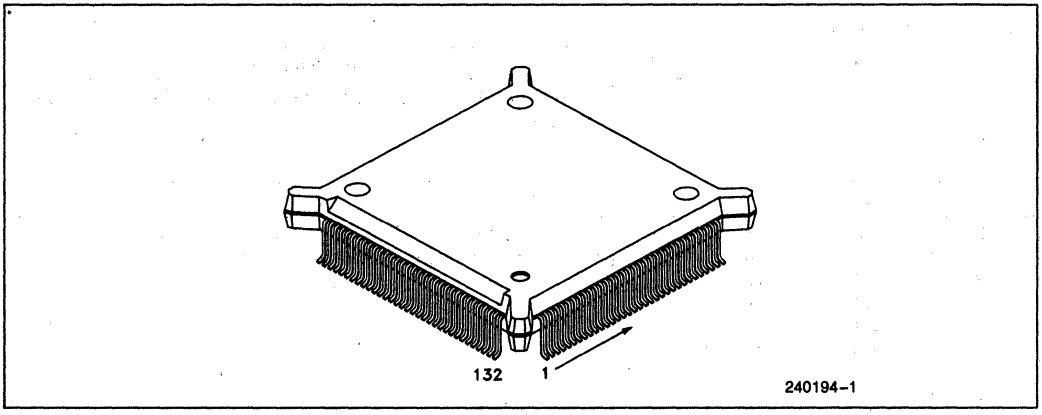


240194-2

Figure 1. Block Diagram

November 1988
Order Number: 240194-002

PLASTIC QUAD FLAT PACK (PQFP)



Pinout (Top View)

82706 PIN DESCRIPTION

Left side of package (top view):

Number	Name	Active	I/O	Description
1	AB4	HI	O	DRAM address bus, planes 2 and 3 (continued from top side of package).
2	AB5	HI	O	
3	AB6	HI	O	
4	AB7	HI	O	
5	V_{DD}			
6	A0	HI	I	System address bus, from current MicroChannel* master. Used to select display buffer words, VGA registers, or video DAC registers.
7	A1	HI	I	
8	A2	HI	I	
9	A3	HI	I	
10	A4	HI	I	
11	A5	HI	I	
12	A6	HI	I	
13	A7	HI	I	
14	A8	HI	I	
15	A9	HI	I	
16	A10	HI	I	
17	A11	HI	I	
18	A12	HI	I	
19	A13	HI	I	
20	A14	HI	I	
21	A15	HI	I	
22	A16	HI	I	
23	A17	HI	I	
24	A18	HI	I	
25	A19	HI	I	
26	$\overline{MCS}$	LO	I	Display buffer (memory) select, generated from A24:20. These inputs do not qualify I/O cycles.
27	RESET	HI	I	Device reset. Tri-States all VGA pins when active.
28	RAMEN	HI	I	RAM Enable. When inactive, tri-states all VGA DRAM interface pins to allow board test of DRAMs or to allow another device to share control of the DRAM.
29	SENSE	HI	I	Switch sense input. The state of this pin can be read from Input Status 0 register, bit 4, and indicates if a monochrome or color monitor is attached to the Display Connector. BIOS requires this information to set the video mode correctly.
30	$\overline{SETUP}$	LO	I	VGA Setup. Used during Programmable Option Select (POS) operation. Analogous to MicroChannel-CD SETUP signals for the other adapter slots. After RESET, the 82706 requires the setup pin to be pulled low. An I/O write to port address XX2h must write a "1" to data bit 0 while setup is low. Then setup must be pulled high. Once this is complete, the 82706 can respond to CPU accesses.
31	VGAEN	HI	I	VGA Enable. Allows VGA to respond to memory or I/O cycles when active.
32	FCO		O	
33	FCI		O	

82706 PIN DESCRIPTION (Continued)

Bottom side of package (top view):

Number	Name	Active	I/O	Description
34	V_{SS}			
35	$\overline{RAS0}$	LO	O	Row Address Strobe for plane 0.
36	$\overline{RAS1}$	LO	O	Row Address Strobe for plane 1.
37	$\overline{RAS2}$	LO	O	Row Address Strobe for plane 2.
38	$\overline{RAS3}$	LO	O	Row Address Strobe for plane 3.
39	$\overline{WE}$	LO	O	RAM Write Enable for all planes.
40	V_{DD}			
41	$\overline{CAS}$	LO	O	Column Address Strobe for all planes.
42	HSYNC		O	Horizontal and Vertical Sync to Display Connector.
43	VSYNC		O	These are programmable to be active high or low.
44	BLANK	LO	O	Video Blank to Video DAC and Display Connector.
45	V_{SS}			
46	$\overline{RD}$	LO	I	Read Strobe, active for memory or I/O cycles.
47	$\overline{WR}$	LO	I	Write Strobe, active for memory or I/O cycles.
48	MIO	HI	I	Memory/$\overline{IO}$; high for memory cycles, low for I/O.
49	O1			Pull up to VDD using 10K resistor.
50	OS2		I	Video clocks. OS2 (28.3 MHz) is used for modes with 720
51	OS3		I	pixel horizontal resolution. OS3 (25.17 MHz) is used for
52	EXTCLK		I	modes with 320 or 640 pixel horizontal resolution. EXTCLK is a MicroChannel video extension signal, which allows using a user-defined clock.
53	V_{SS}			
54	DCLK		O	Pixel clock to Video DAC.
55	$\overline{DACR}$	LO	O	Video DAC read and write strobes.
56	$\overline{DACW}$	LO	O	
57	V_{DD}			
58	P7	HI	O	Pixel output bus to Video DAC.
59	P6	HI	O	
60	P5	HI	O	
61	P4	HI	O	
62	P3	HI	O	
63	P2	HI	O	
64	P1	HI	O	
65	P0	HI	O	
66	V_{SS}			

82706 PIN DESCRIPTION (Continued)

Right side of package (top view):

Number	Name	Active	I/O	Description
67	V_{DD}			
68	D7	HI	I/O	System data bus. The VGA may be accessed by any MicroChannel bus master.
69	D6	HI	I/O	
70	D5	HI	I/O	
71	D4	HI	I/O	
72	D3	HI	I/O	
73	D2	HI	I/O	
74	D1	HI	I/O	
75	D0	HI	I/O	
76	V_{SS}			
77	RDY	HI	O	Bus ready signal.
78	INTR	LO	O	Interrupt request. When enabled, INTR is activated during vertical retrace.
79	$\overline{DEN}$	LO	O	Data Enable to the VGA's data bus transceiver. The direction of the transceiver is controlled by the bus controller array.
80	$\overline{SFDBK}$	LO	O	VGA Selected Feedback. Active when VGA is address selected for a memory or I/O cycle, as an acknowledgement of its presence at the address specified. Used during diagnostics.
81	V_{DD}			
82	D00	HI	I/O	DRAM data bus, plane 0.
83	D01	HI	I/O	
84	D02	HI	I/O	
85	D03	HI	I/O	
86	D04	HI	I/O	
87	D05	HI	I/O	
88	D06	HI	I/O	
89	D07	HI	I/O	
90	V_{SS}			
91	D10	HI	I/O	DRAM data bus, plane 1.
92	D11	HI	I/O	
93	D12	HI	I/O	
94	D13	HI	I/O	
95	D14	HI	I/O	
96	D15	HI	I/O	
97	D16	HI	I/O	
98	D17	HI	I/O	
99	V_{DD}			

82706 PIN DESCRIPTION (Continued)

Top side of package (top view):

Number	Name	Active	I/O	Description
100	V_{SS}			
101	D20	HI	I/O	DRAM data bus, plane 2.
102	D21	HI	I/O	
103	D22	HI	I/O	
104	D23	HI	I/O	
105	D24	HI	I/O	
106	D25	HI	I/O	
107	D26	HI	I/O	
108	D27	HI	I/O	
109	V_{DD}			
110	D30	HI	I/O	DRAM data bus, plane 3.
111	D31	HI	I/O	
112	D32	HI	I/O	
113	D33	HI	I/O	
114	D34	HI	I/O	
115	D35	HI	I/O	
116	D36	HI	I/O	
117	D37	HI	I/O	
118	V_{SS}			
119	V_{DD}			
120	AA0	HI	O	DRAM address bus, planes 0 and 1.
121	AA1	HI	O	
122	AA2	HI	O	
123	AA3	HI	O	
124	AA4	HI	O	
125	AA5	HI	O	
126	AA6	HI	O	
127	AA7	HI	O	
128	V_{SS}			
129	AB0	HI	O	DRAM address bus, planes 2 and 3.
130	AB1	HI	O	
131	AB2	HI	O	
132	AB3	HI	O	

FUNCTIONAL DESCRIPTION

The 82706 interfaces the host processor and video memory, and provides palette DAC support and display of video data. All accesses between the host and video memory go through the 82706. These accesses are arbitrated with display refresh requirements to allow the CPU to read or write video memory at any time without having to wait for display retrace.

Video memory contains 256 Kbytes organized as four 64K x 8 maps. The starting address of the video memory in the host address space is programmable, providing three different start addresses. Display data from video memory is formatted into an 8-bit value clocked out on pins P0–P7, which may drive a DAC or go directly to a TTL monitor interface.

CRT Controller

In addition to generating horizontal and vertical sync timings, the CRT controller (CRTC) generates addressing for DRAM refresh and timings to support the cursor and underline capabilities.

Graphics Controller

The graphics controller provides the interface between video memory and both the host processor and the attribute controller. In alphanumeric (A/N) modes, display data is latched from video memory and sent in parallel to the attribute controller. In All Points Addressable (APA) modes, latched display data is serialized before being sent to the attribute controller.

Two read modes and four write modes are supported. Processor reads to video memory cause one byte from each of the four memory maps to be latched. Read mode 0 causes the host processor to read this latched data from a selected map, allowing access to each bit plane separately. Read mode 1 causes the pixel values in each selected map to be compared to a reference value stored in the Color Compare register. Each bit of the byte read contains a 1 when the latched pixel value matches the reference value.

Memory maps may be masked for write operations, allowing the host processor to update any or all memory maps with a single 8-bit access. In write mode 0, logical operations may be performed between pixel data latched by the previous read operation and either write data, which may be rotated, or data stored in the Set/Reset register. Logical operations supported are AND, OR, XOR, or write data unmodified. Write mode 1 simply copies latched data to the memory maps. Write Modes 2 and 3 are similar to write mode 0. In write mode 2, each enabled memory map is updated with 8 bits of the value in the corresponding bit position of the write data. Write mode 3 writes each enabled map with 8 bits of the value in the Set/Reset register. The bit mask value is derived by ANDing write data with the value in the bit mask register.

Attribute Controller

Display data in APA modes, and character generator and attribute data in A/N modes is sent to the attribute controller from the graphics controller. The attribute controller handles cursor insertion, panning, underlining, and blinking. The 8-bit per pixel output value is available on pins P0–P7.

Sequencer

The sequencer generates DRAM memory timings and arbitrates all accesses to video memory. It inserts CPU memory cycles at appropriate times between display memory fetches. The sequencer contains map mask registers which can prevent maps from being updated by memory accesses.

Preliminary product information describes products for which full characterization data is not yet available. Intel believes this information is accurate and reliable. However, it is subject to change without notice.

Table 1. Modes of Operation

GRAPHICS MODES		
Mode	**Resolution**	**Colors**
4, 5	320 x 200	4 out of 256K
6	640 x 200	2 out of 256K
D	320 x 200	16 out of 256K
E	640 x 200	16 out of 256K
F	640 x 350	Monochrome
10	640 x 350	16 out of 256K
11	640 x 480	2 out of 256K
12	640 x 480	16 out of 256K
13	320 x 200	256 out of 256K

ALPHA (TEXT) MODES					
Mode	**Rows**	**Columns**	**Char. Box**	**Resolution**	**Colors**
0, 1	25	40	9 x 16	320 x 200 (CGA)	16 out of 256K
				320 x 350 (EGA)	16 out of 256K
				360 x 400 (VGA)	16 out of 256K
2, 3	25	80	9 x 16	640 x 200 (CGA)	16 out of 256K
				640 x 350 (EGA)	16 out of 256K
				720 x 400 (VGA)	16 out of 256K
7	25	80	9 x 16	720 x 350 (EGA)	Monochrome
				720 x 400 (VGA)	Monochrome

REGISTER SET

Register Name	R/W	Index	Read Port	Write Port
GENERAL REGISTERS				
Miscellaneous Output	W			03C2
	R		03CC	
Input Status 0	R		03C2	
Input Status 1	R		03?A	
Feature Control	W			03?A
	R		03CA	
GRAPHICS CONTROLLER				
Graphics Address	R/W		03CE	03CE
Set/Reset	R/W	00	03CF	03CF
Enable Set/Reset	R/W	01	03CF	03CF
Color Compare	R/W	02	03CF	03CF
Data Rotate	R/W	03	03CF	03CF
Read Map Select	R/W	04	03CF	03CF
Graphics Mode	R/W	05	03CF	03CF
Miscellaneous	R/W	06	03CF	03CF
Color Don't Care	R/W	07	03CF	03CF
Bit Mask	R/W	08	03CF	03CF
SEQUENCER				
Sequencer Address	R/W		03C4	03C4
Reset	R/W	00	03C5	03C5
Clocking Mode	R/W	01	03C5	03C5
Map Mask	R/W	02	03C5	03C5
Character Map Select	R/W	03	03C5	03C5
Memory Mode	R/W	04	03C5	03C5
ATTRIBUTE CONTROLLER				
Address	R/W		03C0	03C0
Palette Registers	R/W	00–0F	03C1	03C0
Attribute Mode Control	R/W	10	03C1	03C0
Overscan Color	R/W	11	03C1	03C0
Color Plane Enble	R/W	12	03C1	03C0
Horizontal PEL Panning	R/W	13	03C1	03C0
Color Select	R/W	14	03C1	03C0

REGISTER SET (Continued)

Register Name	R/W	Index	Read Port	Write Port
CRT CONTROLLER				
CRT Controller Address	R/W		03?4	03?4
Horizontal Total	R/W	00	03?5	03?5
Horizontal Display Enable	R/W	01	03?5	03?5
Start Horizontal Blanking	R/W	02	03?5	03?5
End Horizontal Blanking	R/W	03	03?5	03?5
Start Horizontal Retrace Pulse	R/W	04	03?5	03?5
End Horizontal Retrace	R/W	05	03?5	03?5
Vertical Total	R/W	06	03?5	03?5
Overflow	R/W	07	03?5	03?5
Preset Row Scan	R/W	08	03?5	03?5
Maximum Scan Line	R/W	09	03?5	03?5
Cursor Start	R/W	0A	03?5	03?5
Cursor End	R/W	0B	03?5	03?5
Start Address High	R/W	0C	03?5	03?5
Start Address Low	R/W	0D	03?5	03?5
Cursor Location High	R/W	0E	03?5	03?5
Cursor Location Low	R/W	0F	03?5	03?5
Vertical Retrace Start	R/W	10	03?5	03?5
Vertical Retrace End	R/W	11	03?5	03?5
Vertical Display Enable End	R/W	12	03?5	03?5
Offset	R/W	13	03?5	03?5
Underline Location	R/W	14	03?5	03?5
Start Vertical Blank	R/W	15	03?5	03?5
End Vertical Blank	R/W	16	03?5	03?5
CRTC Mode Control	R/W	17	03?5	03?5
Line Compare	R/W	18	03?5	03?5

NOTES:
? = B in Monochrome Emulation Modes
? = D in Color Emulation Modes
All addresses are given in Hex

Register Name	R/W	Index	Read Port	Write Port
VIDEO DIGITAL TO ANALOG CONVERTER				
PEL Address (Write Mode)	R/W		03C8	03C8
PEL Address (Read Mode)	W			03C7
DAC State	R		03C7	
PEL Data	R/W		03C9	03C9
PEL Mask	R/W		03C6	03C6

NOTE:
1. DAC state register is located on the 82706. PEL Address, PEL Data, and PEL mask are located on the Palette DAC. The 82706 decodes accesses to these registers to generate DACR and DACW.

82706 PARAMETRICS

ABSOLUTE MAXIMUM RATINGS*

Case Temperature Under Bias .... −40°C to +85°C
Storage Temperature ..... −65°C to +150°C
Voltage to Any Pin with Respect to Ground .... −0.3V to +(V_{CC} +0.3)V
DC Supply Voltage (V_{CC}) ..... −0.3 to +7.0V
DC Input Current ..... ±10 mA

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS T_C = 0°C to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Min	Max	Units	Notes
V_{IL}	Input Low Voltage		0.8	V	
V_{IH}	Input High Voltage	2.0		V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 4 mA (Note 1)
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 4 mA (Note 1)
V_{OL1}	Output Low Voltage		0.4	V	I_{OL} = 2 mA (Note 2)
V_{OH1}	Output High Voltage	2.4		V	I_{OH} = 2 mA (Note 2)
I_{CC}	Power Supply Current		100	mA	
I_{LI}	Input Leakage Current		10	μA	$V_{SS} < V_{IN} < V_{CC}$
I_{OZ}	Tri-State Output Leakage Current	−10	10	μA	$V_{SS} < V_{OUT} < V_{CC}$

NOTES:
1. Applies to all outputs except those listed in Note 2.
2. Applies only to pins INTR, DEN, FC0, FC1, and WE.

A.C. CHARACTERISTICS T_C = 0°C to +70°C, V_{CC} = 5V ±10%

Timing Requirements
AC Timings are referenced to 1.5V

Symbol	Parameter	Min	Max	Units	Notes
T1	Clock Cycle Time	35	10000	ns	
T2	Clock High Time	10	10000	ns	
T3	Clock Low Time	14	10000	ns	
T4	A19:0, MI0 Valid to RD or WR Low	35		ns	
T4A	A19:0, MCS Hold from CAS Low	0		ns	(Note 6)
T4B	A19:0, MCS Hold from RD High	0		ns	(Note 7)
T4C	RDY High to RD, WR High	0		ns	(Note 5)
T5	WR Pulse Width	70		ns	(Note 2)
T6	D7:0 Set-Up to WR High	60		ns	(Note 2)
T7	WR High to D7:0 Hold	16		ns	(Notes 2, 8)
T7A	DEN High to D7:0 Hold	0		ns	(Notes 1, 8)
T7B	RDY High to D7:0 Hold	0		ns	(Note 6)
T7C	WR Low to D7:0 Valid		30	ns	(Note 6)
T8	D37:00 Valid to CLK High	0		ns	
T9	CLK High to D37:00 Hold	40		ns	
T9A	CAS High to D37:00 Hold	0		ns	

Timing Responses
AC Timings are referenced to 1.5V

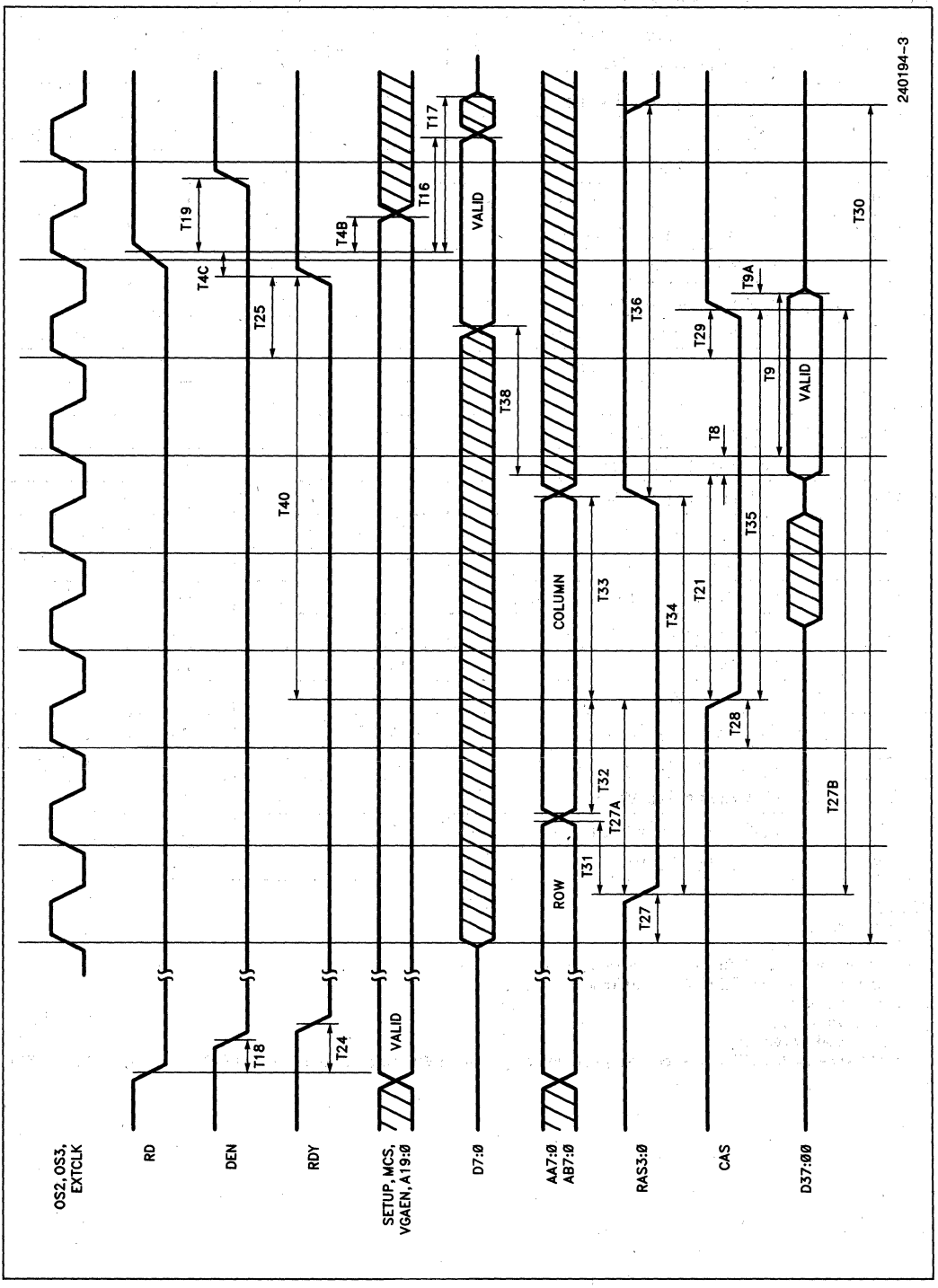
Symbol	Parameter	Min	Max	Units	Notes
T10	DCLK High Time	7		ns	(Note 3)
T11	DCLK Low Time	9		ns	(Note 4)
T12	P7:0 Valid to DCLK High	12		ns	
T13	DCLK High to P7:0 Invalid	15		ns	
T13A	DCLK High to BLANK Valid		40	ns	
T13B	DCLK High to HSYNC, VSYNC Valid		65	ns	
T14	A19:0 MI0 Valid to SFDBK Valid		60	ns	
T15	RD Low to D7:0 Valid		60	ns	
T16	RD High to D7:0 Invalid	0		ns	
T17	RD High to D7:0 Float		20	ns	

A.C. CHARACTERISTICS T_C = 0°C to +70°C, V_{CC} = 5V ±10% (Continued)

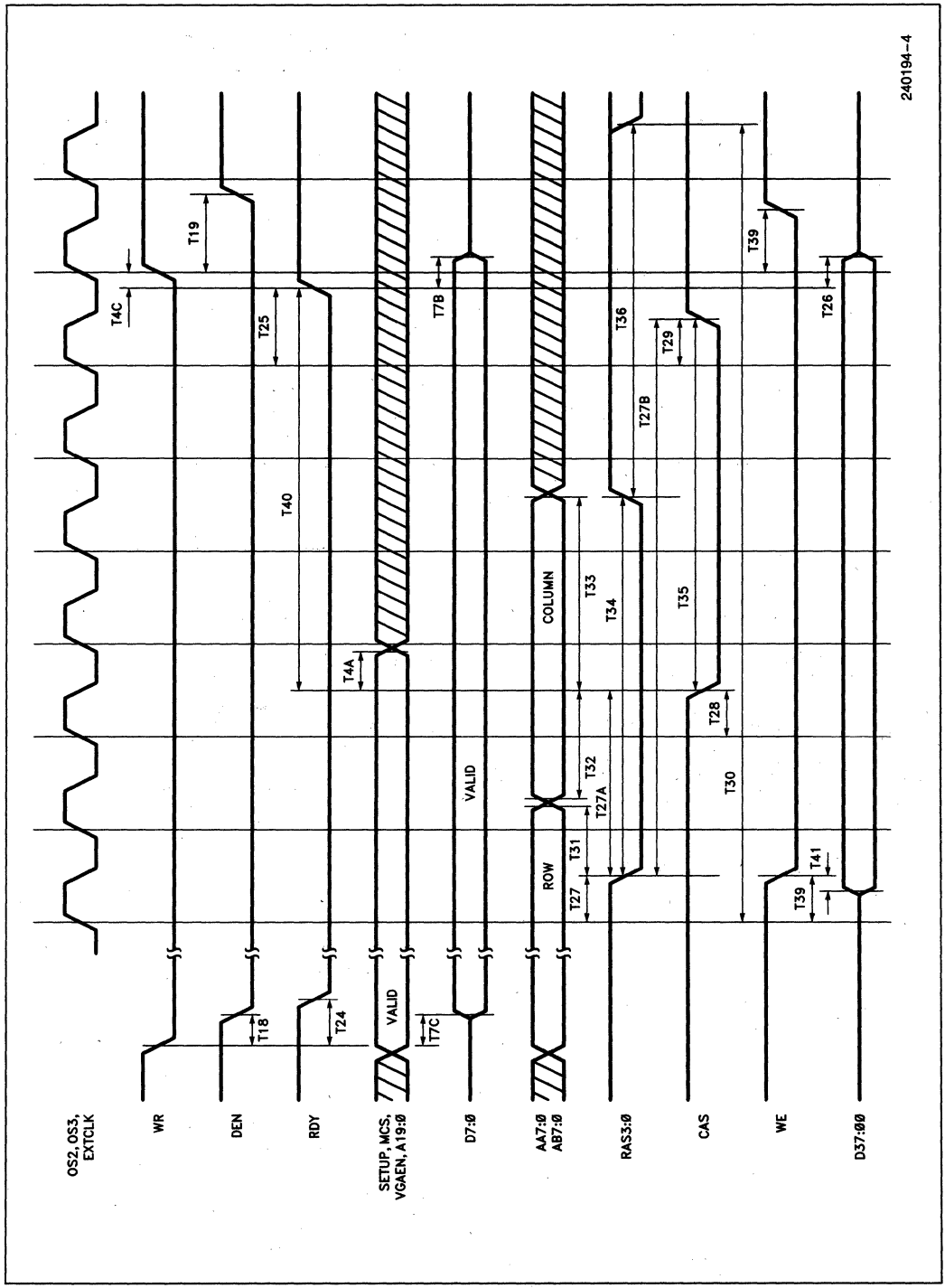
Symbol	Parameter	Min	Max	Units	Notes
T18	RD, WR Low to DEN Low		65	ns	
T19	RD, WR High to DEN High	5	30	ns	
T20	WR High to DEN High	6		ns	(Note 2)
T21	CAS Low to D37:00 Valid		70	ns	
T22	RD, WR Low to DACR, DACW Low		55	ns	
T23	RD, WR High to DACR, DACW High	4	25	ns	
T24	RD, WR Low to RDY Low		40	ns	
T25	CLK High to RDY High	0	60	ns	
T26	DATA Hold from RDY High	0		ns	(Note 6)
T27	CLK High to RAS 3:0 Low		50	ns	
T27A	RAS Low to CAS Low	50	70	ns	
T27B	RAS Low to CAS High	60		ns	
T28	CLK High to CAS Low		45	ns	
T29	CLK High to CAS High		45	ns	
T30	DRAM Cycle Time	8T1-10		ns	
T31	Row Address Hold Time	T1-10		ns	
T32	Column Address Set-up Time	10		ns	
T33	Column Address Hold Time	2T1-10		ns	
T34	RAS Pulse Width	4T1-10		ns	
T35	CAS Pulse Width	4T1-10		ns	
T36	RAS Precharge Time	4T1-15		ns	
T38	D37:00 Valid to D7:0 Valid		76	ns	(Note 7)
T39	CLK High to WE Valid		50	ns	
T40	CAS Low to RDY High	58		ns	
T41	D37:00 Valid to WE	0		ns	

NOTES:
1. I/O cycles.
2. I/O write cycles only.
3. T10 is typically at least T3 (Clock Low Time) − 4.
4. T11 is typically at least T2 (Clock High Time) − 1.
5. Memory cycles only.
6. Memory write cycles.
7. Memory read cycles only.
8. For I/O write cycles, D7:0 must be held past the rising edge of WR for at least T7 or until DEN goes high (T20 + T7A), whichever is least.

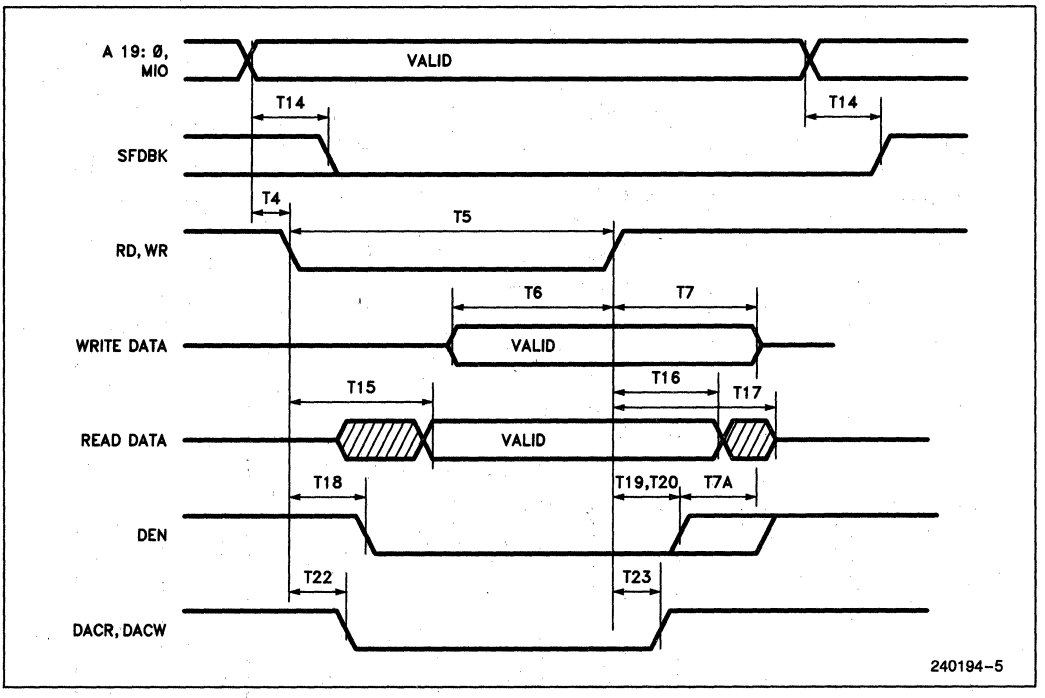
MEMORY READ


240194-3
OS2, OS3, EXTCLK
RD
DEN
RDY
SETUP, MCS, VGAEN, A19:0
D7:0
AA7:0 AB7:0
RAS3:0
CAS
D37:00
T19
T4C
T25
T40
T4B
T17
T16
VALID
T38
T36
T29
T9A
T9
T8
T35
T34
T21
T33
COLUMN
T32
T27A
T31
ROW
T27
T28
T30
T27B
T18
T24
VALID

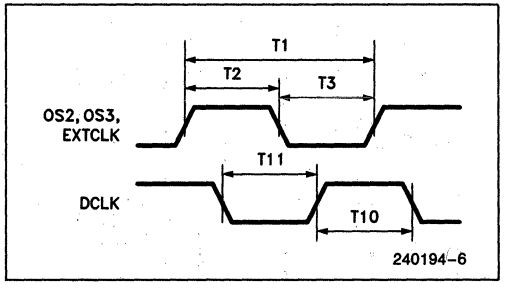
MEMORY WRITE



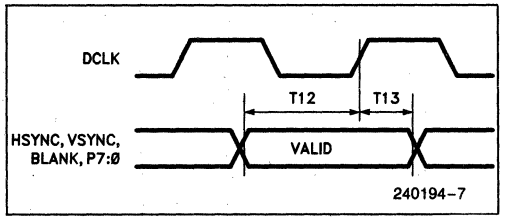
I/O CYCLE



CLOCK TIMINGS



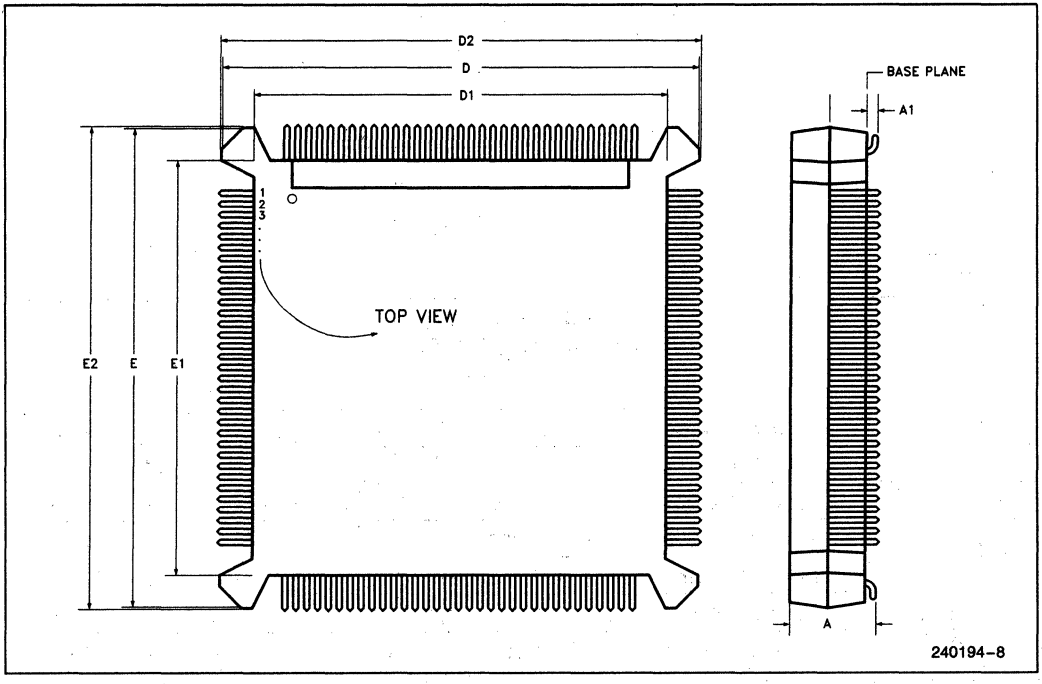
VIDEO TIMINGS



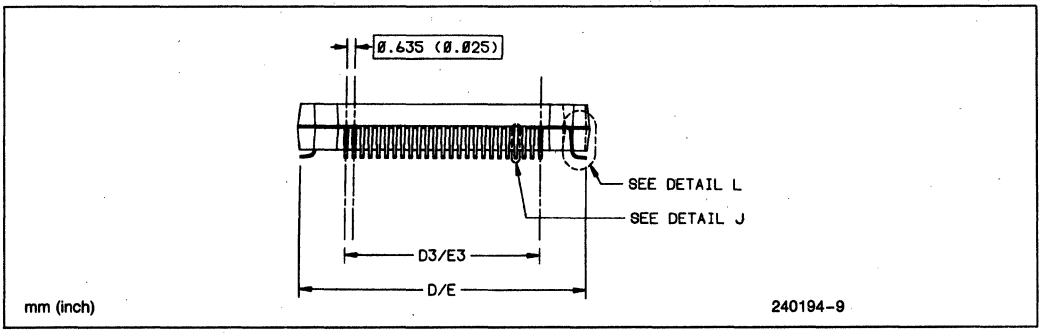
PLASTIC PACKAGING INFORMATION

The 82706 comes in a JEDEC Standard Gull Wing package (25 mil pitch), with "bumpers" on the corners for ease of handling.

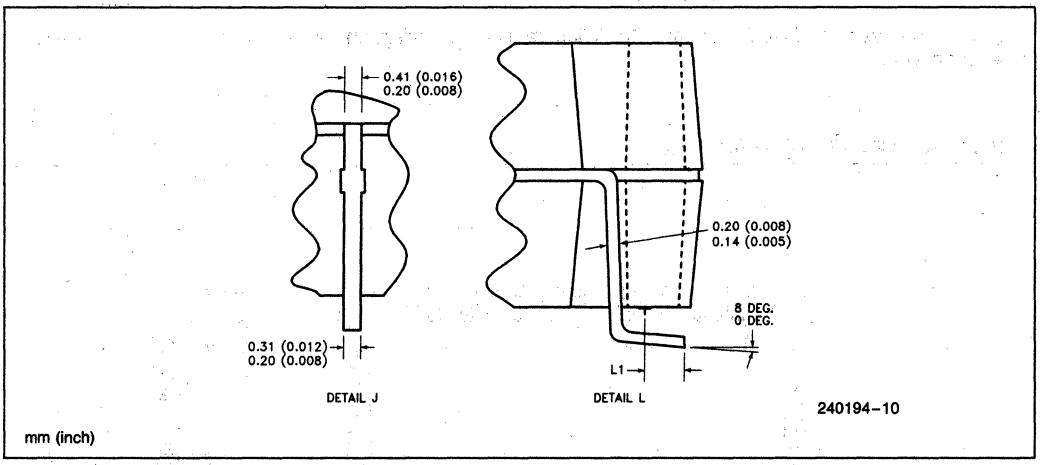
PRINCIPAL DIMENSIONS & DATUMS



TERMINAL DETAILS



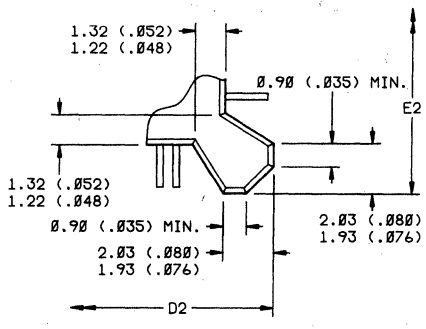
TYPICAL LEAD



Case Outline Drawings
Plastic Fine Pitch Chip Carrier
0.84 mm Pitch

Symbol	Description	Min	Max	Min	Max
N	Lead Count	132		132	
A	Package Height	0.160	0.170	4.06	4.32
A1	Standoff	0.020	0.030	0.51	0.76
D, E	Terminal Dimension	1.075	1.085	27.31	27.56
D1, E1	Package Body	0.947	0.953	24.05	24.21
D2, E2	Bumper Distance	1.097	1.103	27.86	28.02
D3, E3	Lead Dimension	0.800 Ref		20.32 Ref	
L1	Foot Length	0.020	0.030	0.51	0.76
		Inch		mm	

BUMPER DETAIL



mm (inch)

DATA SHEET REVISION REVIEW

This 82706 data sheet, version -002, contains updates and improvements to the previous version. A revision summary is listed here for your convenience.

The sections significantly revised since version -001 are:

Packaging Information — Drawing of ceramic package replaced with drawing of plastic package.
— Plastic package information section added containing mechanical information.

Pin Description — A note was added to the description of the SETUP pin.

Register Set — Video DAC register information added.

Timing Specs — Values for timings T7, T8, and T9 changed.
— Spec T21 added.

Timing Diagrams — Memory Read timing diagram altered as follows:
— SETUP and VGAEN added.
— T8 and T9 are now referenced from clock cycle earlier.
— T21 is added.
— T38 reference is changed.

82335
HIGH-INTEGRATION INTERFACE DEVICE FOR 386SX™ MICROPROCESSOR BASED PC-AT SYSTEMS

- **Operates with the 82230 and 82231 to Provide 100% IBM AT™ Compatibility**
- **Optimized for 16 MHz 386SX™ Microprocesor Based AT Systems**
- **Page Mode, Interleaved DRAM Controller**
- **Address Mapping/Shadow ROM Support**
- **80387SX Numerics Coprocessor Synchronization Interface**
- **Parity Generation and Checking**
- **Low Power, High Speed CHMOS III Technology**
- **Available in 132 Lead Plastic Quad Flat Pack**

The Intel 82335 is a high integration interface device used together with the 82230/82231 to provide the most cost-effective and highest performance system design solution for AT-compatible 386SX™ microprocessor based systems.

The 82335 DRAM control feature is designed and optimized for the 16 MHz 386SX microprocessor bus architecture. The page mode, interleaved memory design allows 0 wait state performance on most memory accesses with 100 ns DRAM.

Several address mapping options are available to provide flexibility in the system memory size and configuration.

The 82335 also provides the necessary interface signals to allow the 387SX numerics coprocessor to run in a PC/AT system. The 82235 with its intergrated parity generation and checking provides system designers with data integrity and reliability.

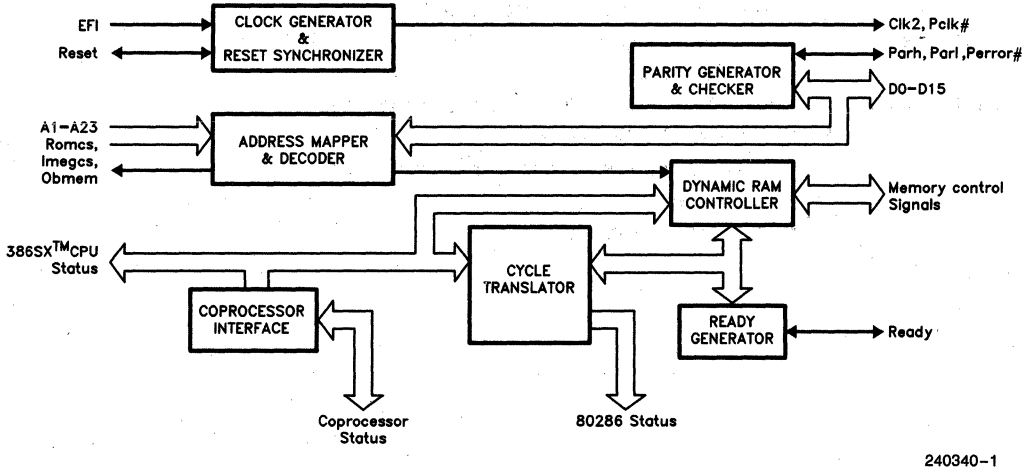


Figure 1.1. 82335 Internal Block Diagram

November 1988
Order Number: 240340-001

1.0 PIN DESCRIPTION

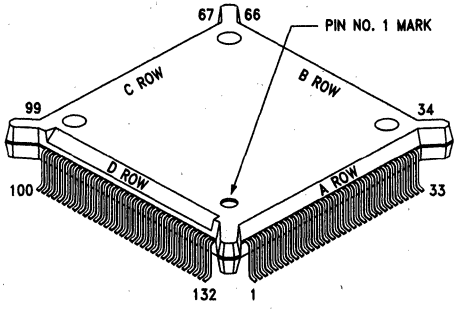


Figure 1.2. 82335 Pin Out Top View

Table 1.1. Pin Assignments

A Row		B Row		C Row		D Row	
Pin	**Label**	**Pin**	**Label**	**Pin**	**Label**	**Pin**	**Label**
1	RESETNPX	34	A20	67	TEST0	100	READYNPX#
2	EFI	35	A21	68	TEST1	101	PEREQNPX
3	V_{SS}	36	A22	69	V_{CC}	102	LMEGCS#
4	PCLK#	37	A23	70	RAS2#	103	HLDASX
5	V_{CC}	38	ROMCS1#	71	CASL2#	104	HRQSX
6	RESETCPU	39	ROMCS0#	72	CASH2#	105	V_{SS}
7	READY286#	40	OBMEM	73	RAS3#	106	NA#
8	M/IO286#	41	MA9	74	CASL3#	107	READYSX#
9	S1#	42	MA8	75	CASH3#	108	V_{SS}
10	V_{SS}	43	V_{SS}	76	DIR	109	CLK2
11	S0#	44	MA7	77	DEN#	110	V_{CC}
12	V_{CC}	45	V_{CC}	78	PARL	111	ADS#
13	ERROR#	46	MA6	79	V_{SS}	112	BLE#
14	A2	47	MA5	80	PARH	113	A1
15	A3	48	MA4	81	V_{CC}	114	BHE#
16	A4	49	V_{SS}	82	D15	115	V_{CC}
17	A5	50	MA3	83	D14	116	M/IO#
18	A6	51	V_{CC}	84	D13	117	D/C#
19	A7	52	MA2	85	D12	118	W/R#
20	V_{SS}	53	V_{SS}	86	D11	119	RESETSX
21	A8	54	MA1	87	D10	120	BUSYSX#
22	V_{CC}	55	V_{CC}	88	D9	121	PEREQSX
23	A9	56	MA0	89	D8	122	HRQ286
24	A10	57	WE#	90	D7	123	REFRESH#
25	A11	58	RAS0#	91	D6	124	PERROR#
26	A12	59	CASL0#	92	D5	125	MEMR#
27	A13	60	CASH0#	93	D4	126	SYSRESET
28	A14	61	RAS1#	94	D3	127	A20GATE
29	A15	62	CASL#	95	D2	128	V_{SS}
30	A16	63	CASH1#	96	D1	129	MEMW#
31	A17	64	V_{SS}	97	D0	130	V_{CC}
32	A18	65	MMS	98	TURBO#	131	BUSYNPX#
33	A19	66	FM	99	EXTRDY	132	HLDA

The 82335 is implemented in a 132-pin plastic flatpack package designed for direct surface mounting on component boards. The following is a description of the physical pin connections.

Table 1.2. Pin Description

Symbol	Pin No.	Type	Name and Function
A1–A23	14–19, 21, 23–37, 113	I*	**ADDRESS INPUTS:** These inputs are used to select the dynamic RAM address for a memory read or write operation.
A20GATE	127	I	This active high input is used by the keyboard controller to force A20 low. When A20GATE is low, A20 is forced low internal to the 82335 during CPU memory cycles (not DMA or master). When A20GATE is high, A20 follows the CPU address input from the A20 pin.
ADS#	111	I	**ADDRESS STATUS:** This active low input indicates that a valid bus cycle definition and address (W/R#, M/IO#, D/C#, BLE#, BHE#, and A1–A23) is being driven by the 386SX microprocessor.
BHE#	114	I	**BYTE HIGH ENABLE:** This active low input is used for the physical address of the high byte of a 16-bit data word. It indicates when data is being transferred on D8–D15.
BLE#	112	I	**BYTE LOW ENABLE:** This active low input is used for the physical address of the low byte of a 16-bit data word. It indicates when data is being transferred on D0–D7.
BUSYNPX#	131	I	**BUSY NP:** This active low input is used by the 80387SX to indicate that it is busy. It is usually connected directly to BUSY# of 80387SX.
BUSYSX#	120	O	**BUSY SX:** This active low output indicates to the 386SX CPU that the 80387SX is busy. It is usually connected directly to BUSY# of the 386SX CPU.
CASH0# – CASH3#	60, 63, 72, 75	O	**COLUMN ADDRESS STROBE (HIGH BYTE):** These outputs are used by the high byte of the dynamic RAM array to latch the column address present on the MA0–MA9 pins. They can drive the dynamic RAM array directly and need no external drivers.
CASL0# – CASL3#	59, 62, 71, 74	O	**COLUMN ADDRESS STROBE (LOW BYTE):** These outputs are used by the low byte of the dynamic RAM array to latch the column address present on the MA0–MA9 pins. They can drive the dynamic RAM array directly and need no external drivers.
CLK2	109	O	**CLOCK2:** This output drives the 386SX CPU and 80387SX input clocks. Its frequency is divided by two internally by the 386SX CPU and 80387SX chips to generate the processor and coprocessor clocks. It is designed for 32 MHz output frequency.
D/C#	117	I	**DATA/CONTROL SELECT:** This input from the 386SX microprocessor is used to distinguish between data and control bus cycles.
DEN#	77	O	**DATA ENABLE:** This active low output is used by the data transceivers to enable the transfer of data between the dynamic RAM array and the local data bus.
DIR	76	O	**DIRECTION:** This signal is used to control the direction input of the data transceivers which connect the dynamic RAM array to the local data bus. When DIR is high, data is being written into memory.

NOTES:
*The following conventions are used in this table:
I = Input
O = Output
I/O = Input or Output
The symbol # following a signal name indicates that the signal is low active.

Table 1.2 Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
D15–D0	82–97	I/O	**DATA/BUS:** These inputs are used by the 82335 for parity generation and checking of data which is transferred between the local bus and the DRAM array. During initialization of the 82335, they are used to write/read control words to/from the internal memory configuration registers.
EFI	2	I	**EXTERNAL FREQUENCY IN:** This input is driven by an external oscillator. It is used by the 82335 to generate the CLK2 and PCLK# output clocks. All internal 82335 logic is also driven by EFI. The EFI frequency is the same as the CLK2 (32 MHz).
ERROR#	13	I	**ERROR:** This input indicates when a numeric coprocessor error has occurred. ERROR# is directly connected to the ERROR# output of the 80387SX and ERROR# input of the 82230. ERROR# has a weak pull-up resistor inside the 82335.
EXTRDY	99	I	**EXTERNAL READY:** The External Ready input is an active high level triggered input which is used to insert additional wait states into 386SX processor bus cycles. Deactivation of EXTRDY during a bus cycle delays the active edge of the 82335 READYSX# output until the EXTRDY input is sampled active.
FM MMS	66 65	I I	FM and MMS are used to select DRAM operating modes. Refer to the DRAM controller section for available modes. These pins should be static after system reset.
HLDA	132	O	**HOLD ACKNOWLEDGE:** This output indicates the 386SX CPU has relinquished control of the local bus. It is asserted in response to activation of the HLDASX input from the 386SX processor. It is usually connected to the 82231 HLDA input.
HLDASX	103	I	**HOLD ACKNOWLEDGE SX:** This input is asserted by the 386SX processor in response to assertion of the 386SX processor's HOLD pin. It indicates that the processor has relinquished control of the local bus.
HRQ286	122	I	**CPU HOLD REQUEST INPUT:** This active high input is driven by the 82231 CPU HRQ pin when requesting DMA or refresh cycles.
HRQSX	104	O	**CPU HOLD REQUEST OUTPUT:** This active high output drives the 386SX processor's HOLD input. It is the HRQ286 input with the trailing edge delayed.
LMEGCS#	102	O	**LOWER MEG CHIP SELECT:** This decodes A23–A20 for local memory accesses.
MA0–MA9	41, 42, 44, 46–48, 50, 52, 54, 56	O	**MULTIPLEXED ADDRESS:** These outputs are designed to provide the row and column addresses for CPU or DMA access, and row addresses for refresh access to the dynamic RAM array.
MEMR#	125	I	**MEMORY READ COMMAND:** This active low input from the 82231 indicates when a DMA memory read cycle is being performed. MEMR# is connected directly to the −XMEMR output of the 82231. MEMR# is allowed to be an asynchronous input.
MEMW#	129	I	**MEMORY WRITE COMMAND:** This active low input from the 82231 indicates when a DMA memory write cycle is being performed. MEMW# is connected directly to the −XMEMW output of the 82231. MEMW# is allowed to be an asynchronous input.

Table 1.2. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
M/IO#	116	I	**MEMORY/IO SELECT:** This input from the 386SX processor is used to distinguish between memory and IO accesses.
M/IO286#	8	O	**MEMORY I/O SELECT 286:** This output emulates the M/IO# output of the 80286. It is used by the 82230/82231 and other system peripherals to distinguish memory access from I/O access. It also indicates halt/shutdown and interrupt acknowledge cycles.
NA#	106	O	**NEXT ADDRESS:** The NA# output is used to control the address pipelining of the 386SX processor. When asserted, the address of the next bus cycle is valid in the T2P state of the current bus cycle. Consecutive local memory accesses are always pipelined after the first access.
OBMEM	40	O	**ON-BOARD MEMORY:** This active high output indicates to the system that the memory being address is on the local bus.
PARH	80	I/O	**PARITY HIGH BYTE:** This three state input/output is used for the upper byte parity bit of data on the local bus (D8–D15). For memory write cycles, the 82335 outputs the internally generated parity bit to the DRAM array via the PARH pin. During a memory read, the 82335 uses the data received at PARH to validate the upper byte of data from the DRAM array.
PARL	78	I/O	**PARITY LOW BYTE:** This 3-state input/output is used for the lower byte parity bit of data on the local bus (D0–D7). Its function is identical to the PARH pin described above.
PCLK#	4	O	**PROCESSOR CLOCK:** This clock output is used to drive the 82230 X3 pin. It is a divide-by-two of the EFI input.
PEREQNPX	101	I	**PROCESSOR EXTENSION REQUEST NP:** This input is used by the 80387SX to indicate that it requires a data transfer. It should be connected directly to PEREQ of the 80387SX. If no numeric coprocessor is installed, this pin should be connected to ground.
PEREQSX	121	O	**PROCESSOR EXTENSION REQUEST SX:** This output to the 386SX processor is used to request a data transfer to or from the numeric coprocessor. It should be connected directly to PEREQ of the 386SX CPU.
PERROR#	124	O	**PARITY ERROR:** This active low output indicates that the 82335 has detected a parity error in either the upper or lower byte of data from the DRAM array. It drives the 82231 DPCK# input.
RAS0# – RAS3#	58, 61, 70, 73	O	**ROW ADDRESS STROBE:** These outputs are used by the dynamic RAM array to latch the row address present on the MA0–MA9 pins. The four outputs support up to a four-way interleaved dynamic RAM configuration with page-mode access. They drive the dynamic RAM array directly and need no external drivers.
READY286#	7	I	**READY 286:** This active low input is used to indicate the completion of I/O bus cycles. It is driven by the 82230 READY# pin.
READYNPX#	100	I	**READY NP:** This active low input indicates the completion of 80387SX bus cycles. It is driven by the 80387SX READYO# output.
READYSX#	107	O	**READY SX:** This active low output indicates the completion of the current bus cycle to the processor. It is a function of the internally generated memory ready signal, and the READY286#, READYNPX#, EXTRDY, and TURBO# inputs.

Table 1.2. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
REFRESH#	123	I	**REFRESH:** This active low input is used to notify the dynamic RAM controller that the dynamic RAM array requires refresh. It is normally driven by the 82231 −REFRESH output.
RESETCPU	6	I	**RESET CPU:** This reset input is driven by the RES CPU output of the 82230. It is activated during system power up, keyboard reset, or when the 386SX processor generates a HALT status.
RESETNPX	1	O	**RESET NPX:** This output drives the RESETIN pin of the 80387SX. RESETNPX is a function of the SYSRESET input and is only activated during power-on reset.
RESETSX	119	O	**RESET SX:** This output drives the RESET pin of the 386SX processor. It is a logical OR function of the SYSRESET and RESETCPU inputs.
ROMCS0# ROMCS1#	39 38	O	**ROM CHIP SELECT:** These outputs are used to support shadow RAM. They select the ROMs or EPROMs during system initialization. Once the ROM or EPROM contents are copied into the DRAM space, the ROMCS0−1 outputs are disabled and the ROM (EPROM) addresses are mapped into the DRAM physical address space by the 82335.
S0# S1#	11 9	O	**BUS CYCLE STATUS:** The S0# and S1# outputs indicate the initiation of a system (non-local) bus cycle and, along with M/IO286#, define the type of bus cycle.
SYSRESET	126	I	**SYSTEM RESET:** This reset input is driven by the +RESET output of the 82230. It is driven active during system power up.
TEST0 TEST1	67 68	I	**TEST MODE:** These inputs are reserved for special test modes, and must be connected to V_{SS} during normal operation.
TURBO#	98	I	**TURBO MODE SELECT:** This active low input, when asserted, allows the 386SX processor local bus to run with maximum performance. Deactivating the TURBO# input causes the 82335 READY generation logic to insert additional wait states into each bus cycle. In the non-turbo mode, 386SX processor performance approximates 80286 bus efficiency.
V_{CC}	5, 12, 22, 45, 51, 55, 69, 81, 110, 115, 130	—	**POWER SUPPLY:** 11 V_{CC} pins total.
V_{SS}	3, 10, 20, 43, 49, 53, 64, 79, 105, 108, 128	—	**GROUND:** 11 V_{SS} pins total.
WE#	57	O	**WRITE ENABLE:** This output is used by the dynamic RAM array to enable input for a write operation. It is designed to drive eight megabytes of DRAM with no additional buffering.
W/R#	118	I	**WRITE/READ SELECT:** This input from the 386SX processor is used to distinguish between read and write cycles.

2.0 FUNCTIONAL DESCRIPTION

2.1 Introduction

The 82335 is a high-integration VLSI companion chip for the Intel 386SX 32-bit microprocessor. It interfaces the 386SX microprocessor to the 80387SX numeric coprocessor and to the 82230/82231 highly integrated peripherals in an AT compatible system by converting 386SX processor bus cycles to 80286 compatible cycles, generating necessary clock signals, and providing local dynamic memory control. Figure 2.1 shows a block diagram of this system.

The 82335 is composed of seven functional blocks:

1. DRAM Controller
2. Address Mapper/Decoder
3. Ready Generator
4. Bus Cycle Translator
5. Math Coprocessor Interface
6. Clock Generator/Reset Synchronizer
7. Parity Generator/Checker

Each functional block is described in the following sections.

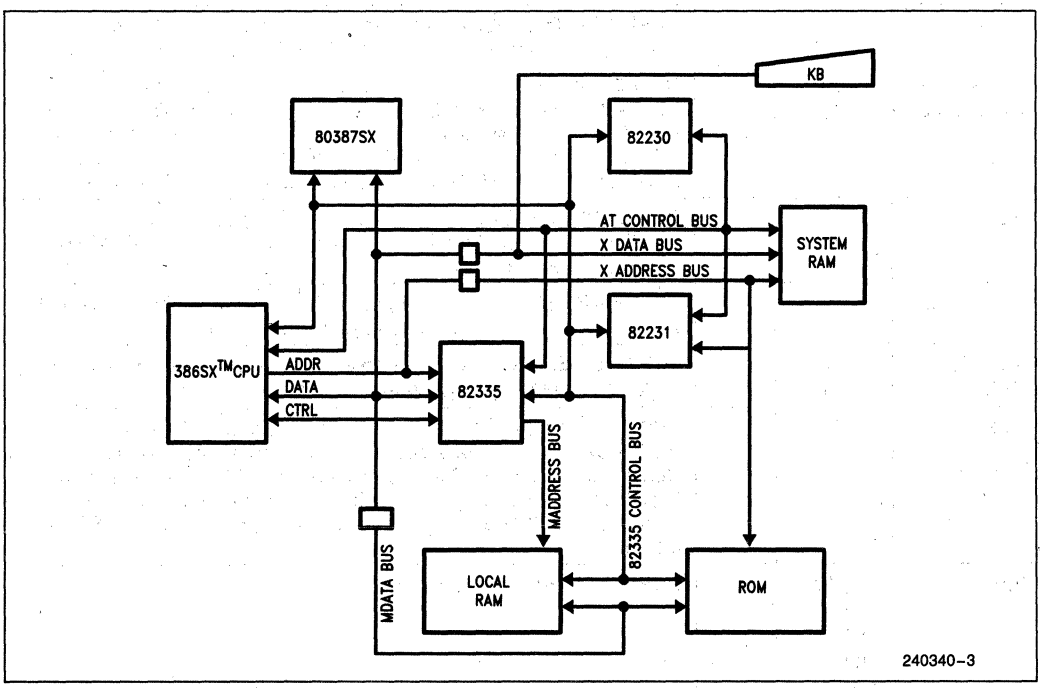


Figure 2.1. 386SX™ CPU with 82335 System Block Diagram

2.2 DRAM Controller

2.2.1 INTRODUCTION

The 82335 dynamic RAM (DRAM) controller is designed and optimized for the Intel 386SX™ Architecture. It keeps track of 386SX CPU bus states and provides the necessary signals to address and refresh up to four 16-bit banks of 256K or 1M dynamic RAMs.

To optimize memory performance and flexibility, the DRAM controller has built in support for both paging and bank interleaving, and can be configured for several different modes of operation. These include four different memory modes to accommodate DRAM with different levels of performance: two for fast page-mode 100 ns DRAM's (F1 and F4) and two for slower DRAM's (W01 and W02). These modes are described in more detail in the DRAM mode configuration section.

In addition to the four memory modes available, the DRAM controller can be configured to operate in either turbo or non-turbo mode. The turbo mode allows the 386SX microprocessor based system to run at peak efficiency, while the non-turbo mode allows it to approximate 80286 bus cycles for timing dependent software.

2.2.2 DRAM BANK CONFIGURATION

The local Dynamic RAM for the 386SX CPU/82335 system can be configured into one to four banks of 256K x 16 bits or 1M x 16 bits each. Each 16-bit bank of memory is further divided into two 8-bit banks, low and high. Each 8-bit bank may contain one extra bit for parity. See Figure 2.2 for a block diagram of the 82335 to DRAM interface.

The exact memory configuration installed is determined during system initialization through execution of a memory autoscan routine in the BIOS. Relevant memory configuration information is then programmed into the memory configuration register, roll compare registers, and address compare registers. See the address mapping/decoding section for details on register programming.

The 82335 internally decodes the address lines A1–A23 from the 386SX processor and outputs multiplexed row and column addresses, row address strobe (RAS), column address strobe (CASL & CASH), and write enable (WE) signals for local memory accesses. Each bank has separate RAS, CASL, and CASH signals.

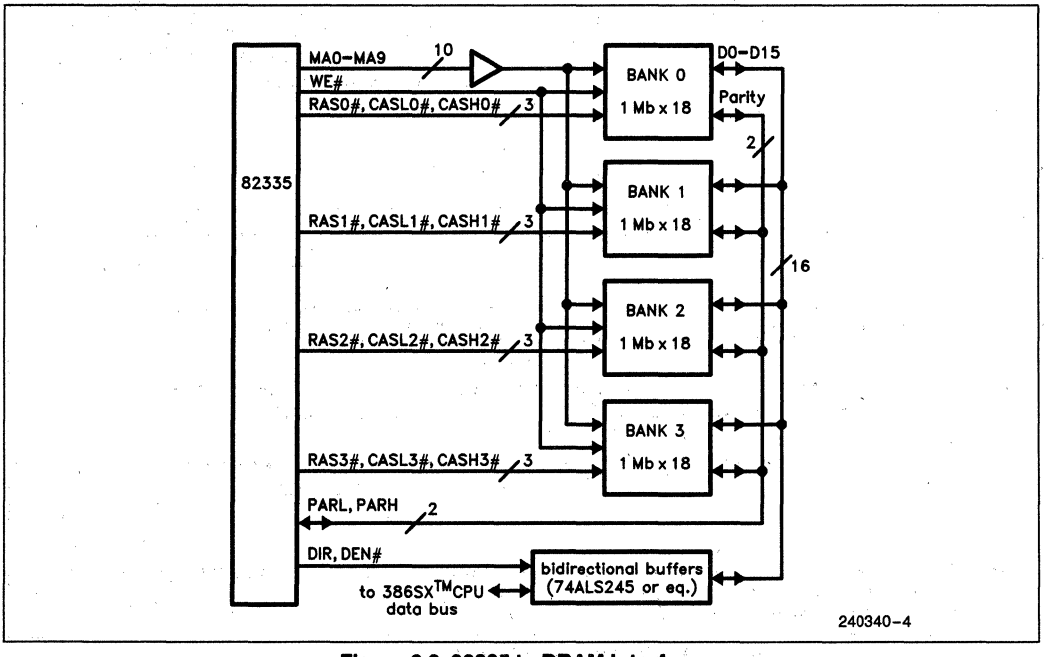


Figure 2.2. 82335 to DRAM Interface

2.2.3. PAGE-MODE DRAM OPERATION

In any DRAM access, read or write, a row address and a column address is required. A DRAM access requiring both a new row address and column address has a long cycle time which causes two or more wait states for a 386SX CPU bus cycle. An example is shown in Figure 2.3. Bus performance is improved in 386SX CPU/82335 systems by the use of page-mode DRAM operation. Memory locations sharing the same row address are in the same memory page. When successive memory accesses are in the same page (a page-hit memory access), only a new column address is required. In this mode of operation, the row address strobe, RAS#, can be kept active, and only a new CAS# edge needs to be generated, thus reducing memory cycle time. An example with both page hit and page miss (opposite of page hit) is shown in Figure 2.4. Page-mode DRAM operation is only effective when successive memory accesses are in the same page, therefore, row addresses are taken from the higher order address bits since they are less likely to change than the lower order address bits in most programs.

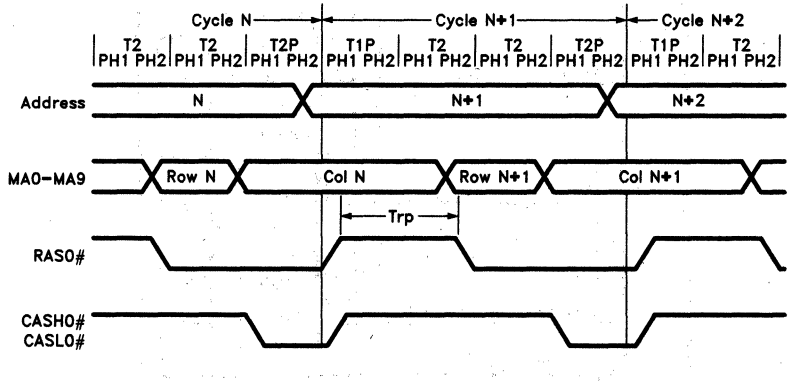
2.2.4 PAGE-MODE BANK INTERLEAVE OPERATION

The effectiveness of page-mode operation in the reduction of number of wait states depends on many factors. Among the most important of them are: page locations, page size and page-mode cycle time.

Page Location: As mentioned in the above paragraph, the memory pages should be arranged such that the row address is unlikely to change in successive memory operations. This can be done by assigning the more often lower order address bits to the column address as done in the 82335 DRAM controller.

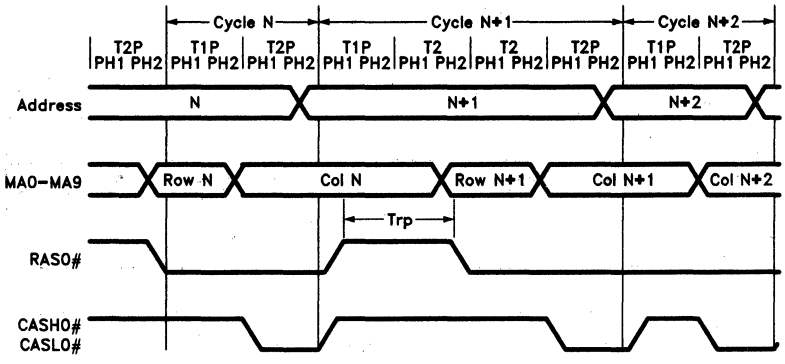
Page Size: A larger page size increases the chance of a page hit. In a DRAM configuration with more than one memory bank, one page of memory can be kept active per bank. In a four bank configuration, a maximum of four pages of memory can be kept active at a time. A successive memory access to an active page in a different bank (a page-hit-bank-miss access) does not require a new row address, thereby requiring no wait states. This effectively increases the DRAM page size by a factor of four.

Page-Mode Cycle Time: Most fast page-mode 100 ns DRAM's have a short page-mode cycle time which allows the 386SX processor to run 0-wait-state bus cycles. Slower DRAM's, however, have longer page-mode cycle times which causes the 386SX processor to run at 1 or 2 wait states. The 82335 uses a memory interleaving scheme to allow the 386SX processor to run 0-wait bus cycles. It works as follows: In a DRAM configuration with more than one memory bank, one page of memory is kept active per bank. Slower memories (W01/W02 modes) use the lower address bit(s) to alternate bank hits for consecutive memory accesses. This increases the number of page-hit-bank-miss cycles which run at 0 wait states.



NOTES:
1. The address of the next bus cycle is read into the 82335 during phase 2 of the first T2P state of the current bus cycle.
2. RAS# must be high for row precharge (T_{rp}) before latching in a new row address.
3. Two wait states are inserted to meet RAS# precharge time.

Figure 2.3. 82335 DRAM Cycle with New Row and Column Addresses



NOTES:
1. Page miss in cycle N+1, thus requiring 2 wait states. Notice RAS0# returns to "1" in the first part of cycle N+1 for precharge (T_{rp}).
2. Page hit on cycle N+2, therefore, no wait state is required between cycle N+1 and N+2. Notice the 2 consecutive CAS# pulses without intervening RAS# pulse.

Figure 2.4. 82335 Page Mode DRAM Cycle with Both Page Miss and Page Hit

The 82335 has 4 built-in watch-dog timers to keep track of RAS# active time. Once timed out, the timer will force RAS# of the corresponding bank to be deactivated at the end of the memory cycle such that the maximum RAS# active time of the DRAM will not be violated. Figure 2.5 shows an example of interleaved memory cycles.

2.2.5 DRAM MODE CONFIGURATION

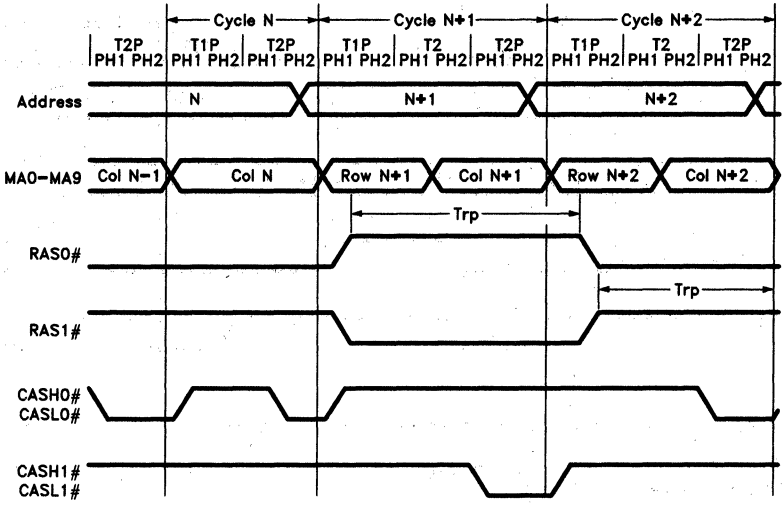
The 82335 can be configured to run in four different modes to operate with DRAM of various performance levels. This allows the system designer considerable flexibility. There are two modes (F1 and F4) for fast page mode 100 ns DRAM and two modes (W01 and W02) for slower DRAM. The mode of operation is selectable by setting the input pins FM and MMS to the values shown in Table 2.1. Table 2.1 shows a summary of the different DRAM modes. A brief description of each mode follows.

F4: This mode is for fast 100 ns DRAM and allows up to four pages to be active simultaneously. The critical timing specifications that determine which 100 ns DRAM can use this mode are CAS access time (t_{CAC}), CAS pulse width (t_{CAS}), and CAS precharge time (t_{CP}). Table 2.1 shows the maximum values for t_{CAC}, t_{CAS} and t_{CP} that can be used in F4 mode.

F1: This mode is the same as F4 except that only one page can be active at a time. Activating only one page at a time reduces power consumption.

W01: This mode can be used for all 100 ns DRAM.

W02: This mode is used for 120 ns DRAM.



NOTES:
1. DRAM access alternates between banks 0 and 1.
2. RAS precharge for bank 0 is hidden behind memory access of bank 1 and vice versa.

Figure 2.5. 82335 DRAM Cycle with Interleaved Memory in F1 Mode

Table 2.1. Summary of DRAM Modes

fm	mms	Mode	Max Pages Active	Wait States					DRAM Type	Critical DRAM Specifications*
				Page Hit		Page Miss		New RAS		
				Bank Hit	Bank Miss	Bank Hit	Bank Miss			
1	1	F4	4	0	0	2	2	1	100 ns Fast Page Mode	$t_{CAS} \leq 35$ ns $t_{CAC} \leq 35$ ns $t_{CP} \leq 20$ ns
1	0	F1	1	0	NA	2	1	1	100 ns Fast Page Mode	$t_{CAS} \leq 35$ ns $t_{CAC} \leq 35$ ns $t_{CP} \leq 20$ ns
0	1	W01	4	1	0	2	2	1	All 100 ns DRAM	
0	0	W02	4	2	0	3	3	2	120 ns DRAM	

NOTES:
*t_{CAS} = Column Address Strobe Pulse Width
t_{CAC} = Column Address Strobe Access Time
t_{CP} = Column Address Strobe Precharge Time

2.2.6 NON-TURBO MODE

In non-turbo mode (TURBO# pin driven to $>V_{IH}$), a fixed number of bus states (six per memory cycle) will be used for all local memory access. Non-turbo mode is designed for compatibility with 80286 programs with software timing loops. The TURBO# input must remain static during local memory bus cycles.

2.2.7. REFRESH CYCLE

The 82335 generates its own DRAM refresh address with a 10-bit refresh counter which increments by one every refresh cycle. During a refresh cycle, the refresh address appears on MA0–MA9, followed by activation of RAS0#–RAS3# in staggered cycles. The staggering of RAS activation reduces current surge during refresh.

2.3 Address Mapper/Decoder

2.3.1 INTRODUCTION

Several address mapping and decoding options are provided to improve performance and allow flexibility in the system memory size and configuration. These options include ROM/EPROM shadowing, mapping up to 512K addresses above the top of physical memory into physical addresses, and decoding input addresses to generate chip select signals. Selection of these options is done via programming of the configuration, roll compare, and address compare registers.

2.3.2 SHADOWING

Shadowing refers to copying data from slow memory devices like ROM and EPROM memories into RAM to speed up memory accesses. Since access to local RAM is much faster than ROM, this can provide a considerable increase in performance. The 82335 has built in support for shadowing three different areas of memory: BIOS ROM, adapter ROM, and video RAM. Shadowing video RAM can improve performance by making access to the video RAM local to the 82335 memory controller. Figure 2.6 shows the memory address ranges available for shadowing.

Shadowing can be selected for the BIOS ROM area, adapter ROM area, or video RAM area by programming the ROMEN#, ENADP#, and ENV# bits respectively of the configuration register. Each area can be selected for shadowing independently of the other areas.

When shadowing the BIOS ROM and adapter ROM, the ROM contents must be copied to the shadow RAM area before the lock bit is set in the configuration register. Once the lock bit is set, both of these RAM areas become read only. If video RAM shadowing has been enabled and the VRO bit (video read only) in the configuration register is set, then the video shadow area will also become read only.

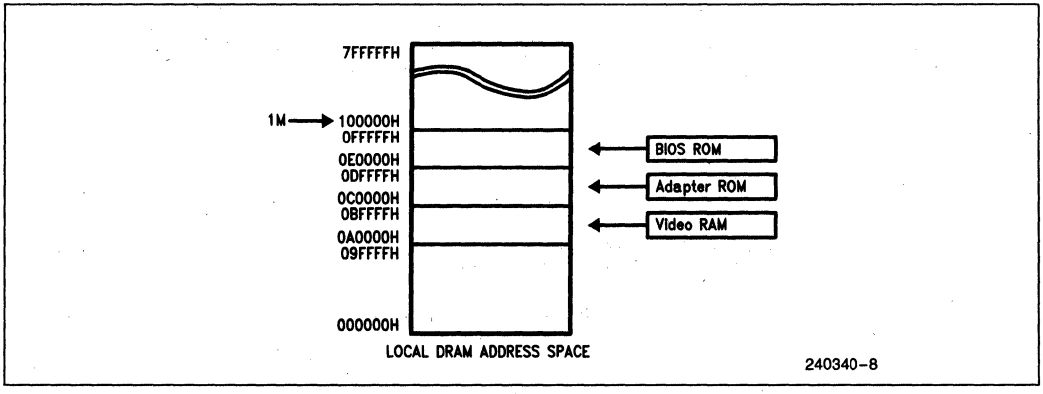


Figure 2.6. Shadow RAM Address Map

2.3.3 ROLL ADDRESS MAPPING

Roll address mapping is a method of utilizing DRAM memory space that may otherwise not be accessible. If ROM or video RAM shadowing is not selected, then any addresses generated in these areas will access the ROM or system video RAM. Any local DRAM with the same physical addresses as the ROM or video RAM cannot be directly addressed. To allow access to this DRAM space, the 82335 can re-map or "roll" logical addresses above the top of the physical address range into this DRAM space in 128 Kbyte segments. Figure 2.7 shows a memory map illustrating roll address mapping.

There are four 128 Kbyte segments of physical memory (512 Kbytes total) that can be re-mapped. These areas are in the top half of the lowest megabyte of memory as illustrated in Figure 2.7. Programming of the memory configuration register bits ROMEN# S640, ENADP#, and ENV# control which segments are available for remapping. (See the **Memory Configuration Register** section for programming details.) Enabling roll address mapping and specification of the logical addresses to be re-mapped is done through programming of the roll compare registers. (See **Roll Compare Register** section for more details.)

2.3.4 REGISTERS

There are five registers in the 82335 that control the operation of the address mapping and DRAM control options. These registers are the configuration, roll compare (RC1 and RC2), and address range compare (CC0 and CC1) registers. Each of these registers reside in the local I/O space of the 82335 and are read/writable until the LOCK bit has been set in the configuration register. The contents and purpose of each register are described in the following sections.

2.3.4.1 Memory Configuration Register

The memory configuration register resides at I/O location 22H upon system reset and is used to select a number of address mapping and DRAM control options. Upon reset, all bits in this register are set to zero. Figure 2.8 shows the bits used in the memory configuration register. The purpose of each bit is described in the following paragraphs.

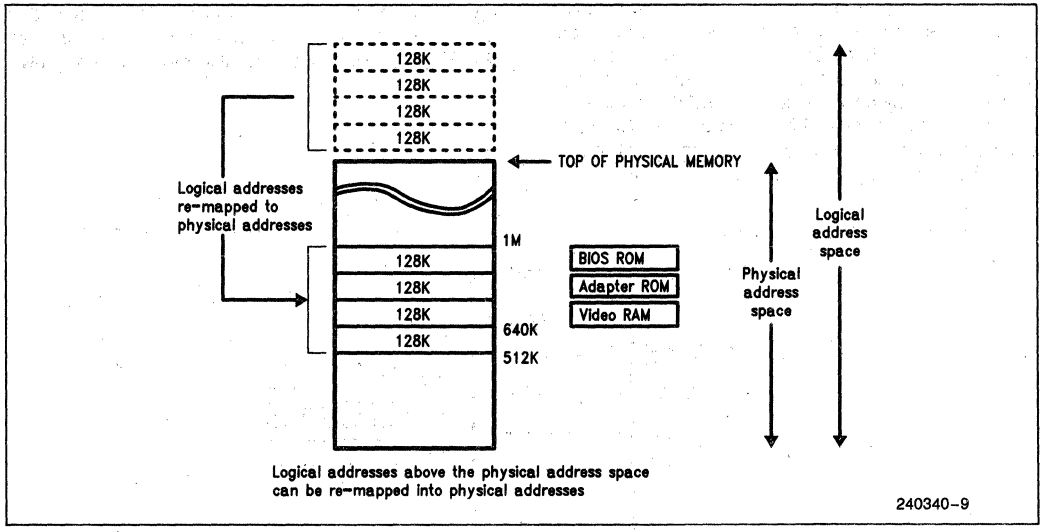
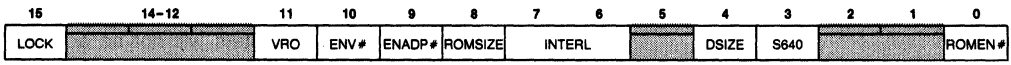


Figure 2.7. Roll Address Mapping Range

Memory Configuration Register = I/O Address 22H



Bit Position	Name	Function
0	ROMEN#	0 = Enable BIOS ROM/EPROM Accesses (0E0000H–0FFFFFH) 1 = Disable BIOS ROM/EPROM Accesses, Shadow Enabled
3	S640	0 = Base Memory Size is 512K 1 = Base Memory Size is 640K
4	DSIZE	0 = 256K 1 DRAM Installed 1 = 1 Mb DRAM Installed
7, 6	INTERL	00 = 1 Mem. Bank Installed (No Interleave) 01 = 2 Mem. Bank Installed* 10 = 3 Mem. Bank Installed* 11 = 4 Mem. Bank Installed*
8	ROMSIZE	0 = 256K ROM/EPROM; 1 = 512K ROM/EPROM
9	ENADP#	0 = Enable Adaptor ROM/EPROM Accesses (0C0000H-0DFFFFH) 1 = Disable Adaptor (ROM/EPROM Accesses, Shadow Enabled
10	ENV#	0 = Enable Video RAM Accesses (0A0000-0BFFFFH) 1 = Disable Video RAM Accesses, Shadow Enabled
11	VRO	Video Read Only 0 = Video Area Read-Write 1 = Video Area Read-Only
15	LOCK	0 = Enable all Configuration Register Accesses 1 = Disable all Configuration Register Accesses

NOTE:
*When more than one bank of memory is installed, banks are always interleaved.

Figure 2.8. Memory Configuration Register

ROMEN#: This bit is used to enable or disable shadowing of the BIOS ROM/EPROM in the address range 0E0000H–0FFFFFH. When this bit is cleared, BIOS ROM shadowing is disabled. A memory access in this range will access ROM by asserting ROMCS0# or ROMCS1# and by deactivating the OBMEM output. If BIOS ROM shadowing is disabled, this memory space can be re-mapped above the top of the physical address space. See roll address mapping for details on re-mapping.

When this bit is set, BIOS ROM shadowing is enabled and memory accesses to this address range are made from local DRAM. During shadow DRAM accesses, the OBMEM signal is asserted and the ROMCS0# and ROMCS1# signals are disabled.

S640: This bit selects the base memory size. When cleared, a base memory of 512K is selected. When set, a base memory of 640K is selected. If a base memory of 512K is selected, the address range 080000H–09FFFFH can be re-mapped above the top of the physical address space. See roll address mapping for details on re-mapping.

DSIZE: This bit is used to indicate the type of DRAM installed. When cleared, it indicates 256K DRAM and when set, it indicates 1M DRAM installed. When 256K DRAM is installed, the multiplexed address line MA9 is not used.

INTERL: These two bits indicate the number of banks of memory installed. When more than one memory bank is installed, the banks are always interleaved.

ROMSIZE: This bit indicates the size of the installed ROM/EPROM. When cleared, it indicates 256K bit ROM/EPROM and when set, it indicates 512K bit ROM/EPROM is installed. This bit also affects the ROMCS0# and ROMCS1# address decode ranges when ROM shadowing is disabled. (See **Chip Select Signals** for further information.)

ENADP#: This bit is used to enable or disable shadowing of the adapter ROM area. If cleared, adapter ROM shadowing is disabled and accesses to the memory range 0C0000H–0DFFFFH will be made from ROM. If set, adapter ROM shadowing is enabled and memory accesses in this range will be from local DRAM. If adapter ROM shadowing is disabled, this memory space can be re-mapped above the top of the physical address space. See roll address mapping for details on re-mapping.

ENV#: This bit is used to enable or disable shadowing of the external video RAM. If cleared, video RAM shadowing is disabled and accesses to the memory range 0A0000H–0BFFFFH will be made from the system video RAM. If set, video RAM shadowing is enabled and memory accesses in this range will be from local DRAM. If video RAM shadowing is disabled, this memory space can be re-mapped above the top of the physical address space. See roll address mapping for details on re-mapping.

VRO: This bit selects either read/write access or read only access from the video RAM area when shadowing is selected. When video RAM shadowing is enabled and this bit is set, the local video RAM area will be read only, otherwise it will be available for both read and write access.

LOCK: This bit enables or disables external access to the configuration, roll compare, and address range compare registers. When this bit is cleared, the following conditions will exist:

- The configuration, roll compare, and address range compare registers will be read/writable at even I/O addresses from 22H-2EH
- The status outputs S0# and S1# will not be generated for these I/O addresses
- The shadowing DRAM area will be available for both reading and writing

When the LOCK bit is set, the following conditions will exist:

- The configuration, roll compare, and address range compare registers will not be accessible external to the 82335
- The status outputs S0# and S1# will be generated for I/O addresses between 22H–2EH
- The shadowing DRAM area will be read only with the exception of 0A0000H–0BFFFFH if the VR0 bit is cleared.

Once the lock bit is set, the configuration, roll compare, and address compare registers can only be accessed again by resetting the system. It is recommended that the LOCK bit be set after the 82335 is properly configured.

The lock bit does not affect the operations of I/O Ports 061H or 0F0H. Writing to these ports writes to both the 82335 and the 82230.

When writing to the 82335 registers, all bits are written over. Care should be taken to insure that all bits of the data to be written are set for the intended operation.

2.3.4.2 Roll Compare Registers

There are two roll compare registers, RC1 and RC2, located at I/O addresses 24H and 26H respectively. These registers are used to re-map logical addresses above the physical address space into physical addresses. Figure 2.9 shows the bit functions in the roll compare registers.

Roll address mapping works as follows. The output of the roll comparator activates the address mapper and causes an address roll-over. This is accomplished by using the address bits A23–A17, and the three outputs from each roll compare register: compare enable, address mask, and compare data. Each time a new address is received by the 82335, it compares the address bits A23–A17 with the compare data C23–C17 using M23–M17 as a mask. For example, if M23–M17 = 1111110, the output of the comparison is true if A23–A18 is identical to C23–C18 with A17 being a don't care. If the comparison is disabled (EN = 0), then the output will always be false. The roll comparator performs an OR function of the comparisons of address input with the two roll registers. The following examples illustrate programming of the roll compare registers.

Example #1

— 4 Banks of 256K x 1 DRAM Installed

— S640 = 0, ROMEN# = 1, ENADP# = 0, ENV# = 0 (384K Roll-Over Avail.)

— Top of Physical Address Space = 1FFFFFH (2M)

— Roll Decode Range = 200000H – 25FFFFH (384K)

Register	EN	C23–C17	M23–M17	Address Range
RC1	1	001000x	111111x	200000H–23FFFFH (256K)
RC2	1	0010010	1111111	240000H–25FFFFH (128K)

In this example, both roll compare registers are required to decode the 384K roll address area.

Example #2

— 3 Banks of 1M x 1 DRAM Installed

— S640 = 1, ROMEN# = 1, ENADP# = 0, ENV# = 0 (256K Roll-Over Avail.)

— Top of Physical Address Space = 5FFFFFH (6M)

— Roll Decode Range = 600000H–63FFFFH (256K)

Register	EN	C23–C17	M23–M17	Address Range
RC1	1	011000x	111111x	600000H–63FFFFH (256K)
RC2	0	xxxxxxx	xxxxxxx	None

In this example, the 256K roll address area can be decoded with only one register. Therefore, the other register has been disabled.

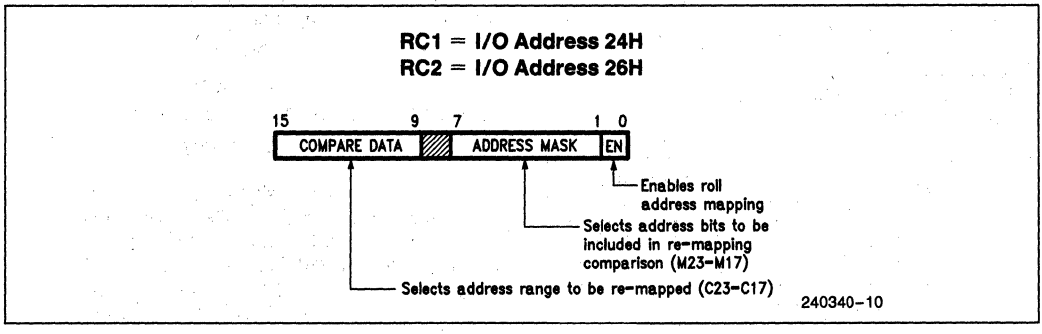


Figure 2.9. Bit Functions of the Roll Compare Registers

2.3.4.3 Address Range Compare Registers

There are two address range compare registers, CC0 and CC1, located at I/O addresses 28H and 2AH respectively. They are used to decode the on-board memory address range. Figure 2.10 shows the bit functions of the address range compare registers. Upon reset, register CC0 will be enabled and register CC1 will be disabled.

The on-board memory address range is decoded in a similar manner as the roll-over address range. Each address range comparator accepts address inputs A23–A19, compare data C23–C19, mask data M23–M19, and an enable bit EN. Each comparator compares A23–A19 with C23–C19 using M23–M19 as a mask. For example, if M23–M19 = 00001, the output of the comparison is true if A19 is identical to C19 with A23–A20 being don't cares. Comparison is disabled if EN = 0, yielding a false regardless of the address input.

Examples of address range compare register programming are shown below.

Example #1
Memory Installed: One 16-Bit Bank of 256K x 1 DRAM (512K)

Register	EN	C23–C19	M23–M19	Address Range
CC0	1	00000	11111	000000H–07FFFFH (512K)
CC1	0	xxxxx	xxxxx	None

Only one address range compare register is required to decode a 512K address space.

Example #2
Memory Installed: Three 16-Bit Banks of 1M x 1 DRAM (6M)

Register	EN	C23–C19	M23–M19	Address Range
CC0	1	00xxx	11000	000000H–3FFFFFH (4M)
CC1	1	010xx	11100	400000H–5FFFFFH (2M)

In this example, two address range compare registers are required to decode the 6 megabyte address space.

2.3.5 CHIP SELECT SIGNALS

The address mapper/decoder uses the configuration, roll compare, and address range compare register contents along with input addresses to generate the following output signals:

ROMCS0# — ROM 0 Chip Select

ROMCS1# — ROM 1 Chip Select

LMEGCS — Lower Meg Chip Select

OBMEM — On-Board Memory Address Range

The ROM chip select signals are functions of the ROMSIZE, and ROMEN# bits in the configuration register as well as the input address. If ROM shadowing is enabled (ROMEN# = 1), then the ROM chip select outputs will be disabled. If ROM shadowing is disabled, then the ROM chip select outputs will be activated as follows:

If ROMSIZE = 0 (256K ROM)

ROMCS0# decodes the address ranges 0E0000H–OEFFFFH and FE0000H—FEFFFFH.

ROMCS1# decodes the address ranges 0F0000H–OFFFFFH and FF0000H—FFFFFFH.

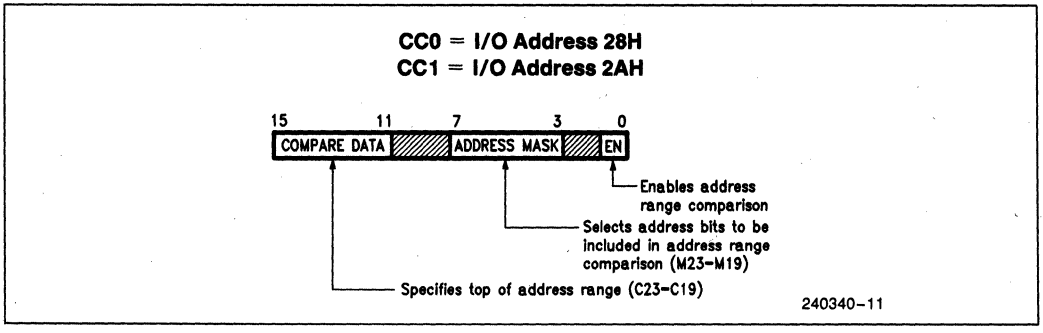


Figure 2.10. Bit Functions of the Address Range Compare Registers

If ROMSIZE = 1 (512K ROM)

ROMCS0# decodes the address ranges 0E0000H–0FFFFFH and FE0000H–FFFFFFH.

ROMCS1# is inactive.

The lower meg chip select output (LMEGCS) is a function of both the input address and M/IO# inputs. It is activated whenever a memory address within the first megabyte of memory is decoded. It is inactive during I/O cycles.

The on-board memory output (OBMEM) is a function of the address range comparators, roll comparators, and the bits ROMEN#, ROMSIZE, DSIZE, and S640 in the configuration register. It is used to differentiate local DRAM access from system RAM, ROM, or I/O accesses. When system memory is being addressed, the OBMEM output is inactive and the NA# output is de-activated to extend the 386SX processor address long enough to be latched onto the system address bus.

2.4 Ready Generator

The 82335 indicates completion of the current bus cycle to the 386SX microprocessor via the READYSX# signal. The ready generator determines the appropriate number of wait states (if any) to insert, and activates the READYSX# output at the correct time. Table 2.1 in the DRAM control section shows the number of wait states during local memory accesses for each mode.

The READY generator has three external inputs: READY286#, READYNPX#, and EXTRDY. The READY286# input is driven by the 82230 READY# pin and is used to identify the completion of system bus cycles. Completion of math coprocessor bus cycles is indicated by the READYNPX# input. This input is usually tied directly to the 80387SX READYO# pin. The EXTRDY input is an active high, level-triggered input which directly gates READYSX#. The READYSX# output is held inactive until the EXTRDY input is sampled active. It is used to extend bus cycles when using slow peripherals or off-board memory. Setup and hold times for these inputs must be met to guarantee correct operation.

2.5 Bus Cycle Translator

The 82335 has a built in interface unit that translates 386SX processor control signals to 80286 control signals. This bus cycle translator identifies the bus cycle being performed, monitors the CPU T-states, and outputs 80286-like bus control signals to the 82230/82231 and other components in a PC/AT system. It also receives 80286 control inputs and translates them into 386SX processor compatible signals when required.

As the bus cycle translator monitors control inputs from both the 386SX processor and the 82231, it determines what type of cycle is being requested. If one of the following cycles is being requested:

1. I/O Access
2. System Memory Access
3. Halt/Shutdown
4. Interrupt

then the bus tracker monitors the timing of the bus cycle and simultaneously outputs 80286-type bus control signals. The control signals output are S0#, S1#, and M/IO286#. The S0# and S1# outputs are not activated for local memory accesses, or for accesses to the 82335 on-chip I/O (programming of the on-chip registers).

In addition to controlling status output to the 82230/82231, the bus cycle translator also controls the hold request (HRQSX) input to the 386SX processor. A hold request signal coming from the 82230 (HRQ286) is translated into a 386SX processor-compatible output and driven to the 386SX processor. The 386SX processor responds with a hold acknowledge (HLDASX) to the 82335 which then translates that to a HLDA output to the 82230/82231.

2.6 Math Coprocessor Interface

The 82335 provides synchronous interface signals to allow the 80387SX coprocessor to run in a PC/AT system with proper error handling. It also has logic built in to automatically sense when an 80387SX is installed. If a 80387SX coprocessor is not present,

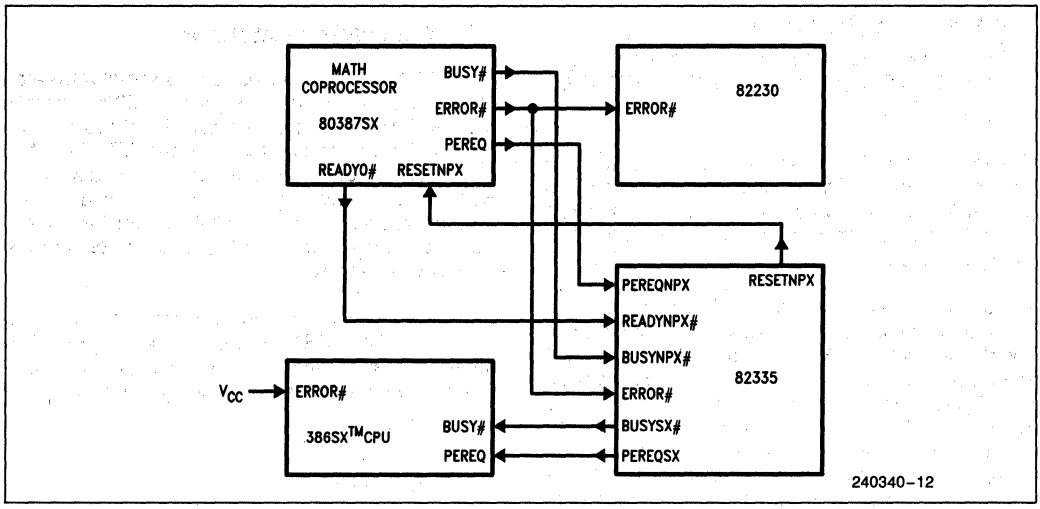


Figure 2.11. Math Coprocessor Interface

BUSYSX# is toggled by REFRESH# and PEREQSX is forced low. Otherwise BUSYSX# and PEREQSX function normally. If a 80387SX coprocessor is not installed, the READYNPX# input should be pulled up and PEREQNPX should be grounded. A diagram of the coprocessor interface is shown in Figure 2.11.

If a 80387SX coprocessor is installed and a numeric exception occurs, the following sequence will occur. BUSYSX# is latched low and PEREQSX is forced high. This holds processing on the 386SX processor while completing the 80387SX coprocessor transfers. The ERROR# output from the 80387SX coprocessor becomes active causing the 82230 to issue an interupt request on IRQ13. The interrupt handler performs an IOWR cycle to address 0xF0 which clears the BUSY latching hardware. The interrupt handler then clears the numerics ERRORNPX signal and normal operation resumes.

2.7 Clock Generator/Reset Synchronizer

The 82335 clock generator is used to synchronize the CPU, coprocessor, and peripherals by converting an input frequency into the system clock outputs CLK2 and PCLK#. The input frequency must be provided by a 32 MHz external oscillator connected to the 82335 EFI pin. This EFI input is internally buffered and output to the 386SX processor and 80387SX coprocessor via the CLK2 output. It is also divided by two (to 16 MHz) and output to the 82230/82231 using the PCLK# output.

The reset synchronizer receives the RESETCPU and SYSRESET inputs from the 82230 and generates the synchronous outputs RESETSX and RESETNPX to reset either the processor or the entire system. When the SYSRESET signal is activated, both the RESETSX and RESETNPX outputs are asserted to reset the system. Activating the RESETCPU input will only assert the RESETSX output to reset the 386SX processor. Both RESETCPU and SYSRESET can be asynchronous inputs.

2.8 Parity Generator/Checker

The 82335 has a built in parity generator and checker to maintain data integrity for the local memory. This parity generator/checker has local data bus input/outputs D0–D15, and two parity input/output pins, parity high byte (PARH) and parity low byte (PARL). PARH and PARL are three-state input/output pins which are designed to directly drive the DRAMs without data transceivers.

During memory write cycles, the D0–D15 pins are evaluated by the internal parity generator and the parity bits PARH and PARL are output to the DRAM. During memory read cycles, the data from the DRAMs D0–D15 are combined with the DRAM parity outputs and checked for parity errors. If a parity error is detected, the parity error pin (PERROR#) is asserted.

A parity register bit is provided within the 82335 for resetting the PERROR# output. Bit 2 of the I/O address 61H is the PARCHKEN (parity check enable) bit. When programmed to "0", parity checking is enabled. Otherwise, parity checking is disabled. This is a write-only register and cannot be read.

2.9 General System Considerations

1. The RAS0#-RAS3#, CASH0#-CASH3#, and CASL0#-CASL3# output buffers are designed to directly drive the heavy capacitive loads of the dynamic RAM arrays. To keep the RAM driver outputs from ringing excessively in the system environment it is necessary to match the output impedance with the RAM array by using series resistors. Each application may have different impedance characteristics and may require different series resistance values. The series resistance values should be determined for each application.
2. If the capacitive loading on the MA0-MA9 outputs exceeds the maximum capacitive loading specification (see AC DRAM Timing Specifications), then buffering of the MA0-MA9 outputs is recommended. The MA0-MA9 outputs can directly drive approximately two megabytes of memory.
3. The NA# pin on the 82335 must be connected to NA# on the 386SX processor.
4. If a math coprocessor is not installed, the READYNPX# pin should be pulled high and the PEREQNPX pin should be grounded to avoid extraneous processor extension requests.
5. If there is no DRAM installed in the physical address space 080000H-0FFFFFH, then shadowing and roll address mapping must be disabled.
6. When setting the LOCK bit in the configuration register, the entire contents of the configuration register must be written. Writing to a register in the 82335 will overwrite all bits in that register.

3.0 MECHANICAL DATA

3.1 Package Dimensions

The 82335 is available in a 132 lead plastic quad flat pack (PQFP) package. Table 3.1 and Figures 3.1-3.5 show the physical dimensions of this package.

Table 3.1. Intel Case Outline Dimensions for 132 Lead Plastic Quad Flat Pack 0.025 Inch Pitch

Symbol	Description	Inch		mm	
		Min	Max	Min	Max
A	Package Height	0.160	0.170	4.06	4.32
A1	Standoff	0.020	0.030	0.51	0.76
D, E	Terminal Dimension	1.075	1.085	27.31	27.56
D1, E1	Package Body	0.947	0.953	24.05	24.21
D2, E2	Bumper Distance	1.097	1.103	27.86	28.02
D3, E3	Lead Dimension	0.800 REF		20.32 REF	
L1	Foot Length	0.020	0.030	0.51	0.76
Issue	IWS Preliminary 1/15/87				

Symbol List

Letter or Symbol	Description of Dimensions
A	Package Height: Distance from Seating Plane to Highest Point of Body
A1	Standoff: Distance from Seating Plane to Base Plane
D/E	Overall Package Dimension: Lead Tip to Lead Tip
D1/E1	Plastic Body Dimension
D2/E2	Bumper Distance
D3/E3	Footprint
L1	Foot Length

NOTES:
1. All dimensions and tolerances conform to ANSI Y14.5M-1982.
2. Datum plane H located at the mold parting line and coincident with the bottom of the lead where lead exits plastic body.
3. Datums A B and D to be determined where center leads exit plastic body at datum plane H.
4. Controlling Dimension, Inch.
5. Dimensions D1, D2, E1, and E2 are measured at the mold parting line and do not include mold protrusion. Allowable mold protrusion is 0.18 mm (0.007 in) per side.
6. Pin 1 identifier is located within one of the two zones indicated.

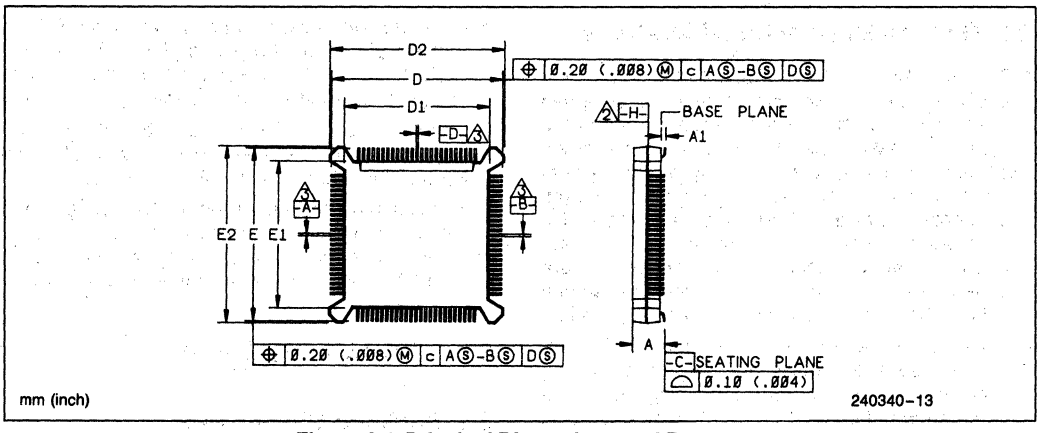


Figure 3.1. Principal Dimensions and Datums

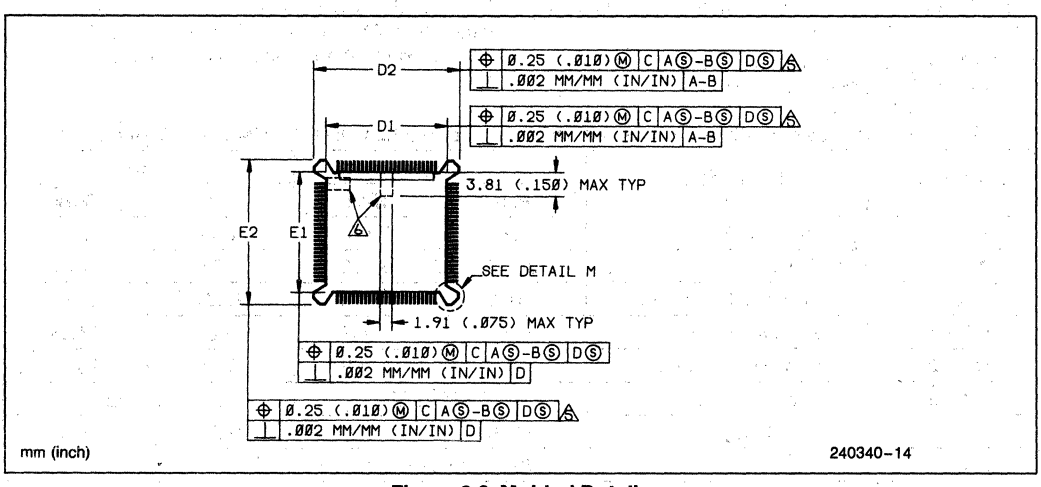


Figure 3.2. Molded Details

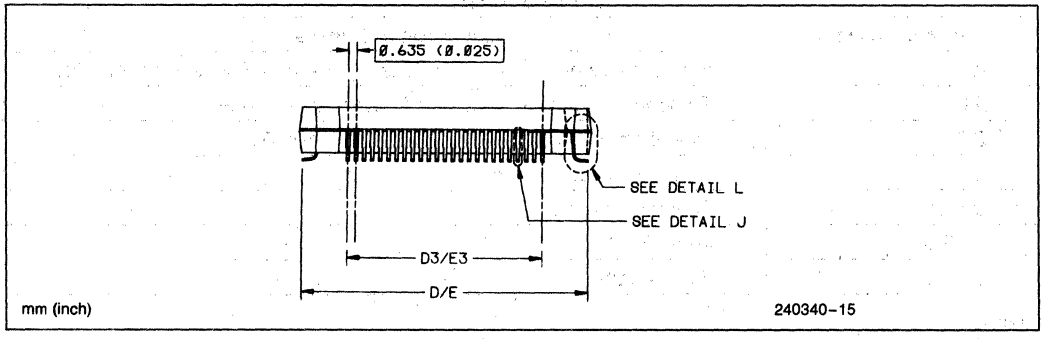


Figure 3.3. Terminal Details

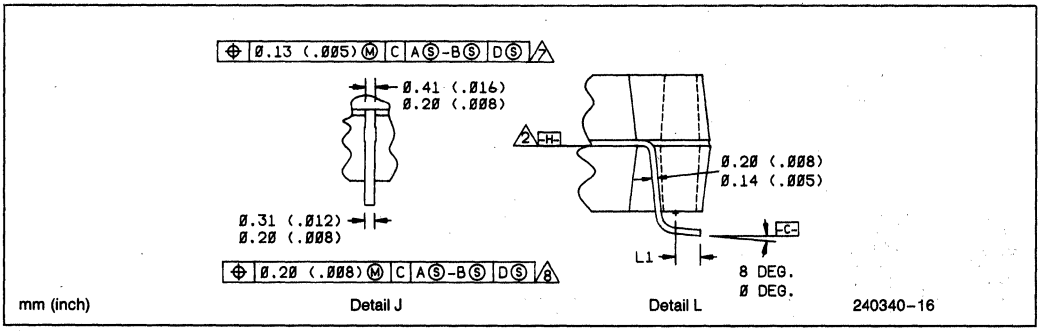


Figure 3.4. Typical Lead

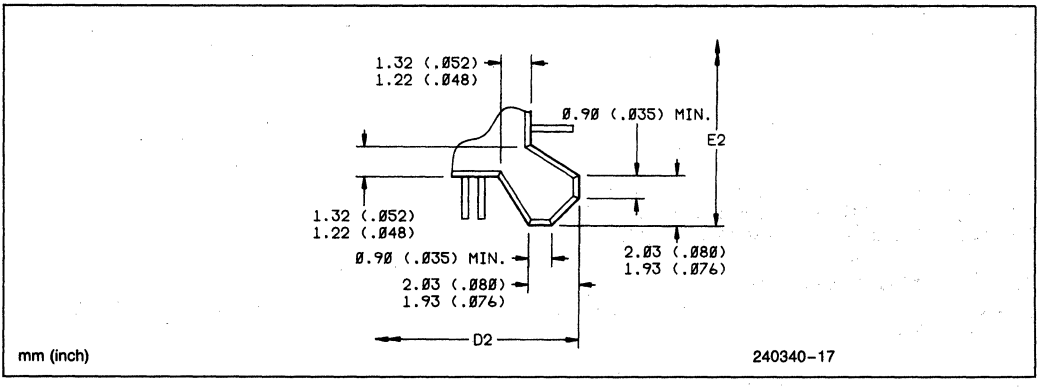


Figure 3.5. Detail M

3.2 Package Thermal Specifications

The 82335 is specified for operation when the case temperature is within the range of 0°C–85°C. The case temperature may be measured in any environment to determine whether the 82335 is within the specified range of operation. The case temperature should be measured at the center of the top surface opposite the pins. Table 3.2 shows the thermal resistance for this package.

Table 3.2. Thermal Resistances (°C/Watt)

θ Junction to Case	12
θ Case to Ambient	40

4.0 ELECTRICAL DATA

4.1 D.C. Electrical Specifications

Functional Operating Range: V_{CC} = 4.5V to 5.5V, T_{CASE} = 0°C to +85°C

Symbol	Parameter	Min	Max	Unit	Notes
V_{IL}	Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{ILC}	EFI Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IHC}	EFI Input High Voltage	V_{CC}−0.8	V_{CC}+0.3	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 1 mA
I_{LI}	Input Leakage Current for All Pins except BUSYNPX# and ERROR#		±15	μA	0 < V_{IN} < V_{CC}
I_{IL}	Input Sustaining Current (BUSYNPX# and ERROR# pins)		−400	μA	$\|V_{IL}$ = 0.45V, (Note 2)
I_{LO}	3-State Output Leakage Current		±15	μA	V_{OL} < V_{IN} < V_{CC}
I_{CC}	Supply Current		150	mA	(Note 3)

NOTES:
1. The min value, −0.3, is not tested.
2. BUSYNPX# and ERROR# inputs each have an internal pullup resistor.
3. CLK2 = 32 MHz, maximum loading on DRAM address and control pins.

4.2 A.C. Specifications

Unless otherwise specified, all timings are referenced at V = 1.5V, all timing values are in nano-seconds, and all voltages are in volts.

A.C. Timings for Clocks

Symbol	Parameter	Figure	Min	Max	Notes
	Operating Frequency			16	MHz (Note 1)
T1	EFI Period	4.1	31		
T2a	EFI High Time	4.1	12		at 2V
T2b	EFI High Time	4.1	5		at (V_{CC} − 0.8)
T3a	EFI Low Time	4.1	12		at 2V
T3b	EFI Low Time	4.1	7		at 0.8V
T4	EFI Fall Time	4.1		8	(V_{CC} − 0.8) to 0.8 (Note 2)
T5	EFI Rise Time	4.1		8	0.8 to (V_{CC} − 0.8) (Note 2)
T6	CLK2 Period	4.1	31		
T7a	CLK2 High Time	4.1	9		at 2V
T7b	CLK2 High Time	4.1	5		at (V_{CC} − 0.8)
T8a	CLK2 Low Time	4.1	9		at 2V
T8b	CLK2 Low Time	4.1	7		at 0.8V
T9	CLK2 Fall Time	4.1		8	(V_{CC} − 0.8) to 0.8 (Note 2)
T10	CLK2 Rise Time	4.1		8	0.8 to (V_{CC} − 0.8) (Note 2)
T11	PCLK# Delay from CLK2	4.1		8	

NOTES:
1. This device is only guaranteed for 16 MHz operation.
2. These are not tested. They are guaranteed by design characterization.

Output Loadings: CLK2 75 pF Max.
PCLK# 75 pF Max.

A.C. Timings for DRAM Controller Unit

Symbol	Parameter	Figure	Min	Max	Unit	Notes
T12	RAS# Active Delay from CLK2	4.3		55	ns	
T13	RAS# Inactive Delay from CLK2	4.3	8		ns	
T14	Row Addr Setup to RAS# Active	4.3	20		ns	
T15	Row Addr Hold from RAS# Active	4.3	20		ns	
T16	RAS# Precharge Time	4.5	92		ns	
T17	Col Addr Setup to CAS# Active	4.3	25		ns	
T18	Col Addr Hold from CAS# Active	4.3	25		ns	
T19a	CAS# Active Delay from CLK2	4.3		55	ns	(Note 1)
T19b	CAS# Active Delay from CLK2 (Read)	4.6		43	ns	(Note 2)
T19c	CAS# Active Delay from CLK2 (Write)	4.4		39	ns	(Note 3)
T20	CAS# Inactive Delay from CLK2	4.3	8		ns	
T21a	CAS# Active Pulse Width	4.4	37		ns	(Note 4)
T21b	CAS# Active Pulse Width	4.6	62		ns	(Note 5)
T23a	CAS# Precharge Pulse Width	4.4	22		ns	(Note 4)
T23b	CAS# Precharge Pulse Width	4.6	52		ns	(Note 5)
T24	Read Data Setup before CLK2	4.4	11		ns	(Note 6)
T25	Read Data Hold from CAS# Inactive	4.4	2		ns	
T26	Write Data Delay from CLK2	4.4		9	ns	(Note 7)
T27	Write Data Hold after CLK2	4.4	0		ns	
T28	PARL/PARH Setup to CAS# Active	4.4	0		ns	
T29	PARL/PARH Hold from CAS# Active	4.4	28		ns	
T30	DEN# Active from CLK2	4.4	0	20	ns	
T31	DEN# Inactive from CLK2	4.4	8		ns	
T32	DIR Delay from CLK2	4.4	25	55	ns	
T33	WE# Setup to CAS# Active	4.6	5		ns	
T34	386SX CPU Addr, BHE#, BLE# Setup to CLK2	4.3	26		ns	
T79	RAS# Active Delay from MEMx#	4.10		36	ns	(Note 8)
T80	RAS# Pulse Width	4.10	150		ns	(Note 8)
T81	CAS# Inactive from Memx#	4.10	9	40	ns	(Note 8)
T82	CAS# Active Delay from RAS# Active	4.10	55	100	ns	(Note 8)
T83	Refresh Pulse Width	4.10	375		ns	

NOTES:
1. All cycle except a) F1/F4 page-mode write, b) W01/02 page-hit-bank-miss.
2. Page-hit-bank-miss read cycles in W01/W02.
3. Page write cycles in F1/F4, or page-hit-bank-miss write cycles in W01/W02.
4. F1/F4 mode only.
5. W01/W02 mode only.
6. For parity checker.
7. To guarantee PARL, PARH timings.
8. DMA/MASTER mode timings.

Output Loadings: RAS#: 125 pF to 250 pF; CAS#: 75 pF to 150 pF; WE#: 100 pF to 500 pF; MA, DEN#, DIR: 25 pF to 75 pF

Other A.C. Timings

Symbol	Parameter	Figure	Min	Max	Notes
T40	W/R#, M/IO#, D/C#, ADS# Setup to CLK2	4.3	29		
T41	RESETSX Valid before CLK2	4.2	15		C_{Load} = 30 pF
T42	RESETSX Hold after CLK2	4.2	6		C_{Load} = 30 pF
T43	READYSX# Valid before CLK2	4.11	20		C_{Load} = 30 pF
T44	READYSX# Hold after CLK2	4.11	6		C_{Load} = 30 pF
T45	NA# Valid before CLK2	4.3	7		C_{Load} = 30 pF
T46	NA# Hold after CLK2	4.3	27		C_{Load} = 30 pF
T47	HRQSX Valid before CLK2	4.12	28		C_{Load} = 30 pF
T48	HRQSX Hold after CLK2	4.12	7		C_{Load} = 30 pF
T54	READYNPX# Setup to CLK2	4.11	31		
T55	READYNPX# Hold from CLK2	4.11	0		
T57	READY286# Setup to PCLK#	4.11	15	90	
T58	READY286# Hold from PCLK#	4.11	10		
T59	RESETCPU Setup to CLK2	4.2	15		(Note 1)
T60	RESETCPU Hold from CLK2	4.2	15		(Note 1)
T61	A20GATE Setup to CLK2	4.12	40		(Note 1)
T62	A20GATE Hold from CLK2	4.12	15		(Note 1)
T63	HRQ286 Setup to CLK2	4.12	50		
T64	HRQ286 Hold from CLK2	4.12	5		
T66	MEMR#, MEMW#, REFRESH#, TURBO# Setup to CLK2	4.12	50		(Note 1)
T67	MEMR#, MEMW#, REFRESH#, TURBO# Hold from CLK2	4.12	5		(Note 1)
T68	EXTRDY Setup to CLK2	4.11	50		
T69	EXTRDY Hold from CLK2	4.11	6		
T70	M/IO286#, S0# and S1# Output Delay from PCLK#	4.12	5	25	C_{Load} = 100 pF
T71	ROMCS#, LMEGCS# Output Delay from Valid Address	4.3		40	C_{Load} = 50 pF
T72	OBMEM Output Delay from Valid Address	4.3		40	C_{Load} = 100 pF
T73	D0–D15 Output Delay form PCLK#	4.8	5	40	C_{Load} = 120 pF
T74	PERROR# Output Delay from PCLK#	4.4		50	C_{Load} = 50 pF
T75	HLDA Active Delay from HRQ286	4.10	92		
T76	HLDA Inactive Delay from HRQ286	4.10	185		
C_{IN}	Input Capacitance			10	pF (Note 2)
C_{OUT}	Output or I/O Capacitance			12	pF (Note 2)
C_{CLK}	CLK2 Capacitance			20	pF (Note 2)

NOTES:
1. Asynchronous parameters, provided to assure recognition at a specific clock edge.
2. Not tested. These are guaranteed by design characterization.

4.3 A.C. Timing Diagrams

The following diagrams illustrate AC timing relations.

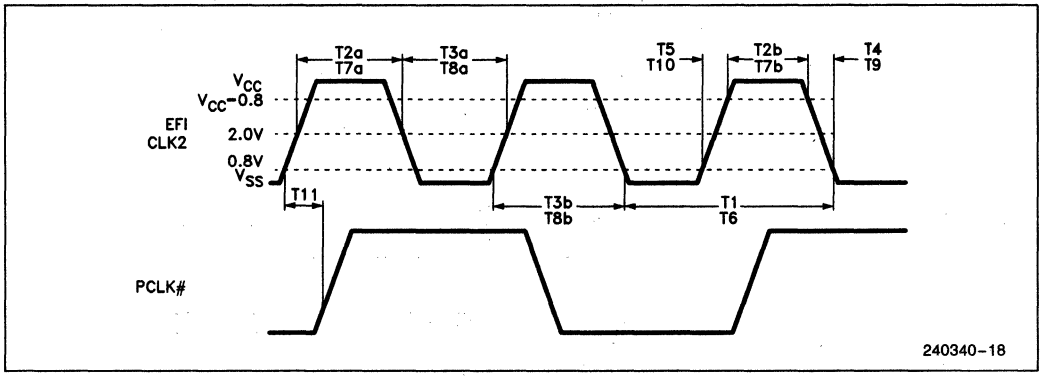


Figure 4.1. Clock Timings

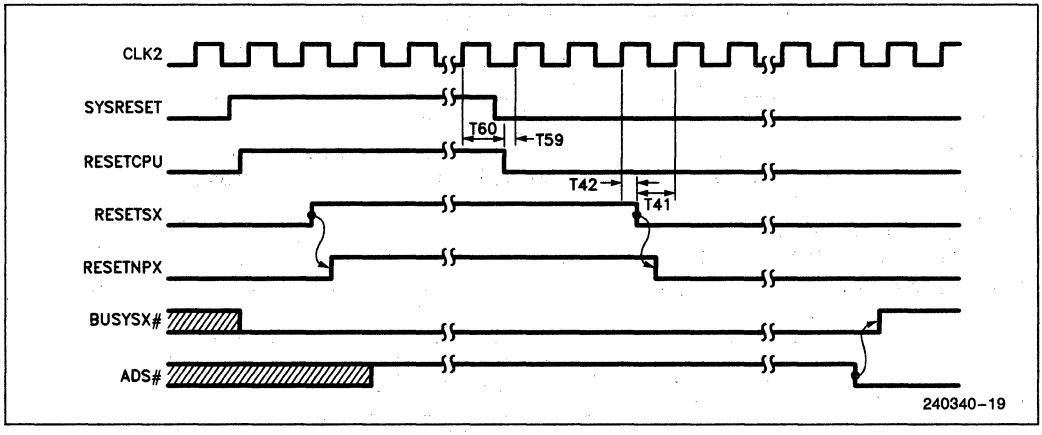


Figure 4.2a. Power-On Reset Sequence

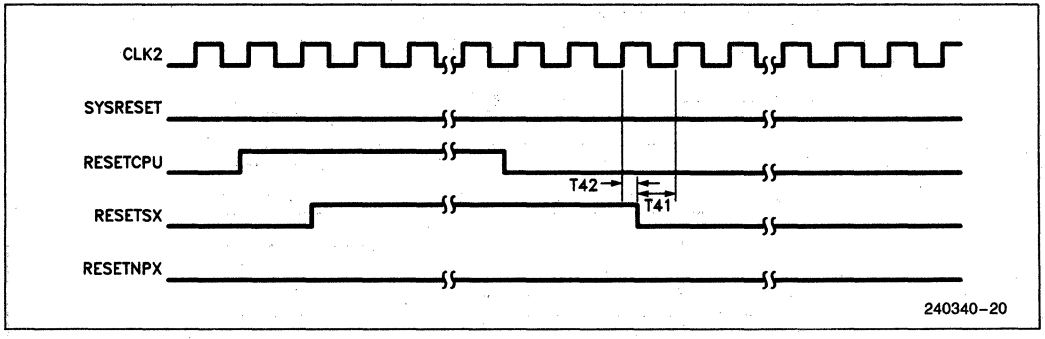


Figure 4.2b. RESETCPU Sequence

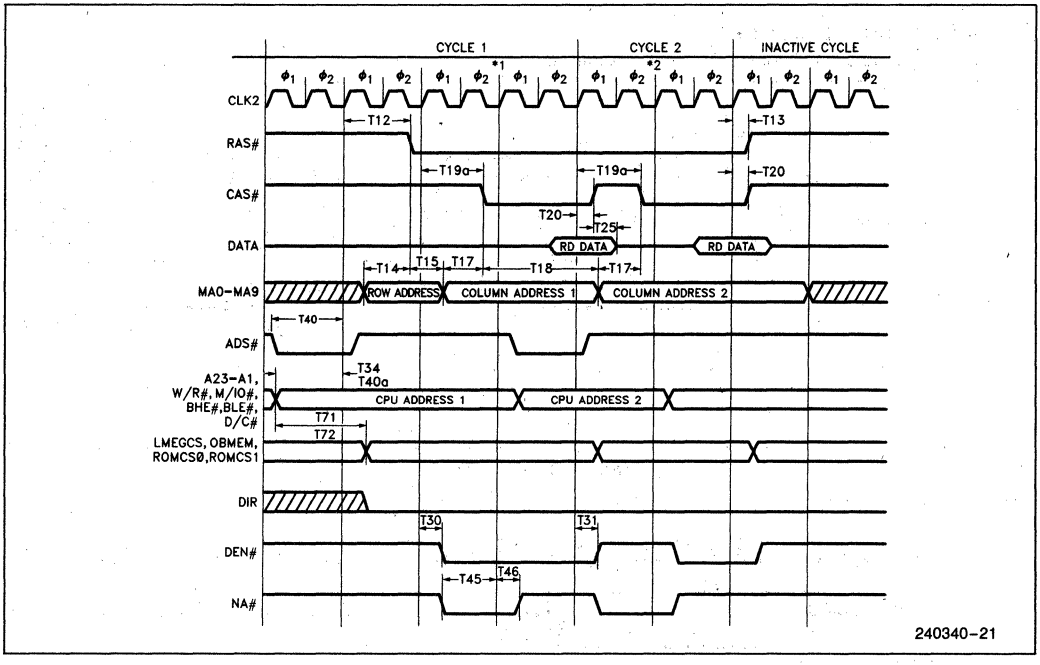


Figure 4.3. DRAM Cycles—Inactive . . . Read . . . Page Read . . . Inactive

NOTES:
1. Add 1 Wait State for W01 Mode.
2. Add 1 Wait State for W01 Mode, 2 Wait States for W02 Mode.

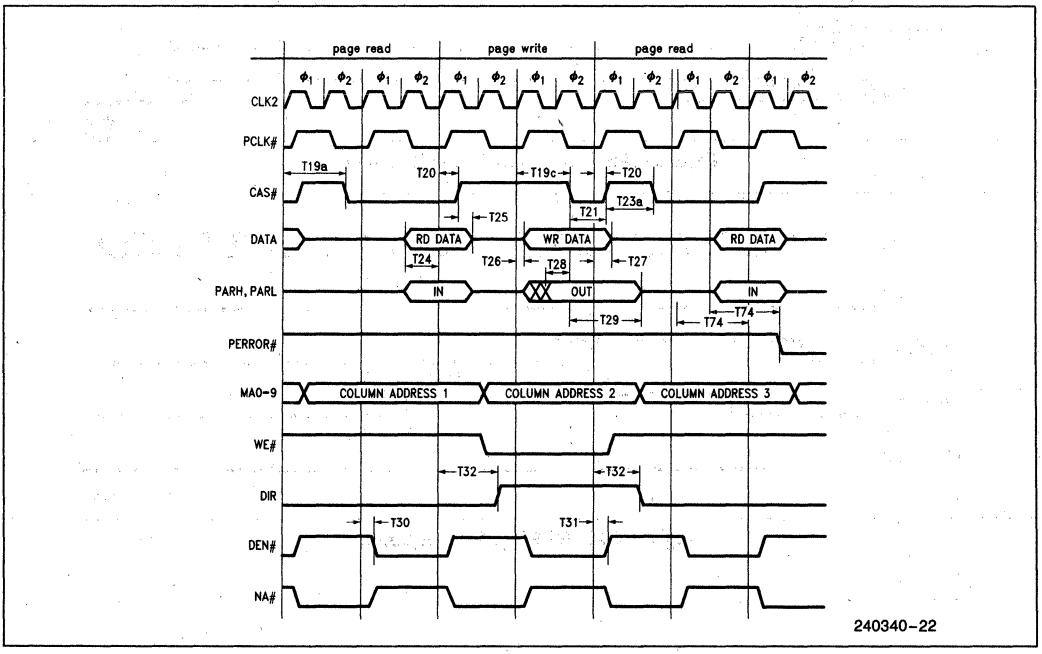


Figure 4.4. DRAM Cycles—Page Read . . . Page Write . . . Page Read F1/F4 Mode

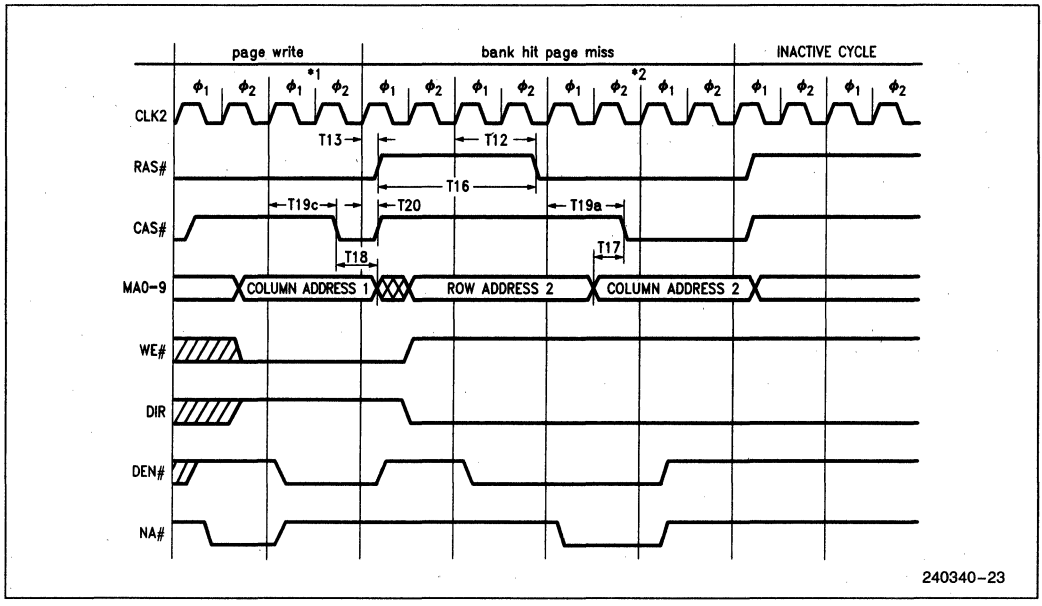


Figure 4.5. DRAM Cycles—Page Write . . . Bank-Hit-Page-Miss Read . . . Inactive

NOTES:
1. Add 1 Wait State for W01 Mode or 2 Wait States for W02 Mode.
2. Add 1 Wait State for W02 Mode.

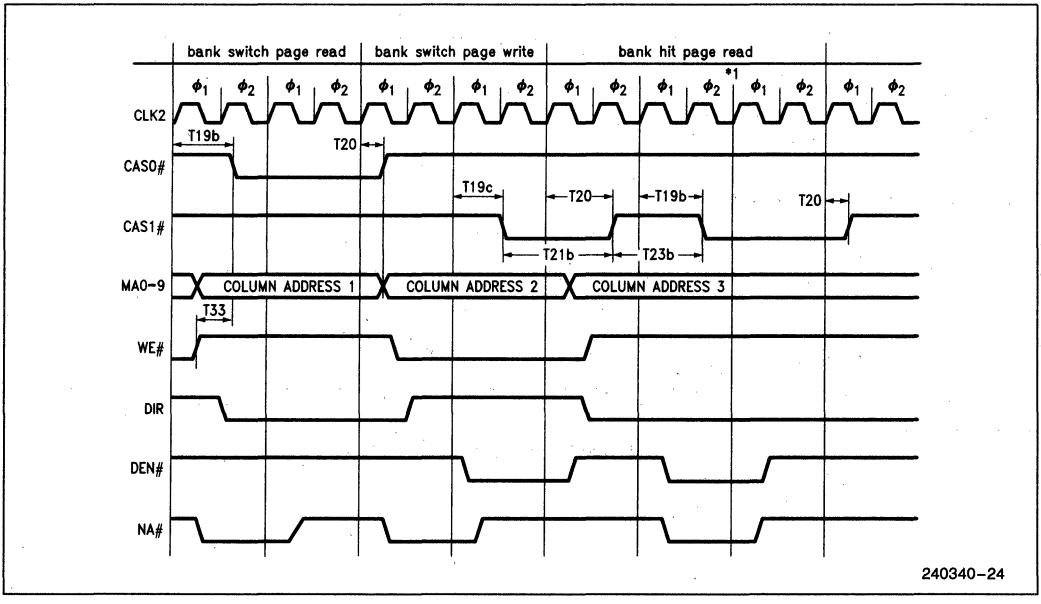


Figure 4.6. DRAM Cycles—Bank Switch Page Read . . . Bank Switch Page Write . . . Bank Hit Page Read for W01/W02 Mode

NOTE:
1. Add 1 Wait State for W02 Mode.

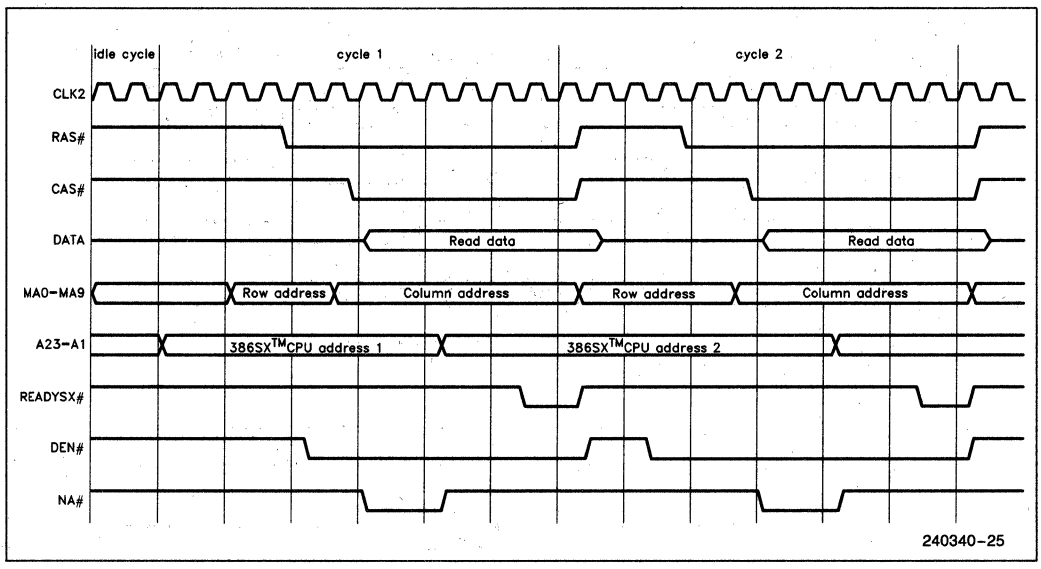


Figure 4.7. DRAM Cycles—Non-Turbo Mode Read

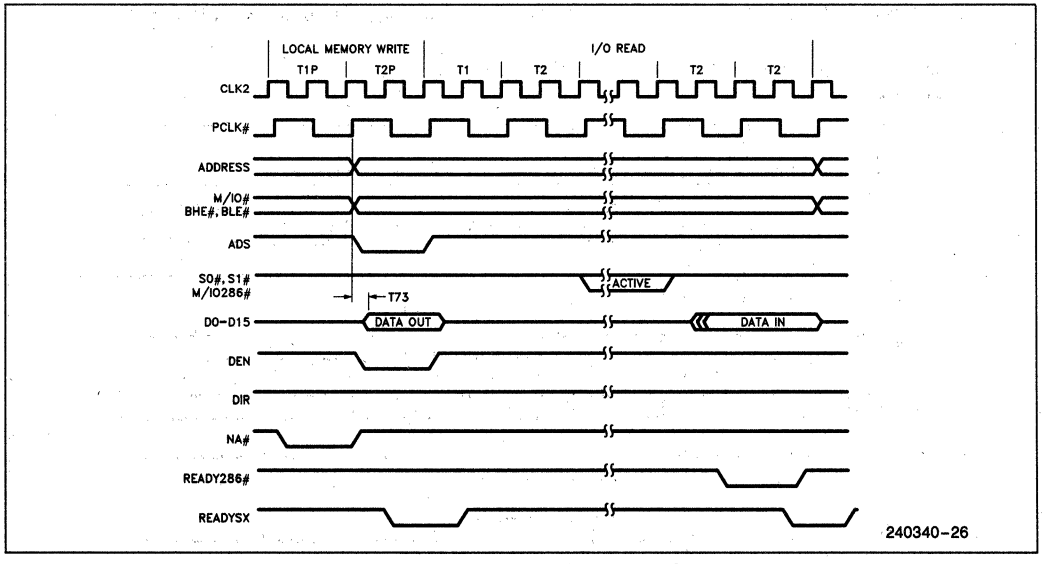


Figure 4.8. Local Memory Write . . . I/O Read

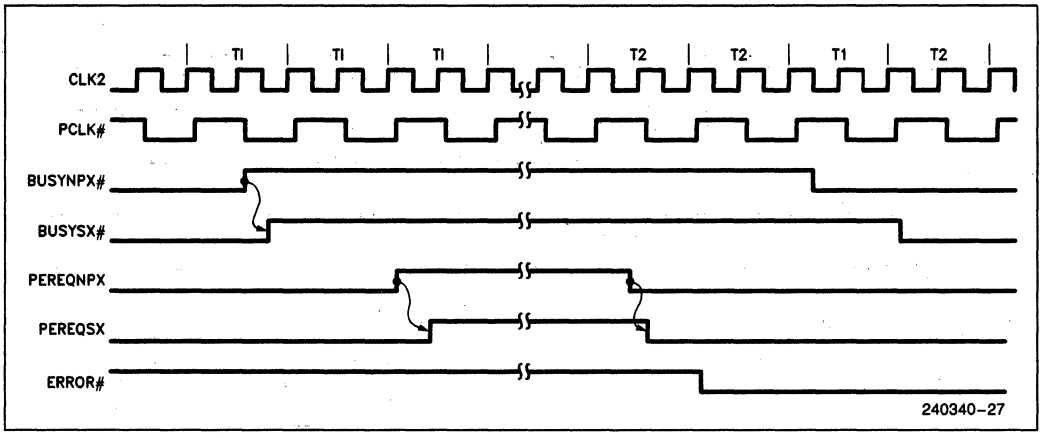


Figure 4.9. Numeric Coprocessor Interface Timing

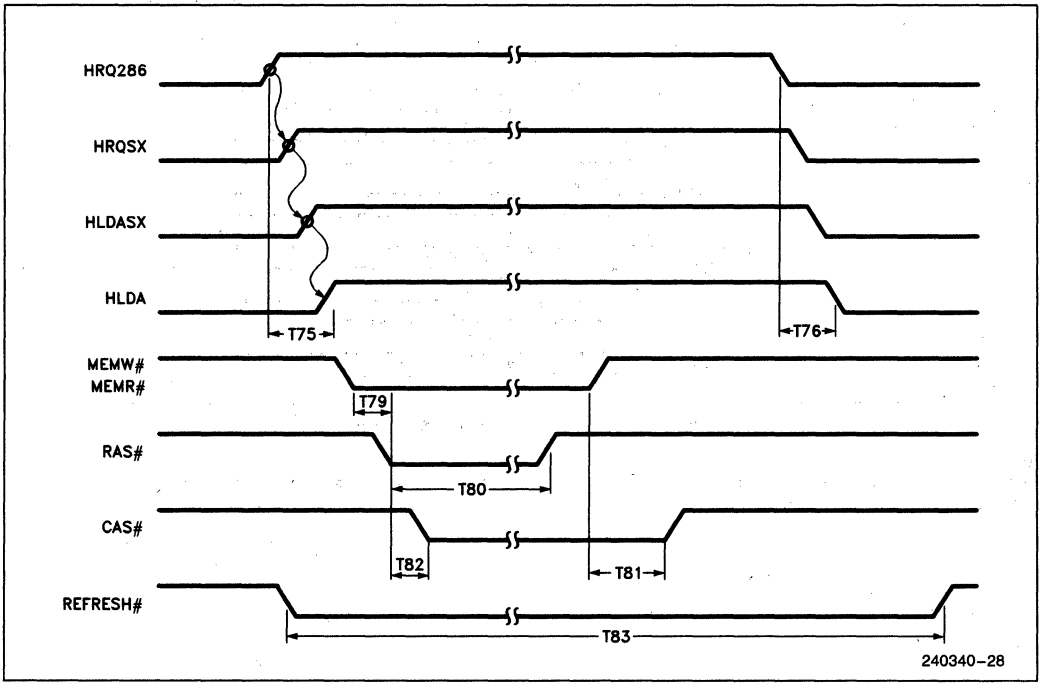


Figure 4.10. DMA/MASTER DRAM Access and Refresh Timings

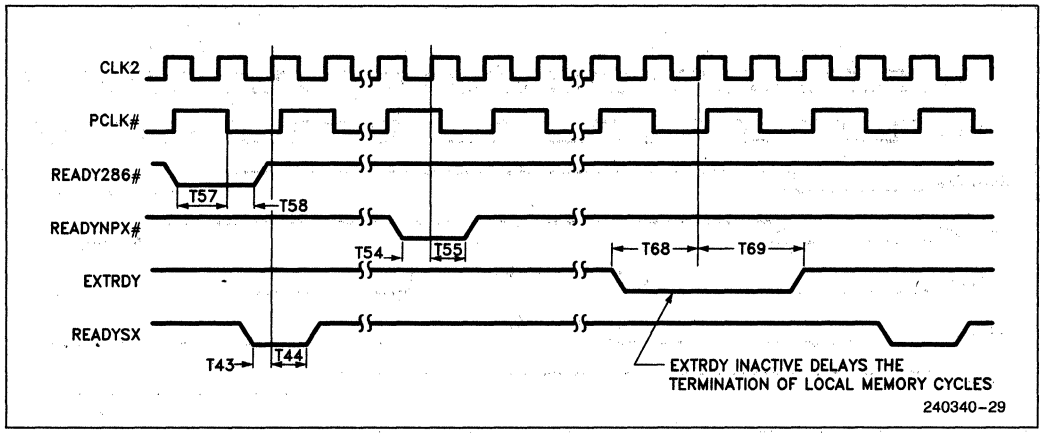


Figure 4.11. Ready Setup and Hold Timing

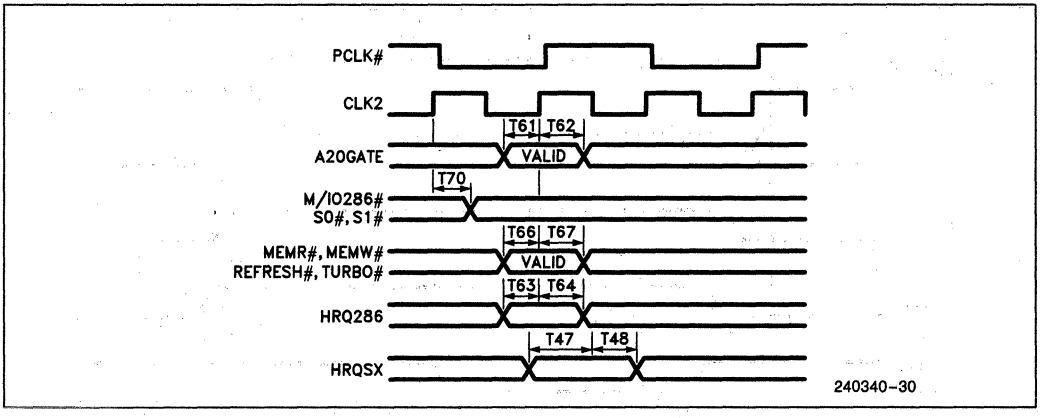


Figure 4.12. Misc. Setup and Hold Timings

NOTE:
The information contained in this data sheet is advance information. It is accurate at the time of printing, however, Intel reserves the right to make changes without notice.

82230/82231
HIGH INTEGRATION AT*-COMPATIBLE CHIP SET

- **Fully IBM PC-AT* System Compatible**
- **Two Chip Set Replaces the Major Logic Functions of the IBM PC-AT Motherboard Including the Functions of all the Microprocessor Peripherals:**
  - **— 8259A Programmable Interrupt Controller (Master)**
  - **— 8259A Programmable Interrupt Controller (Slave)**
  - **— 8254 Programmable Interval Timer**
  - **— 8284A Clock Generator**
  - **— 82284 Clock Generator & Ready Interface**
  - **— 82288 Bus Controller**
  - **— 8237 DMA Controller (2)**
  - **— 6818 Real Time Clock**
  - **— 74LS612 Memory Mapper**
- **Includes:**
  - **— Refresh Generation Logic**
  - **— Refresh/DMA Arbitration**
  - **— Address/Data Bus Control**
  - **— 16- to 8-Bit Conversion Logic**
- **Memory Refresh Controller Drives Up to 4 Mb DRAMs**
- **Numeric Processor Control Logic**
- **Up to 12 MHz System Clock Utilizing RAMs with Zero Wait States**
- **Single +5V Power Supply**
- **CHMOS Technology**
- **84 Pin PLCC Packages**
  (See Packaging Specification Order #231369-004)

The 82230 and 82231 are a two-chip implementation of LSI/MSI/SSI logic controlling the IBM Personal Computer AT. The devices provide a low power, highly integrated PC-AT design solution that may be applied to any 80286-based system.

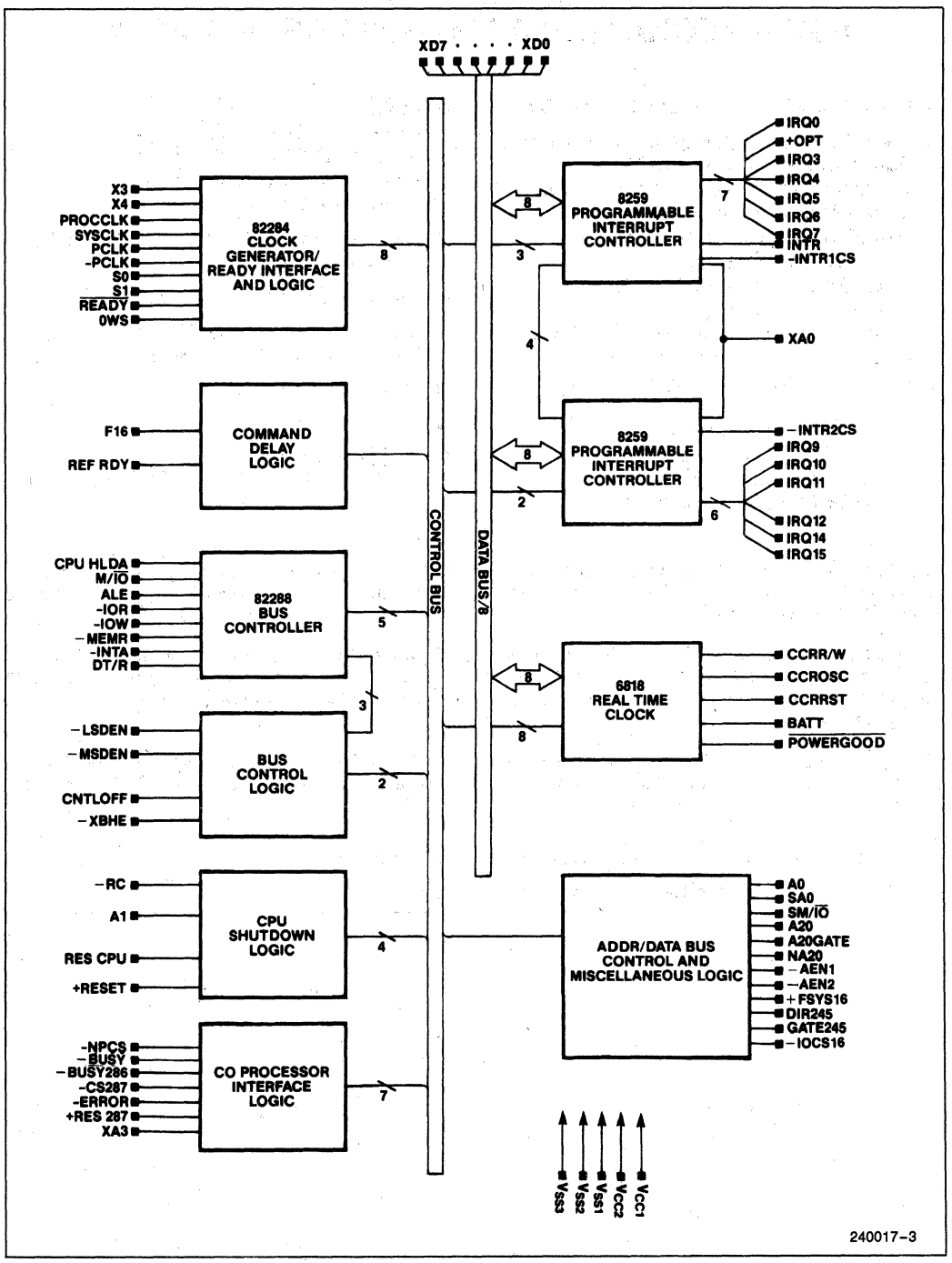
The 82230 performs the functions of the 82284 Clock Generator & Ready Interface, 82288 Bus Controller for 80286 processors, 6818 Real Time Clock/RAM, and the Master-Slave implementation of the dual 8259A Programmable Interrupt Controllers as well as Command Delay, Shut Down, Address/Data Bus Control and Ready Generation logic.

The 82231 includes the 8254 Programmable Interval Timer, 8284A Clock Generator, LS612 Memory Mapper and the dual 8237 DMA Controller functions as well as Refresh Generation and Refresh/DMA Arbitration Logic.

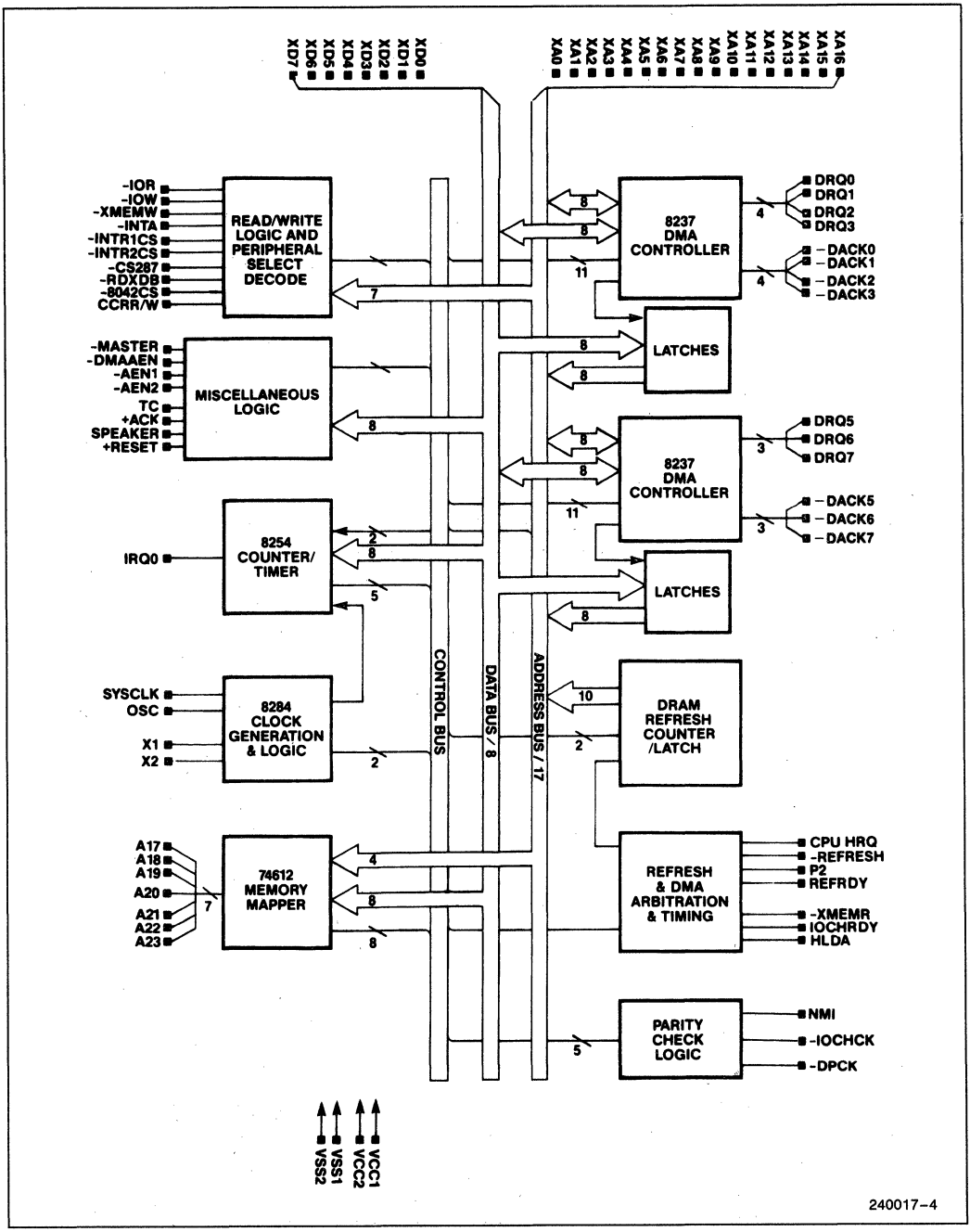
NOTE:
"+" and "−" in front of signal names is consistent with PC-AT Documentation.

*PC-AT is a Trademark of International Business Machine Corporation.

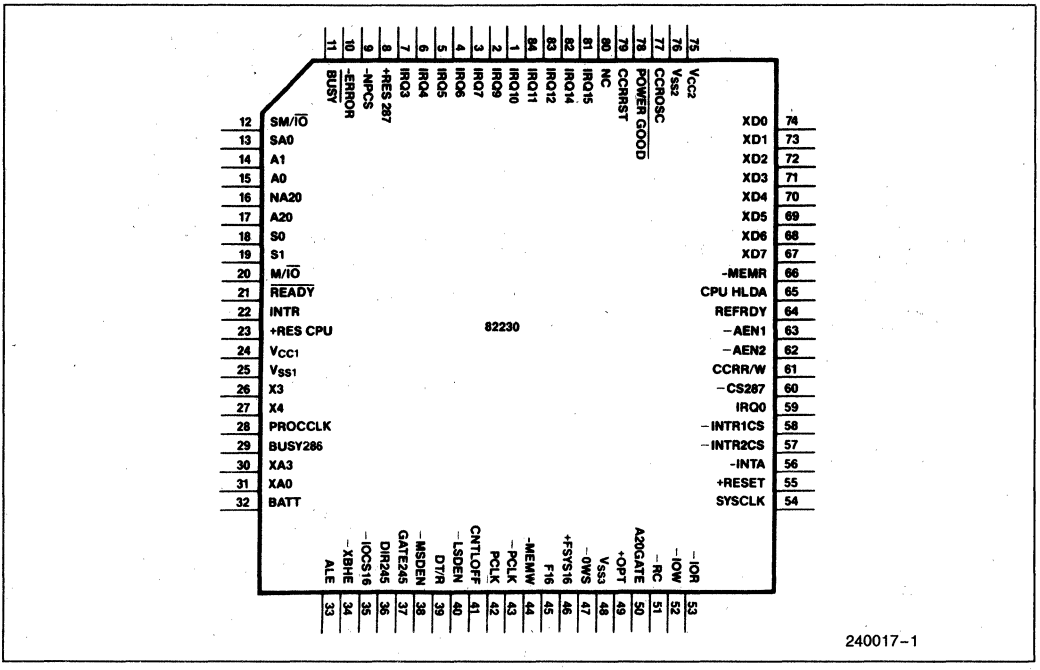
August 1988
Order Number: 240017-002



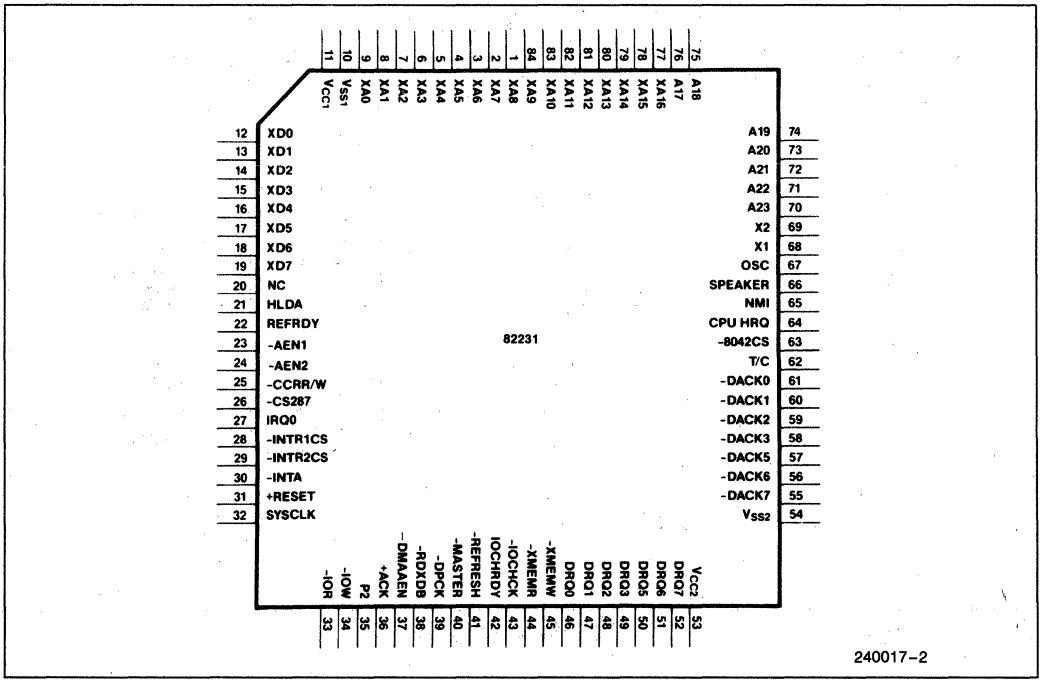
82230—Block Diagram



82231—Block Diagram



Pin Diagram 82230



Pin Diagram 82231

82230 PIN DESCRIPTION

Symbol	Pin No.	Type	Description
A0	15	I	ADDRESS 0 input from the CPU. It is used to generate SA0.
A1	14	I	ADDRESS 1 input from the CPU. It is used in conjunction with M/$\overline{IO}$, S0 and S1 to detect a CPU Shutdown condition.
A20	17	O	ADDRESS 20 is the A20 (NA20) line from the CPU after conditioning by the A20GATE signal. During a CPU Hold A20 goes to a high impedance state.
A20GATE	50	I	A20GATE from the Keyboard Controller is used to force A20 low. When A20GATE is low, A20 on the CPU Address Bus is forced low. When A20GATE is high, A20 follows the CPU Address 20. Tie directly to the P21 Pin of the Keyboard Controller.
−AEN2 −AEN1	62 63	I	ADDRESS ENABLE 1 & 2 from DMA's 1 & 2, respectively. The signal is the result of the DMAAEN pin NAND'd with −MASTER. Tie directly from the −AEN1 and −AEN2 pins of 82231.
ALE	33	O	ADDRESS LATCH ENABLE is an active high signal that controls the address latches used to hold addresses during bus cycles. ALE is held inactive for Halt bus cycles.
BATT	32	I	BATTERY Power to the Clock Calendar and RAM.
−BUSY286	29	O	−BUSY286 is an active low output indicating the operating condition of the 80287 coprocessor to the processor. It is normally tied to the processor −BUSY pin.
−BUSY	11	I	−BUSY is an active low input from the 80287 to indicate that it is currently executing a command. It is used to generate the −BUSY286 output signal.
CCROSC	77	I	CLOCK CALENDAR OSCILLATOR; 32.768 KHz signal.
CCRRST	79	I	CLOCK CALENDAR RESET signal for the Real Time Clock. This is an active low input.
CCRR/W	61	I	CLOCK CALENDAR READ/WRITE signal for the Real Time Clock. A high enables READ/WRITE operation to the real-time clock. Tie directly from the CCRR/W Pin of 82231.
CNTLOFF	41	O	CONTROL OFF is used to enable the low byte data bus latch during byte accesses. This signal is active high.
CPU HLDA	65	I	CPU HOLD ACKNOWLEDGE is an active high input from the processor. An active condition indicates that the CPU has relinquished the bus to another bus master in the system.
−CS287	60	I	CHIP SELECT 287 is used to derive the −NPCS signal. Tie directly from the −CS287 pin of 82231.
DIR245	36	O	DIRECTION-245 controls the high to low byte and low to high byte conversion during data transfers to and from 8-bit peripherals.

82230 PIN DESCRIPTION (Continued)

Symbol	Pin No.	Type	Description
DT/$\overline{R}$	39	O	DATA TRANSMIT/RECEIVE establishes the data direction to and from the local data bus. When high, this output signals a CPU write bus cycle. A low indicates a CPU read bus cycle is being performed. This signal is always high when no bus cycle is active.
−ERROR	10	I	ERROR is a negative edge triggered input from the numeric processor indicating that an unmasked error condition exists. Tie directly from the −ERROR Pin of the 80287.
F16	45	I	F16 is an active high input indicating a word memory access. It is used to inhibit command delays for memory accesses.
+FSYS16	46	I	A latched version of F16.
−GATE245	37	O	GATE245 is an active low output. When active it enables the bus transceiver that performs the high to low byte conversion with the DIR245 signal. Conversion does not take place if A0 = 0 which indicates a word transfer.
−INTA	56	O	INTERRUPT ACKNOWLEDGE instructs an interrupting device that its interrupt request is being acknowledged. This signal is active low. −INTA is tri-stated when CPU HLDA is high and CNTLOFF is low. Tie directly to the −INTA pin of 82231.
INTR	22	O	INTERRUPT REQUEST is connected directly to the CPU's interrupt pin. INTR is active high, and is generated when a valid interrupt request has been asserted.
−INTR1CS	58	I	INTERRUPT CONTROLLER 1 (MASTER) CHIP SELECT is an active low output that is used to select the Interrupt Controller as an I/O device. This allows communication between the master interrupt controller and the CPU via the 'X' Data Bus. Tie directly from the −INTR1CS pin of 82231.
−INTR2CS	57	I	INTERRUPT CONTROLLER 2 (SLAVE) CHIP SELECT is an active low output that is used to select the Interrupt Controller as an I/O device. This allows communication between the slave interrupt controller and the CPU via the 'X' data bus. Tie directly from the −INTR2CS Pin of 82231.
−IO CS 16	35	I	I/O 16-BIT CHIP SELECT signals the system that the current data transfer is a 16-bit, one wait-state, I/O cycle. It is derived from an address decode and is an active low signal.
−IOR	53	I/O	I/O READ signal instructs a selected I/O device to drive its data onto the data bus. The −IOR signal is active low. It is tri-stated when CPU HLDA is high and CNTLOFF if low.
−IOW	52	I/O	I/O WRITE signal instructs a selected I/O device to read the data on the data bus. The −IOW signal is active low. It is tri-stated when CPU HLDA is high and CNTLOFF is low.

82230 PIN DESCRIPTION (Continued)

Symbol	Pin No.	Type	Description
IRQ0	59	I	INTERRUPT REQUEST 0 (system timer) receives interrupt requests from channel 0 of the timer/counter. Tie directly from the IRQ0 pin of 82231.
IRQ7–IRQ3 IRQ10–IRQ9 IRQ12–IRQ11 IRQ15–IRQ14	3-7 1-2 83-84 81-82	I I I I	INTERRUPT REQUESTS 3–7, 9–12, and 14–15 are used to signal the CPU that an I/O device needs attention. The interrupt requests are prioritized with IRQ9–IRQ12 and IRQ14–IRQ15 having the highest priority (IRQ9 highest) and IRQ3–IRQ7 having the lowest priority (IRQ7 lowest). IRQn signals are active high. The requesting signal is held high until the CPU acknowledges the interrupt request.
–LSDEN	40	O	LEAST SIGNIFICANT DATA ENABLE is an active low output. When active, it enables the transceiver/receiver connected to the least significant byte of the local data bus.
–MEMR	66	I/O	MEMORY READ COMMAND instructs a memory device to drive data onto the data bus. This signal is active low. –MEMR is active on all memory read cycles. It is tri-stated when CPU HLDA is high and CNTLOFF Output is low.
–MEMW	44	I/O	MEMORY WRITE COMMAND instructs a memory device to read the data on the data bus. This signal is active low. –MEMW is active on all memory write cycles. It is tri-stated when CPU HLDA is high and CNTLOFF Output is low.
–MSDEN	38	O	MOST SIGNIFICANT DATA ENABLE is an active low output. When active, it enables the transceiver connected to the most significant byte of the local data bus.
M/$\overline{IO}$	20	I	MEMORY-$\overline{INPUT\ OUTPUT}$ is the M/$\overline{IO}$ signal from the CPU. When high, it indicates a memory access. When low, it indicates an I/O access. It is used to generate the memory and I/O signals for the system.
NA20	16	I	NA20 is the CPU address 20. 82230 conditions this signal with A20GATE to produce A20. NA20 is tied directly from the CPU A20 output.
NC	80		Do Not Connect.
–NPCS	9	O	NUMERICAL PROCESSOR CHIP SELECT is an active low output used to select the 80287 Numerical Processor. It is tied directly to the $\overline{NPS1}$ pin of the 80287.
+OPT	49	I	KEYBOARD OUTPUT BUFFER FULL is an active high signal from the Keyboard Controller P24 Pin. The signal is an interrupt request (IRQ1) signaling a full keyboard buffer.
–0WS	47	I	ZERO WAIT STATE option. When pulled active (low), the current processor cycle can be terminated.

82230 PIN DESCRIPTION (Continued)

Symbol	Pin No.	Type	Description
PCLK	42	O	PERIPHERAL CLOCK is half the frequency of PROCCLK. It is used to clock peripheral controllers, specifically XTAL1 of the Keyboard Controller.
−PCLK	43	O	PERIPHERAL CLOCK INVERTED is the inverse of PCLK. It has been made available specifically for XTAL2 of the Keyboard Controller.
PROCCLK	28	O	PROCESSOR CLOCK provides the clock signal for the CPU and 80287 Numerical Processor. It is equal to the frequency of the crystal across pins X3 and X4. Tie directly to the CLK Pins of the 80286 and 80287.
$\overline{\text{POWER GOOD}}$	78	I	$\overline{\text{POWER GOOD}}$ is an active low input that indicates that system power is sufficient to maintain the integrity of the system. If high, it will force a system reset.
RC	51	I	RESET CPU from the keyboard controller P21 Pin.
$\overline{\text{READY}}$	21	O	$\overline{\text{READY}}$ is an active low output which signals that the current bus cycle is to be completed. S0, S1, $\overline{\text{POWER GOOD}}$, and 0WS control the $\overline{\text{READY}}$.
REFRDY	64	I	REFRESH/IO-CHANNEL-READY is generated by 82231. It is used to preset the READY Interface Asynchronous READY ($\overline{\text{ARDY}}$).
+RES 287	8	O	RESET 80287 is the reset signal for the 80287 Numerical Processor.
RES CPU	23	O	RESET CPU is the reset signal for the CPU. Active high, RESCPU is generated when either $\overline{\text{POWERGOOD}}$ or RC become active, or when the CPU generates a HALT status by forcing M/$\overline{\text{IO}}$ high. S0, S1 and A1 low. If this signal is initiated by RC, or by M/$\overline{\text{IO}}$, S0, S1 and A1, it will remain active for 16 PROCCLK cycles.
+RESET	55	O	RESET (SYSTEM) is an active high output derived from the $\overline{\text{POWER GOOD}}$ input. +RESET is used to force the system into an initial state. When +RESET is active, $\overline{\text{READY}}$ will also be active (Low).
S0, S1	18, 19	I	STATUS inputs from the CPU. The status signals are used by the bus controller to determine the state of the CPU.
SA0	13	O	ADDRESS 0 of the CPU bus. SA0 outputs A0 from the CPU during local CPU cycles. During a CPU Hold SA0 goes to a high impedance state so that another master on the expansion bus can take control. During an interrupt acknowledge this signal will be forced low.
SM/$\overline{\text{IO}}$	12	I	SYSTEM MEMORY-INPUT OUTPUT is the M/$\overline{\text{IO}}$ signal from the CPU, conditioned by ALE.
SYSCLK	54	O	SYSTEM CLOCK is the result of PROCCLK divided by two, thus synchronized to the processor's T-states. It may be used to clock peripheral devices that must be synchronized to the CPU.

82230 PIN DESCRIPTION (Continued)

Symbol	Pin No.	Type	Description
V_{CC1} V_{CC2}	24 75		POWER: +5V supply.
V_{SS1} V_{SS2} V_{SS3}	25 76 48		GROUND.
X3 X4	26 27	I O	CRYSTAL inputs used to generate PROCCLK and SYSCLK. The crystal frequency must be twice the processor clock frequency. Alternatively, an oscillator may be connected to X3.
XA0	31	I	ADDRESS 0 is used by the 8259A to decipher command words the CPU issues. XA0 works in conjunction with the read, write and chip select signals to the interrupt controller in determining whether the CPU wishes to issue a command or read the status of the controller.
XA3	30	I	ADDRESS 3 is used for generating the chip select and reset signals for the 80287.
−XBHE	34	I/O	BUS HIGH ENABLE is an active low signal which is used by 82230 to generate the MSDEN signal.
XD7–XD0	67–74	I/O	Data Bus 0–7 for the peripheral bus. The direction of the bus is determined by the −RDXDB signal from 82231. It is used by the 8259A to decipher command words the CPU issues.

82231 PIN DESCRIPTION

Symbol	Pin No.	Type	Description
−8042CS	63	O	8042 CHIP SELECT is an active low, chip select signal for the Keyboard Controller.
A23–A17	70–76	O	A23–A17 are the Address bits 17–23 of the CPU Address bus. They are outputs directly from the Memory Mapper Pins MO1–MO7 and supply page information during DMA transfers. These outputs are tri-stated unless HLDA and −MASTER are high.
+ACK	36	O	ACKNOWLEDGE is an active low output. When active it enables the bus transceiver between the system and peripheral (XBUX) bus. +ACK is used in conjunction with −RDXDB which controls the direction of the bus transceiver.
−AEN1 −AEN2	23 24	O O	ADDRESS ENABLE FROM DMAs 1 & 2, respectively. The signal is the result of the DMA's AEN signal NAND'd with −MASTER. Tie directly to the −AEN1 and −AEN2 pins of 82230.
CCRR/W	25	O	CLOCK CALENDAR READ/WRITE signal for the real-time clock. A high enables READ/WRITE operations to the real-time clock. Tie directly to the CCRR/W pin of 82230.
CPU HRQ	64	O	CPU HOLD REQUEST is an active high output indicating a DMA request to the CPU. It is also active during refresh cycles. CPU HRQ is normally connected to the 80286 HOLD Pin.

82231 PIN DESCRIPTION (Continued)

Symbol	Pin No.	Type	Description
−CS287	26	O	CHIP SELECT 287 is used by 82230 to derive the −NPCS signal. Tie directly to the −CS287 pin of 82230.
−DACK0−3 −DACK5−7	61−58 57−55	O O	DMA ACKNOWLEDGE 0−3 and 5−7 are used to acknowledge DMA requests (DRQ0−3 & 5−7). The output signal is an active low.
−DMAAEN	37	O	DMA ADDRESS ENABLE is an active low signal and is active when an I/O device is making a DMA access to system memory or during refresh.
−DPCK	39	I	DATA PARITY CHECK is used to generate NMI. This input is active low.
DRQ0−3 DRQ5−7	46−49 50−52	I I	DMA REQUEST 0−3 & 5−7 are synchronous channel requests used by peripheral devices and I/O processors to gain DMA service. The requests are prioritized with DRQ0 having the highest and DRQ7 having the lowest priorities. A DRQ line must be held active (high) until the corresponding DACK line goes active.
HLDA	21	I	HOLD ACKNOWLEDGE is an active high input that is equivalent to CPU HLDA. An active condition inidicates that the CPU has relinquished the bus to another bus master in the system.
−INTA	30	I	INTERRUPT ACKNOWLEDGE instructs an interrupting device that its interrupt is being acknowledged, and the device may place its interrupt vector onto the data bus. This input signal is active low. −INTA is used by 82231 in the generation of −RDXDB. Tie directly from 82230 Pin 56.
−INTR1CS	28	O	INTERRUPT CONTROLLER 1 (MASTER) CHIP SELECT is an active low output that is used by 82230 to select the Interrupt Controller as an I/O device. This allows communication between the Master Interrupt Controller and the CPU via the 'X' Data Bus. Tie directly to the −INTR1CS pin of 82230.
−INTR2CS	29	O	INTERRUPT CONTROLLER 2 (SLAVE) CHIP SELECT is an active low output that is used by 82230 to select the Interrupt Controller as an I/O device. This allows communication between the Slave Interrupt Controller and the CPU via the 'X' Data Bus. Tie directly to the −INTR2CS Pin of 82230.
−IOCHCK	43	I	I/O CHANNEL CHECK is an active low input. It is used to indicate an uncorrectable system error. It provides the system with parity error information about memory or devices on the I/O channel.
IOCHRDY	42	I	I/O CHANNEL READY is generated by an I/O device. When low it indicates a 'not ready' condition and forces the insertion of wait states in I/O or Memory accesses by the I/O device. When active (high), it will allow the completion of a memory or an I/O access by the I/O device.

82231 PIN DESCRIPTION (Continued)

Symbol	Pin No.	Type	Description
−IOR	33	I/O	I/O READ signal instructs a selected I/O device to drive its data onto the data bus. The −IOR signal is active low. It is used for data transfers between the CPU and I/O devices and by DMA transfers.
−IOW	34	I/O	I/O WRITE signal instructs a selected I/O device to read the data on the data bus. The −IOW signal is active low. It is used for data transfers between the CPU and I/O devices and by DMA transfers.
IRQ0	27	O	INTERRUPT REQUEST 0 (System Timer) from Channel 0 of the Timer/Counter. Tie directly to the IRQ0 Pin of 82230.
−MASTER	40	I	−MASTER is an active low input used in conjunction with a DRQ line to gain control of the system. A DMA controller or processor on the I/O channel may issue a DRQ to a DMA channel and receive a −DACK. The I/O processor may then activate −MASTER which will allow it to control the system address, data, and control lines.
NC	20		Do Not Connect.
NMI	65	O	NON-MASKABLE INTERRUPT is an active high output that is connected to the CPU NMI pin.
OSC	67	O	OSCILLATOR output is the clock frequency of the crystal connected across X1−X2. It is the OSC output from the Clock Generator.
P2	35	O	P2 is an active high output indicating that a valid refresh address is available on the XA bus.
−RDXDB	38	O	READ X-DATA BUS controls the direction of the bidirectional buffer between the least significant byte of the 'S' Data Bus and the 'X' Data Bus. −RDXDB is used in conjunction with +ACK to control XBUS activity. When +ACK is active (low) and −RDXDB is low, data is to be read from the peripheral bus. When +ACK is active (low) and −RDXDB is high, data is to be written to the peripheral bus.
REFRDY	22	O	REFRESH/IO-CHANNEL-READY is generated by +REFRESH OR'd with IOCHRDY. It is used by 82230 to preset the Clock Generator & Ready Interface Asynchronous Ready ($\overline{\text{ARDY}}$).
−REFRESH	41	I/O	REFRESH is an active low output used to initiate a refresh cycle for the dynamic RAMs.
+RESET	31	I	RESET (SYSTEM) is an active high input from 82230. +RESET is used to force 82231, as well as the system, into an initial state. Tie directly from 82230 Pin 55.
SPEAKER	66	O	SPEAKER DATA is an output of the Programmable interval timer tone signal used to drive the speaker.
SYSCLK	32	I	SYSTEM CLOCK input from 82230. It is used to synchronize 82231 to the system. Tie directly from 82230 SYSCLK Pin.
TC	62	O	TERMINAL COUNT provides a pulse when the terminal count for any DMA channel is reached.

82231 PIN DESCRIPTION (Continued)

Symbol	Pin No.	Type	Description
V_{CC1} V_{CC2}	11 53		POWER: +5V supply.
V_{SS1} V_{SS2}	10 54		GROUND.
X1 X2	68 69	I O	CRYSTAL inputs for the internal oscillator used to generate clocking for I/O devices. A parallel resonant fundamental frequency mode crystal is required. An alternative oscillator may be connected to X1.
XA8–XA0 XA9 XA16–XA10	1–9 84 77–83	I/O I/O O	XBUS ADDRESSES 0–16 are the peripheral addresses for the local I/O bus.
XD0–XD7	12–19	I/O	Data Bus 0–7 for the peripheral bus. The direction of the bus is determined by the –RDXDB signal from 82231.
–XMEMR	44	I/O	MEMORY READ signal indicating a DMA read operation from peripheral devices or memory.
–XMEMW	45	O	MEMORY WRITE signal indicating a DMA write operation to peripheral devices or memory. It is tri-stated except during DMA transfers.

FUNCTIONAL DESCRIPTION

Introduction

The 82230 and 82231 are a two-chip implementation of LSI/MSI/SSI logic controlling the IBM Personal Computer AT. The devices provide a low power, highly integrated PC-AT design solution that may also be applied to any 80286-based system. With the 82230 and 82231, a PC-AT system can be designed to operate at 12 MHz with zero wait state RAM accesses.

These standard cell products contain most of the logic peripheral to the microprocessors and memory on the "standard" AT motherboard. The LSI peripherals which support the AT design reside as supercells on the 82230/82231 chips. These peripherals are compatible with the products they replace and the user should refer to the standard product data sheets for additional information on the operation of these devices.

The 82230 performs the functions of the 82284 Clock Generator & Ready Interface, 82288 Bus Controller, 6818 Real Time Clock/RAM, and the Master-Slave implementation of the dual 8259A Programmable Interrupt Controllers as well as Command Delay, Shut Down, Address/Data Bus Control and Ready Generation logic.

The 82231 includes the 8254 Programmable Interrupt Timer, 8284A Clock Generator, LS612 Memory Mapper and the dual 8237 DMA Controller functions as well as Refresh Generation and Refresh/DMA Arbitration Logic.

PC-AT BLOCK DIAGRAM

The block diagram is shown below in Figure 1 which shows the 82230 and 82231 being used in a PC-AT compatible design. Note how the basic structure of the PC-AT is retained in the design. The five address busses and four data busses are present. The 82230 and 82231 'sit' on the X Address and Data Busses and monitor the status and control line outputs from the 80286 in order to operate the peripheral supercells.

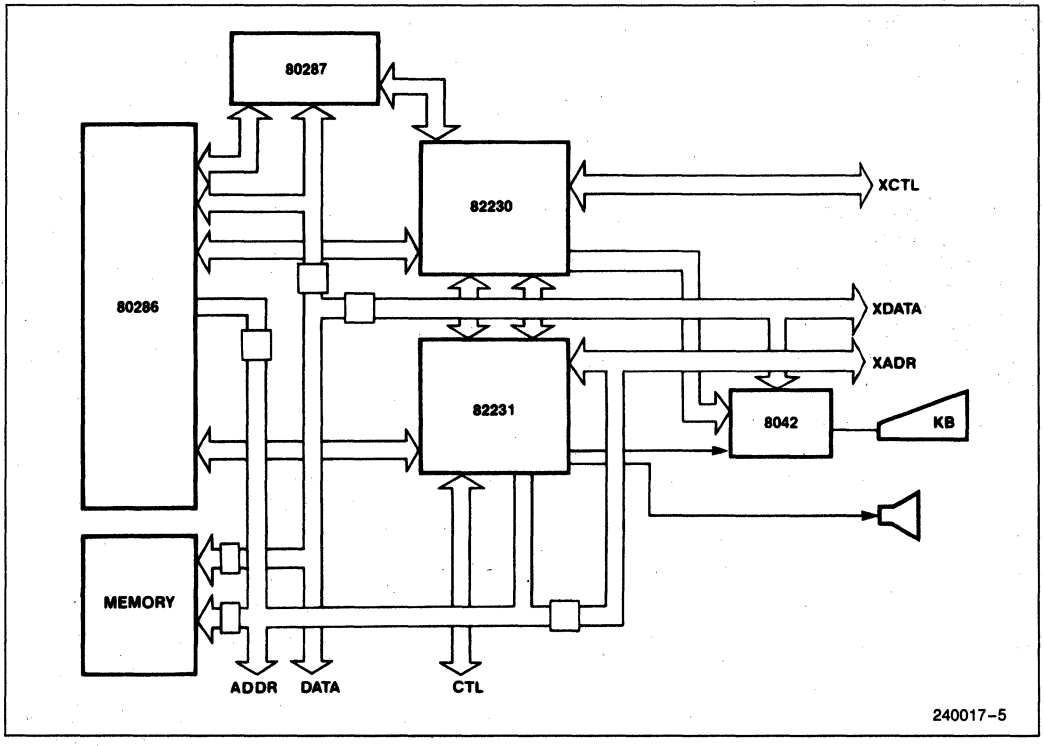


Figure 1. Block Diagram of 82230 and 82231 for PC/AT

A brief description of each of the basic PC-AT busses is included in order to show how a system design using the 82230/82231 allows for the retention of the PC-AT bus structure.

ADDRESS BUSSES

The 82230/82231 allows a straightforward PC-AT design which preserves the five basic address busses. System designers can interact with these busses, create new busses, or eliminate busses, depending upon design objectives. Each bus must be independently buffered and separated by buffers/latches. Most handshake signals are generated by the 82230/82231.

The Local Address Bus is comprised of the 24 address pins emanating from the 80286. External buffers are required in order to separate the local address bus from the system address bus. A0, however, is input directly into the 82230 and is used in conjunction with XBHE in order to enable the appropriate memory bank. The 82230 inputs CPU address bit NA20 and outputs A20.

The System Address Bus is the main AT address bus. This is a latched version of the local address bus, and is a 20-bit bus. SA0 is an output of the 82230 and SA1–SA19 are latched from the local address bus. The latch signal is ALE, which is generated by the 82288 supercell in the 82230. CPU HLDA, generated by the 80286 and active high, should be used as the output enable signal.

The Memory Address Bus applies to the RAM on the PC-AT system board only. It is a multiplexed version of the System Address Bus with nine address lines, MA0–MA8. External multiplexers are used to generate the row and column addresses.

The X Address Bus is separated from the System Address Bus. The X address lines are input/outputs into the 82230/82231. The X Address Bus is a motherboard address bus which is used to address ROM (BIOS) and motherboard I/O. In addition, the X Address Bus generates addresses for DMA and memory refresh.

The L Address Bus is an unlatched 7-bit address bus. LA17–LA23 can allow a PC-AT design up to 16 MBytes of address space. The L Address Bus should be made available at the expansion bus connector.

DATA BUSSES

The Local Data Bus is the name for the data bus lines emanating directly from the 80286. The local data bus has 16 lines, D0–D15. Because the 80286 can do word and byte transfers and because word transfers need not be aligned, it is necessary to design a bus interface which differentiates between the high bus byte and the low bus byte.

The System Data Bus is the main data bus of a PC-AT system and interfaces with all other data busses. The 82230 and 82231 are designed to control these interfaces in order to simplify system design and maximize bus flexibility.

The Memory Data Bus interfaces both DRAM and ROM. It is a 16-bit bus and connects with the System Data Bus through buffers.

The X Data Bus is the bus intended primarily for system board I/O functions. It interfaces to functions such as the DMA controllers, Interrupt controllers, Keyboard controller, and Real Time Clock.

82230/82231 Interface

The 82230 and 82231 are relatively independent of each other; the 82230 generates most of the timing and control signals and the 82231 controls the X Address Bus for DMA and refresh. Both chips have additional functions but because of the desire to partition the system design such that the 82230 and the 82231 could be assembled in low cost 84-pin PLCC packages, each chip relies on the other for certain functions. This entails introducing dedicated interface signals between the 82230 and 82231 into a system design. The 82230/82231 interface requires 14 pins on each device and these pins are described below.

- REFRDY is generated by the 82231 and used to tell the 82230 to insert wait-states in response to the 82231 input IOCHRDY. IOCHRDY is active high.
- −AEN1 and −AEN2 are signals generated by the 82231 which indicate DMA byte (−AEN1) or word (−AEN2) transfers, and are used by the 82230 to generate bus buffer control signals.
- CCRR/W is generated by the 82231 and used by the 82230 as part of the 6818 chip select.
- −INTR1CS and −INTR2CS are generated by the 82231 and used by the 82230 as the interrupt controller chip selects.
- −CS287 is generated by the 82231 and is used by the 82230 to generate 80287 control signals.
- IRQ0 is the output of the 8254 timer 0 on the 82231 and is connected to interrupt request 0 on the master interrupt controller in the 82230.

- +RESET is generated by the 82230 in response to $\overline{\text{POWERGOOD}}$ and is used by the 82231 for initialization.
- SYSCLK is PROCCLK divided by two.
- −INTA, −MEMR, −IOR, and −IOW are commands generated by the 82230 in response to the 80286 status inputs S0 and S1, and are used by the 82231 as basic system commands.

Coprocessor Interface

The 82230 contains a coprocessor interface logic block to allow interfacing with an 80287 math coprocessor. The coprocessor interface includes the IBM PC-AT compatible error handling hardware.

Memory Operations

When the 82230 and 82231 are used in a PC-AT system design, the system can be designed to operate with the full 16 MBytes of memory that the 24 address lines of the 80286 allow.

The 82230/82231 chipset normally operates with one wait state inserted for memory operations and four wait-states inserted for I/O operations. The number of wait states may be increased for slow memory or I/O devices, or decreased if use of high-speed memory or I/O devices is desired in order to provide higher system performance.

During normal operation, the number of ROM accesses are relatively few compared to RAM accesses, so ROM subsystem speed does not significantly affect system performance. When designing high performance PC-AT systems, it should be verified that the ROM subsystem is fast enough for operation. Note also that older versions of the IBM BIOS may not operate in systems faster than 8 MHz. Modifications of both RAM and ROM subsystem performance are covered in detail in the Intel 286EX Application Note.

System Clocks and Oscillators

Because of the high level of integration of the 82230/82231, several different clock frequencies are present on the chips. Figure 2 shows the relationships between the various clock signals.

PROCCLK is the main system clock which is used in a PC-AT system to generate all the basic system and bus timings. The clock is generated by the 82230 and operates at twice the system clock frequency. See Table 1 for processor clock capacitance. The tolerance of the PROCCLK oscillator is independent of the 82230/82231 and is primarily determined by the requirements of the 80286 and 80287 processors. For reliable operation at the specified V_{DD} and temperature condition, the processor timing specifications must not be exceeded.

Table 1. Recommended Fundamental Mode Crystal Characteristics and Recommended Load Capacitance for PROCCLK

Oscillator	Crystal Freq.	C_{LOAD} (pF)	C_{IN} (pF) X3	C_{OUT} (pF) X4
PROCCLK	12 MHz	20	30	10
PROCCLK	16 MHz	20	30	10
PROCCLK	20 MHz	20	27	8
PROCCLK	24 MHz	20	22	8

SYSCLK is PROCCLK divided by two, and should be used as the expansion bus clock. SYSCLK is an output of the 82230. To ensure compatibility with expansion cards, SYSCLK should not be above 8 MHz.

X1 is an input to the 82231 which is 12 times the frequency used to clock the three counters in the 8254 timer on the 82231. In order to be compatible with AT hardware and software, it should be a 14.318 MHz fundamental mode crystal with C_{LOAD} = 32 pF, and a 27 pF capacitor should be placed in series with the crystal. Alternately, a 14 MHz funda-

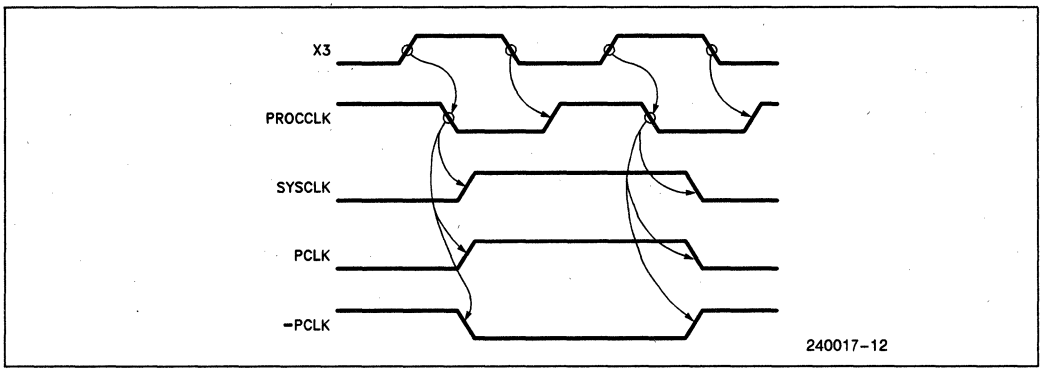


Figure 2. 82230 Clock Timing

mental mode crystal (C_{LOAD} = 32 pF) may be trimmed with a 5–50 pF trimmer capacitor in series for high accuracy applications such as NTSC or RS170 video-compatibility with chroma colorburst.

Note that the trim cap must be adjusted with a low-capacitance nylon or teflon tuning wand for accurate trimming. Tolerance for this clock is 0.1% or less for non-video or non time-critical use, 0.01% or greater if used for video color-burst or time-critical applications.

The CCR clock is the low-frequency oscillator used to clock the 6818 Clock/Calendar/RAM on the 82230. CCROSC tolerance is variable and dependent upon real-time clock accuracy requirements. See Table 2 for CCROSC clock tolerance and accuracy.

Table 2. CCROSC Tolerance/Accuracy

Tolerance	Accuracy
0.001% or 10 ppm	5 minutes/year trimming required
0.01% or 100 ppm	1 minute/week no trimming required
0.02% or 200 ppm	2 minutes/week no trimming required
0.05% or 500 ppm	5 minutes/week no trimming required

EXTERNAL OSCILLATORS

External CMOS output drive oscillators may be used for either PROCCLK or OSC. Simply connect the external oscillator outputs to PROCCLK inputs X1 or X3. TTL output oscillators may be used if the output drive V_{OH} is greater than 4.1V; pull-up resistors will generally suffice. The oscillator invertor outputs X2 and X4 may be left open, or may be used to drive one moderate rise-time CMOS load if needed.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature .....−65°C to +150°C
Voltage on any Pin
with Respect to Ground ....−0.5V to V_{CC} +0.5V
Power Dissipation 1W

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

NOTICE: Specifications contained within the following tables are subject to change.

82230/82231 DC CHARACTERISTICS

V_{CC} = 5V ±5%, V_{BAT} = 2.8V to V_{CC}, T_A = 0°C to +70°C

Parameter	Conditions	Min	Max	Units
V_{IL}			0.5	V
V_{IH}		2.0		V
V_{IL} 82230 Pins 26, 77, 78, 79 82231 Pins 41, 42, 68			0.5	V
V_{IH} 82230 Pins 26, 77, 78, 79 82231 Pins 41, 42, 68		4.5		V
V_{IH} 82230 Pin 47		4.5(1)		V
I_{I0}	V_{IN} = 0	−100		µA
I_{I0} 82230 Pins 26, 77, 78, 79	V_{IN} = 0	−10		µA
I_{I1}	V_{IN} = V_{CC}		10	µA
I_{OH} Except for 82231 Pin 41(2)	V_{OH} = 2.4		−4	mA
I_{OL}	V_{OL} = 0.45V	4		mA
I_{OL} 82230 Pins 13, 28, 44, 52, 53 54, 56, 66 82231 Pins 33 34, 67	V_{OL} = 0.45V	16		mA
I_{OL} 82231 Pin 41	V_{OL} = 0.45V	18		mA
I_{OZ}	V_O = 0 to V_{CC}	−10	+10	µA
I_{CC} 82230	F = 10 MHz F = 12 MHz		55 60	mA mA
I_{CC} 82231	F = 10 MHz F = 12 MHz		45 50	mA mA
I_{CC} 82230 from Battery	F = 32.768 KHz V_{BAT} = 5V V_{CC} = 0V		25	µA
I_{CC} 82230 from Battery	F = 32.768 KHz V_{BAT} = 2.8V V_{CC} = 0V		20	µA

NOTES:
1. −0WS (82230 Pin 47) is driven by an open collector output. It is pulled up to CMOS voltage levels of V_{CC} − 0.5V by a pullup resistor.
2. −REFRESH (82231 Pin 41) is an open collector output.

82230 AC CHARACTERISTICS (V_{DD} = 5V ±5%, T_A = 0°C to 70°C)

Symbol – Figure	Parameter	10 MHz		12 MHz		Units	Notes
		Min	Max	Min	Max		
53–15	–0WS Setup Time to PROCCLK ↓	25		25		ns	13
54–15	–0WS Hold Time from PROCCLK ↓	0		0		ns	13
	–0WS Setup Time to PROCCLK ↓	36		36		ns	11
	–0WS Hold Time from PROCCLK ↓	0		0		ns	11
69–22	A0 Setup Time to ALE	30		25		ns	
	A0 Hold Time from ALE	0		0		ns	
26–6	A1 Setup Time to S1, S0	27		22		ns	
27–6	A1 Hold Time from S1, S0	0		0		ns	
43–13	A20 Delay from NA20		27		22	ns	
44–13	A20 Delay from A20GATE		37		32	ns	
45–13	A20 Disable Delay from CPUHLDA ↑		35		30	ns	4
46–13	A20 Enable Delay from CPUHLDA ↓		35		30	ns	
17–3	ALE Active Delay from PROCCLK ↓		25		23	ns	
18–3	ALE Inactive Delay from PROCCLK ↓		30		25	ns	
68–21	$\overline{\text{BUSY286}}$ Delay from $\overline{\text{BUSY}}$, –IOW		35		35	ns	
	CCROSC High Time	25		25		µs	5
	CCROSC Low Time	25		25		µs	5
	CCROSC Input Rise/Fall Time		20		20	ns	12
39–10, 11	CCRR/W Setup Time to IOR/IOW ↓	0		0		ns	
40–10, 11	CCRR/W Hold Time from IOR/IOW ↑	17		15		ns	
	CCRRST Pulse Width	100		83		ns	
67–20	CNTLOFF Delay from PROCCLK ↓		30		25	ns	
70–23	CPUHLDA Setup Time to PROCCLK ↓	20		15		ns	
71–23	CPUHLDA Hold Time from PROCCLK ↓	0		0		ns	
49–14	DIR245 Delay from –IOR ↓, –IOW ↓		17		15	ns	
49–14	DIR245 Delay from –MEMR, –MEMW		17		15	ns	
	DIR245 Delay from –AEN1, –AEN2		40		35	ns	
55–16	DT/R Delay High from PROCCLK ↓		45		40	ns	
56–16	DT/R Delay Low from PROCCLK ↓		45		40	ns	
60–16	F16 Setup Time to PROCCLK ↓	30		30		ns	
61–16	F16 Hold Time from PROCCLK ↓	0		0		ns	
	+FSYS16 Setup Time to PROCCLK ↓	100		83		ns	
	+FSYS16 Hold Time from PROCCLK ↓	50		40		ns	
50–14	–GATE245 Delay from –IOR ↓, –IOW ↓		22		20	ns	
50–14	–GATE245 Delay from –MEMR, –MEMW		22		20	ns	
	–GATE245 Delay from –AEN1, –AEN2		45		40	ns	

82230 AC CHARACTERISTICS (V_{DD} = 5V ±5%, T_A = 0°C to 70°C) (Continued)

Symbol –Figure	Parameter	10 MHz		12 MHz		Units	Notes
		Min	Max	Min	Max		
37–9	Interrupt Request Pulse Width	100		100		ns	8
29–7, 8	–IOR, –IOW Active Delay from PROCCLK ↓	3	15	3	15	ns	
30–7, 8	–IOR, –IOW Inactive Delay from PROCCLK ↓	3	15	3	15	ns	
	–IOR, –IOW Enable/Disable Delay from CPUHLDA		40		40	ns	
	–INTA Active Delay from PROCCLK ↓	3	45	3	35	ns	
	–INTA Inactive Delay from PROCCLK ↓	3	45	3	35	ns	
	–INTA Enable/Disable Delay from CPUHLDA		40		40	ns	
38–9	INTR Delay from Interrupt		175		150	ns	
33–7, 8	–INTR1CS, – INTR2CS Setup Time to –IOR, –IOW ↓	0		0		ns	
34–7, 8	–INTR1CS, – INTR2CS Hold Time from –IOR, –IOW ↑	0		0		ns	
72–24	–IO CS 16 Setup Time to SYSCLK ↓	85		75		ns	
73–24	–IO CS 16 Hold Time from SYSCLK ↓	0		0		ns	
57–16	–LSDEN, –MSDEN Active Delay from PROCCLK ↓		45		40	ns	
58–16	–LSDEN, –MSDEN Inactive Delay from PROCCLK ↓		35		30	ns	
66–19	–LSDEN, –MSDEN Delay from –NPCS		15		15	ns	
	–LSDEN, –MSDEN Active Delay from SM/$\overline{IO}$ after –CS287 Inactive		30		30	ns	
79–16	–MEMR, –MEMW Active Delay from PROCCLK ↓	3	15	3	15	ns	
80–16	–MEMR, –MEMW Inactive Delay from PROCCLK ↓	3	15	3	15	ns	
	–MEMR, –MEMW Enable/Disable Delay from CPUHLDA		40		40	ns	
64–17	–MSDEN Delay from –XHBE		27		25	ns	
62–16	M/$\overline{IO}$ Setup Time to PROCCLK ↓	28		25		ns	
63–16	M/$\overline{IO}$ Hold Time from PROCCLK ↓	0		0		ns	
65–18	–NPCS Delay from SM/$\overline{IO}$–CS287, XA3, –INTA		40		35	ns	
11–2	PCLK, –PCLK High Time	45		35		ns	
12–2	PCLK, –PCLK Low Time	45		35		ns	
13–2	PCLK, –PCLK Delay from PROCCLK		45		40	ns	
14–2	PCLK, –PCLK Rise/Fall Times		7.5		5	ns	12
19–4	$\overline{POWER\ GOOD}$ Setup Time to PROCCLK ↓	26		26		ns	3
	$\overline{POWER\ GOOD}$ Hold Time from PROCCLK ↓	50		41		ns	3
8–4	$\overline{POWER\ GOOD}$ Rise/Fall Times		20		20	ns	12
	$\overline{POWER\ GOOD}$ Inactive Pulse Width	1		1		μs	
5–2	PROCCLK Delay from X3	5	25	5	20	ns	
6–2	PROCCLK High Time	16		13		ns	
7–2	PROCCLK Low Time	12		11		ns	
9–2	PROCCLK Rise/Fall Time		8		8	ns	12
22-5	–RC Setup Time to SYSCLK ↑	30		30		ns	3
23-5	–RC Pulse Width	100		83		ns	
51–15	$\overline{READY}$ Active Delay from PROCCLK ↓		22		18	ns	
52–15	$\overline{READY}$ Inactive Delay from PROCCLK ↓		70		60	ns	

82230 AC CHARACTERISTICS (V_{DD} = 5V ±5%, T_A = 0°C to 70°C) (Continued)

Symbol – Figure	Parameter	10 MHz		12 MHz		Units	Notes
		Min	Max	Min	Max		
	REFRDY Pulse Width	50		40		ns	
	REFRDY Hold Time from PROCCLK ↓	34		34		ns	
	REFRDY Setup Time to PROCCLK ↓	−14		−14		ns	
74–25	RES 287 Delay from −IOW		60		50	ns	
21–4	RES CPU Delay from PROCCLK ↓		27		22	ns	
20–4	+RESET Delay from PROCCLK ↓		50		50	ns	
41–12	SA0 Enable Time from CPU HLDA		60		50	ns	
42–12	SA0 Disable Time from CPU HLDA		60		50	ns	4
15–3	S1, S0 Setup Time to PROCCLK ↓	28		15		ns	
16–3	S1, S0 Hold Time from PROCCLK ↓	0		0		ns	
10–2	SYSCLK Delay from PROCCLK ↓	5	20	5	20	ns	
1–2	X3 Period	50		41.7		ns	
2–2	X3 Low Time	17		15		ns	
3–2	X3 High Time	23		20		ns	
4–2	X3 Rise/Fall Times		5		3	ns	12
35–8, 11	XD0–XD7 Delay Time from −IOR ↓		45		40	ns	
36–8, 11	XD0–XD7 Hold Time from −IOR ↑		17		15	ns	
31–7	XD0–XD7 Setup Time to −IOW ↑	100		83		ns	
32–7	XD0–XD7 Hold Time from −IOW ↑	0		0		ns	

82231 AC CHARACTERISTICS (V_{DD} = 5V ±5%, T_A = 0°C to 70°C)

Symbol – Figure	Parameter	10 MHz		12 MHz		Units	Notes
		Min	Max	Min	Max		
96–27	−8042CS Delay from XA_X		60		48	ns	
98–28	A17–A23 Delay from SYSCLK ↑		150		125	ns	
99–28	A17–A23 Enable Delay from HLDA or −MASTER		100		83	ns	
97–28	A17–A23 Disable Delay from HLDA or −MASTER		100		83	ns	4
100–29	+ACK Delay from HLDA		45		40	ns	
101–29	+ACK Delay from −MASTER		45		40	ns	
109–30	−AEN1, −AEN2 Delay from SYSCLK ↑		130		115	ns	
94–27	CCRR/W Delay from XA_X		60		48	ns	
102–29	CCRR/W Delay from HLDA or −MASTER		50		41	ns	
108–30, 39	CPU HRQ Delay from SYSCLK ↑		80		70	ns	
95–26	−CS287 Delay from XA_X		60		48	ns	
103–29	−CS287 Delay from HLDA or −MASTER		50		41	ns	
113–30	−DACK0–3, −DACK5–7 Delay from SYSCLK ↑		110		100	ns	
110–30	−DMAAEN Delay from SYSCLK ↑		140		120	ns	

82231 AC CHARACTERISTICS (V_{DD} = 5V ±5%, T_A = 0°C to 70°C) (Continued)

Symbol – Figure	Parameter	10 MHz		12 MHz		Units	Notes
		Min	Max	Min	Max		
132–32	–DPCK Setup Time to –XMEMR ↑	8		6		ns	
133–32	–DPCK Hold Time from –XMEMR ↑	5		5		ns	
107–30	–DRQ0–3, –DRQ5–7 Setup Time to SYSCLK ↑	0			0	ns	3, 7
	HLDA Setup Time to SYSCLK ↑	70		65		ns	
	HLDA Hold Time from SYSCLK ↑	0		0		ns	
28–7	–INTR1CS, –INTR2CS Delay from XA_X		60		41	ns	
104–29	–INTR1CS, –INTR2CS Delay from HLDA or –MASTER		60		41	ns	
134–33	–IOCHCK Pulse Width		25		20	ns	
	IOCHRDY Setup Time to SYSCLK ↑ (During Refresh)	25		25		ns	
	IOCHRDY Hold Time from SYSCLK ↑ (During Refresh)	25		25		ns	
114–30	–IOR, –IOW Active Delay from SYSCLK ↑ (During DMA Transfers)		125		100	ns	
115–30	–IOR, –IOW Inactive Delay from SYSCLK ↑ (During DMA Transfers)		115		100	ns	
116–30	–IOR, –IOW Float to Inactive Delay from SYSCLK ↑ (During DMA Transfers)		120		100	ns	
117–30	–IOR, –IOW Inactive to Float Delay from SYSCLK ↑ (During DMA Transfers)		170		156	ns	4
152–37	–IOW Active Pulse Width (During CPU Transfers)	90		75		ns	
137–34	IRQ0 Delay from X1		100		100	ns	
	NMI Delay from –XMEMR ↑		100		83	ns	
135–33	NMI Delay from –IOCHCK ↓		100		83	ns	
143-35	OSC Low Time	20		20		ns	
144-35	OSC High Time	20		20		ns	
145-35	OSC Rise/Fall Times		15		15	ns	12
146-35	OSC Delay from X1		30		24	ns	
	P2 Delay from SYSCLK ↑		100		83	ns	
148–36	–RDXDB Delay from –IOR		100		83	ns	
149–36	–RDXDB Delay from –INTA		100		83	ns	
	REFRDY Delay from IOCHRDY	10	35	10	35	ns	
157–39	–REFRESH Delay from HLDA		28		24	ns	
158–39	–REFRESH Delay from SYSCLK ↑		100		83	ns	
	+RESET Active Pulse Width	200		160		ns	
138–34	SPEAKER Delay from X1		100		100	ns	
91–26	SYSCLK Period	100		83		ns	
92–26	SYSCLK Low Time	40		30		ns	
93–26	SYSCLK High Time	40		30		ns	

82231 AC CHARACTERISTICS (V_{DD} = 5V ±5%, T_A = 0°C to 70°C) (Continued)

Symbol – Figure	Parameter	10 MHz		12 MHz		Units	Notes
		Min	Max	Min	Max		
124–30	TC Delay from SYSCLK ↑		110		100	ns	
140–35	X1 Low Time	30		30		ns	
141–35	X1 High Time	30		30		ns	
142–35	X1 Rise/Fall Times		5		5	ns	12
150–37	XA_X Input Setup Time to –IOW ↓ (During CPU Transfers to 82231)	100		83		ns	
151–37	XA_X Input Hold Time from –IOW ↑ (During CPU Transfers to 82231)	45		40		ns	
	XA_X Input Hold Time from –IOR ↑ (During CPU Transfers to 82231)	100		67		ns	
111–30	XA_X Valid Delay from SYSCLK ↑ (During DMA Transfers)		160		150	ns	
159–39	XA_X Valid Delay from SYSCLK ↑ (During Refresh)		90		75	ns	
112–30	XA_X Disable Delay from SYSCLK ↑ (During DMA Transfers)		150		130	ns	4
153-37	XD_X Input Setup Time to –IOW ↑	100		83		ns	
154-37	XD_X Input Hold Time from –IOW ↑	17		15		ns	
155-38	XD_X Output Delay from –IOR ↓		125		100	ns	
156-38	XD_X Output Hold Time from –IOR ↑	17	70	15	60	ns	
125–31	–XMEMR Active Delay from SYSCLK ↑		110		100	ns	
126–31	–XMEMR Inactive Delay from SYSCLK ↑		110		100	ns	
127–31	–XMEMR Enable/Disable Delay from SYSCLK ↑		120		120	ns	4, 12
121–30	–XMEMW Active Delay from SYSCLK ↑		110		100	ns	
122–30	–XMEMW Inactive Delay from SYSCLK ↑		110		100	ns	
123–30	–XMEMW Enable/Disable Delay from SYSCLK ↑		120		120	ns	4, 12

NOTES:
1. To provide clearly understood information, the complex timing diagrams depict operation in a standard IBM PC AT system design. Combinational logic data paths are shown with less complex timing diagrams. The signal source (82230, 82231, PROCESSOR, LOGIC, etc.) follows the signal name.
2. The direction control signals are delayed to PROCCLK ↓ on an –IOW cycle. This is done to avoid changing the direction of the byte-swapping bus transceivers while data is still on the bus.
3. This signal is an asynchronous input. The timing specification is provided for testing purposes only to assure recognition at a specific clock edge.
4. The output float or high impedance condition occurs when output current is less than I_{OZ} in magnitude.
5. The frequency of CCROSC sets the count rate for the real time clock. CCROSC frequency, accuracy and stability, should be maintained as close as possible to 32.768 KHz to insure the validity of time and data information.
6. Input rise and fall times are assumed to be less than 20 ns unless otherwise specified.
7. DRQ_X must be held active with $DACK_X$ is returned.
8. The interrupt request inputs include IRQ0, IRQ3–7, IRQ9–12, IRQ14–15 and +OPT.
9. Address XA_{0-15} are output for byte DMA operations. XA_{0-16} are output for word DMA operations, with XA0 low.
10. A minimum of 16 PROCCLK cycles must occur before $\overline{\text{POWERGOOD}}$ becomes valid.
11. At the end of TC phase 1 after TCW2 or TCW3 for 16-bit transfer to 8-bit source/destination, for the first 8 bits of transfer.
12. These are not tested. They are guaranteed by design characterization.
13. At the end of TC phase 1 for 16 bit back plane memory transfers and 8-bit transfers after TCW2 or TCW3.

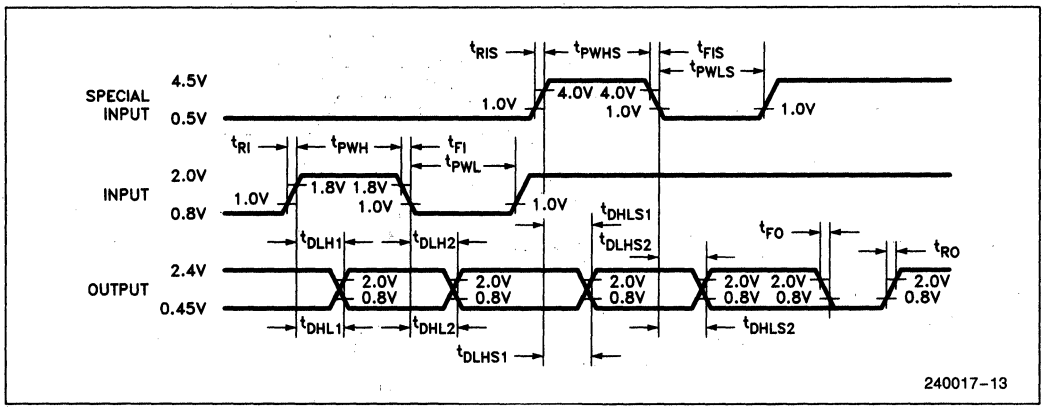


Figure 1A. Delay Time and Pulse Width Measurements

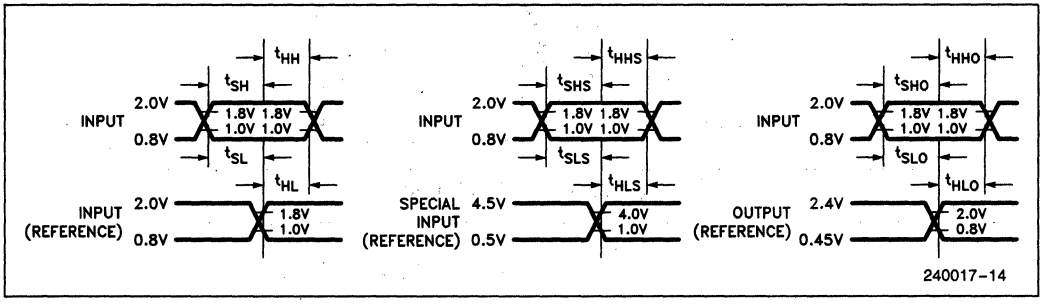


Figure 1B. Setup/Hold Time Measurements

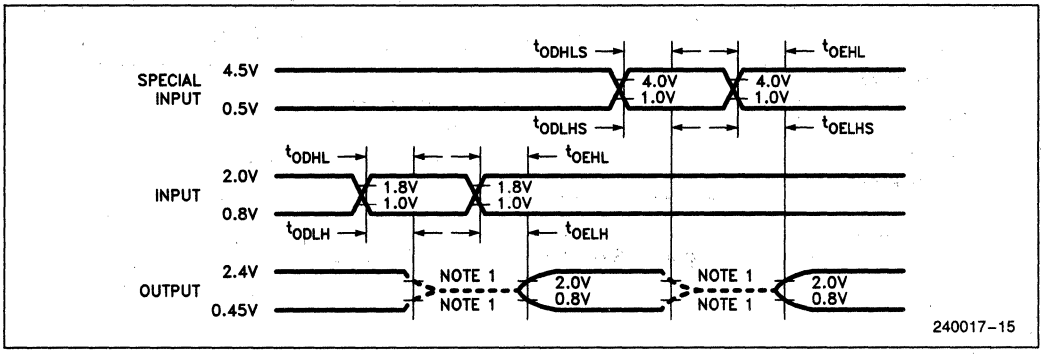


Figure 1C. Output Enable/Disable Time Measurement

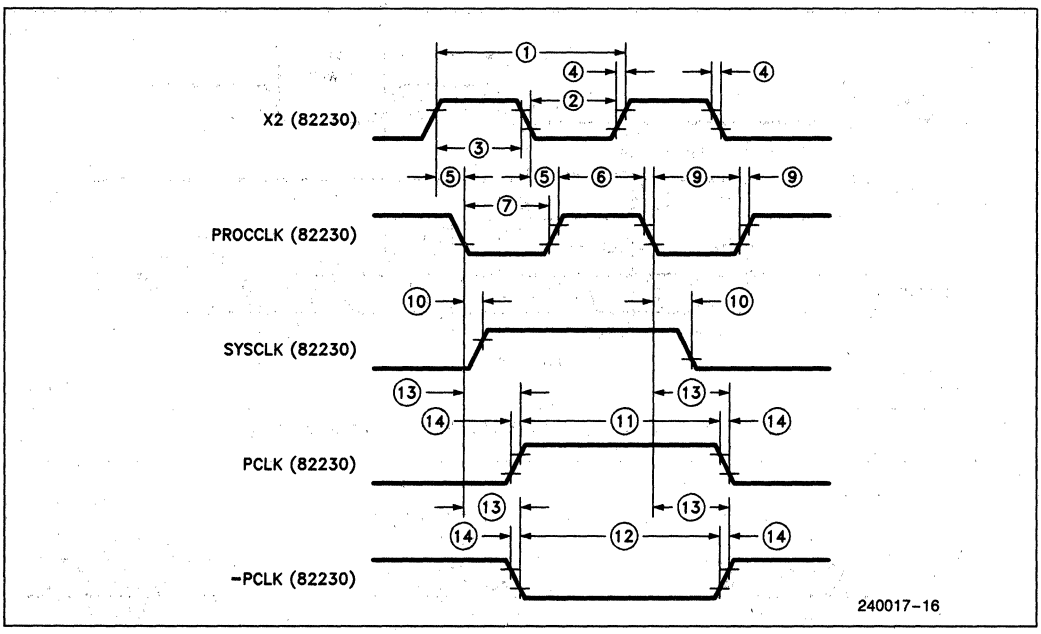


Figure 2. 82230 Clock Timing

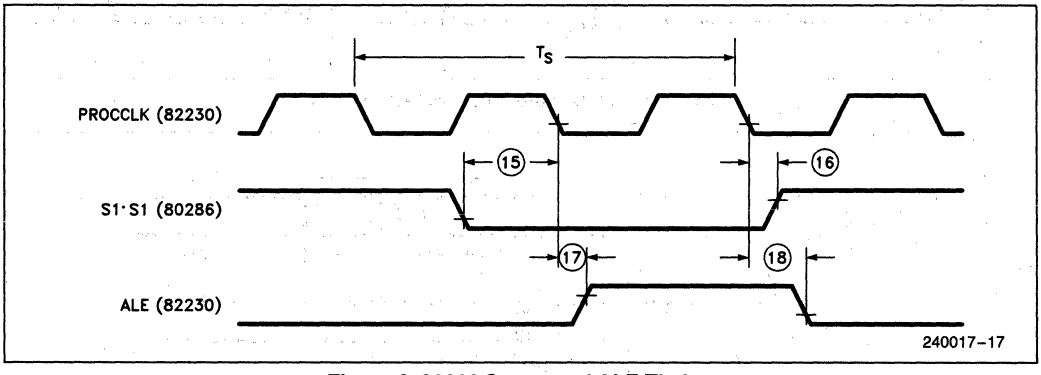


Figure 3. 82230 Status and ALE Timing

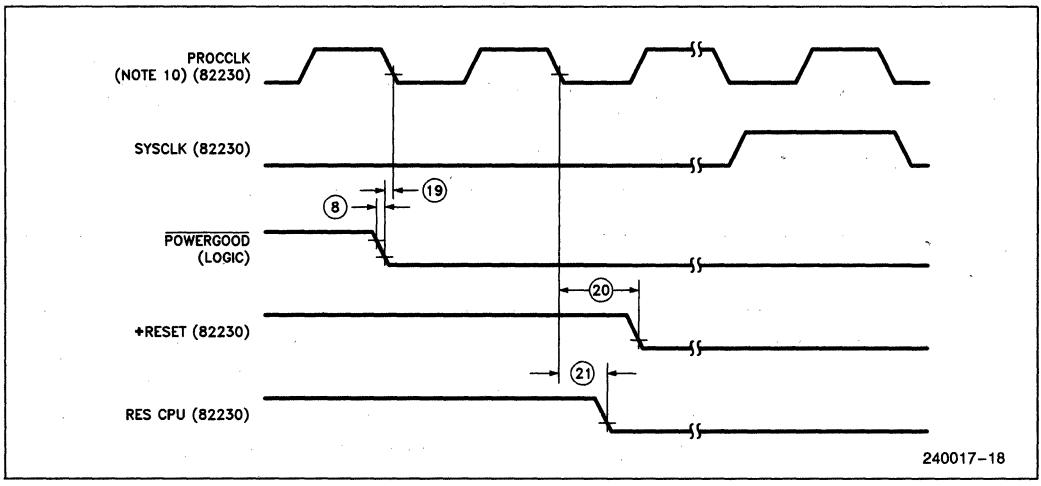


Figure 4. 82230 Power on Initiated Reset

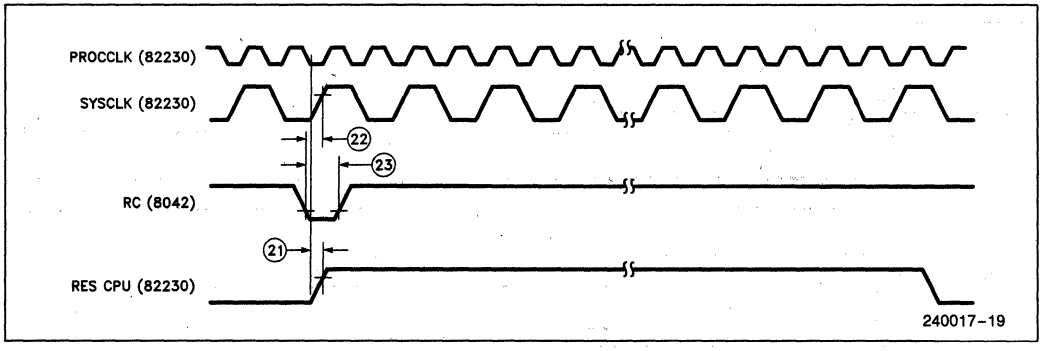


Figure 5. 82230 Keyboard Initiated Reset

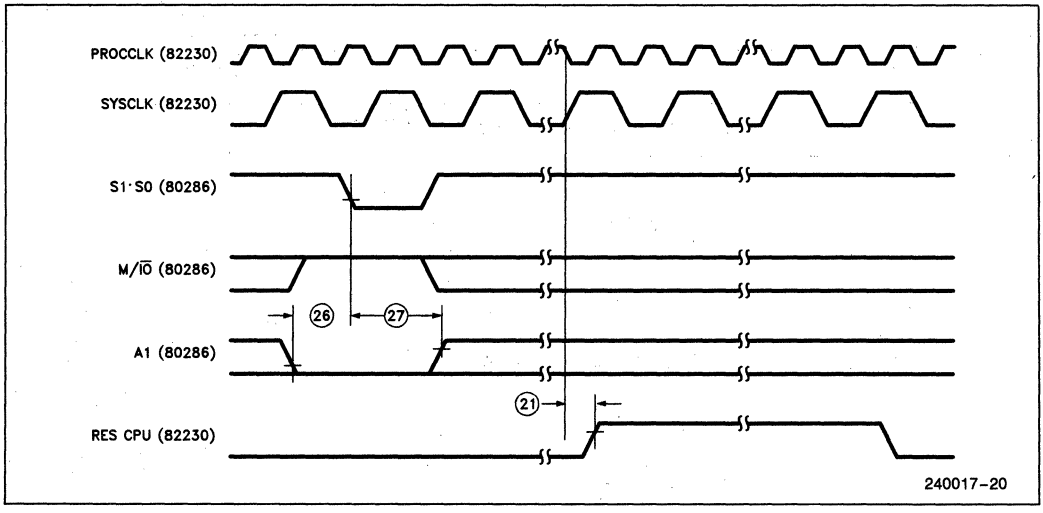


Figure 6. 82230 Processor Shutdown Initiated Reset

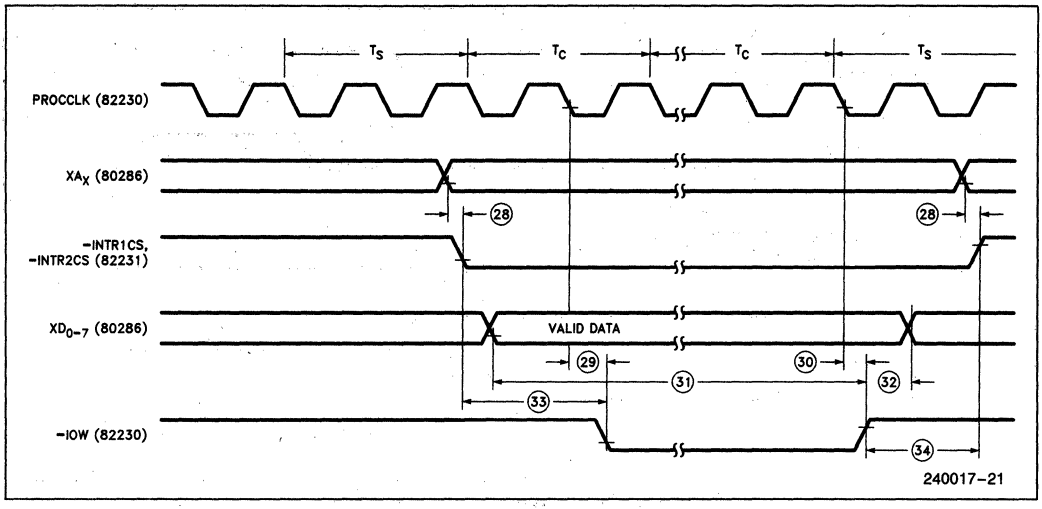


Figure 7. 82230 8254 Bus Write Timing

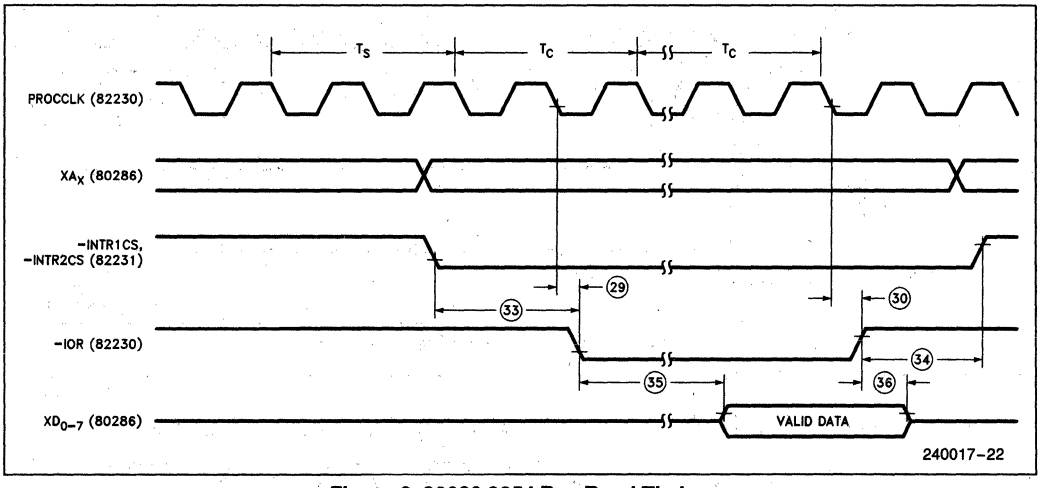


Figure 8. 82230 8254 Bus Read Timing

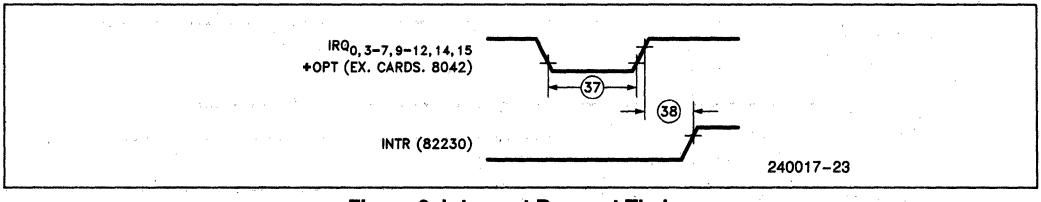


Figure 9. Interrupt Request Timing

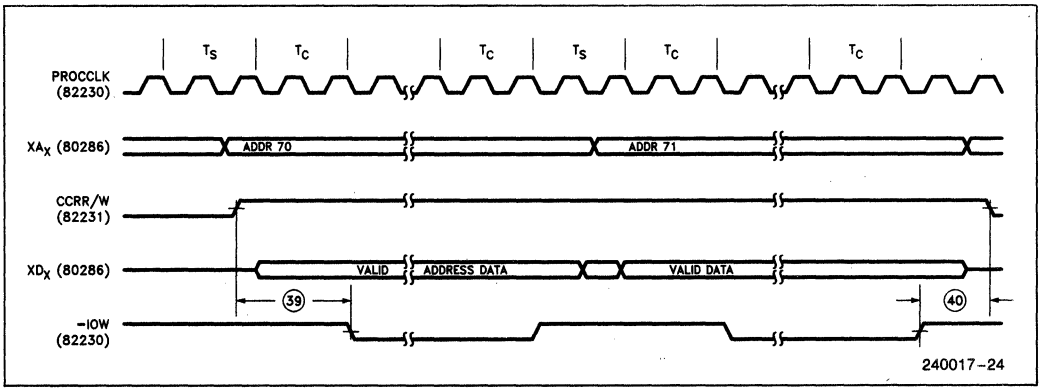


Figure 10. 82230 6818 Write Cycle

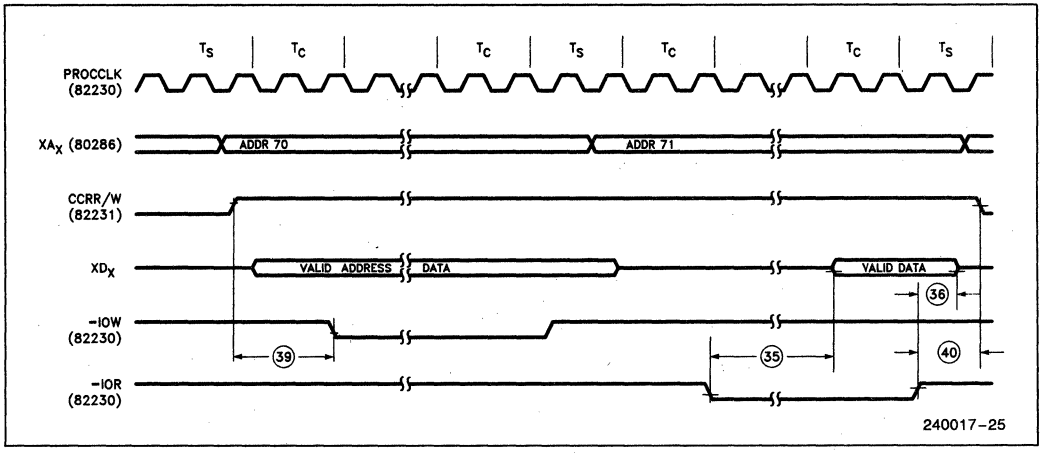


Figure 11. 82230 6818 Read Cycle

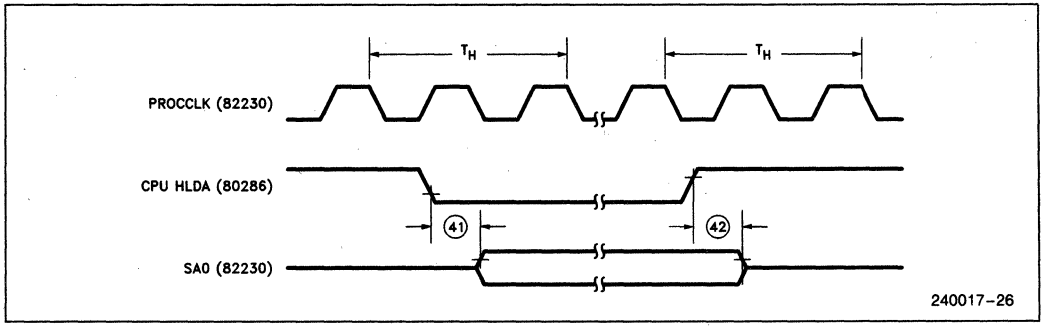


Figure 12. 82230 SA0 Timing

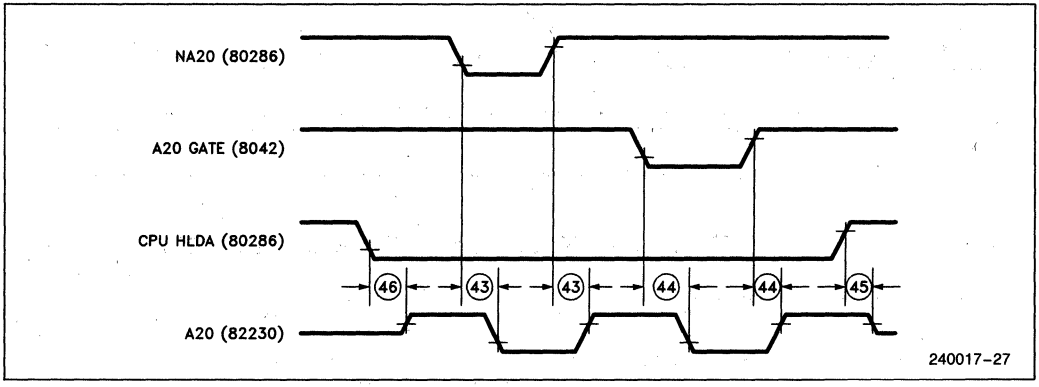


Figure 13. 82230 A20 Timing

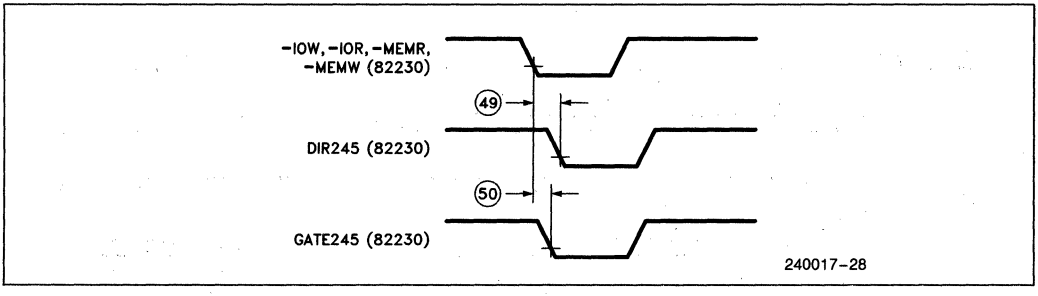


Figure 14. 82230 DIR245, GATE245 Timing

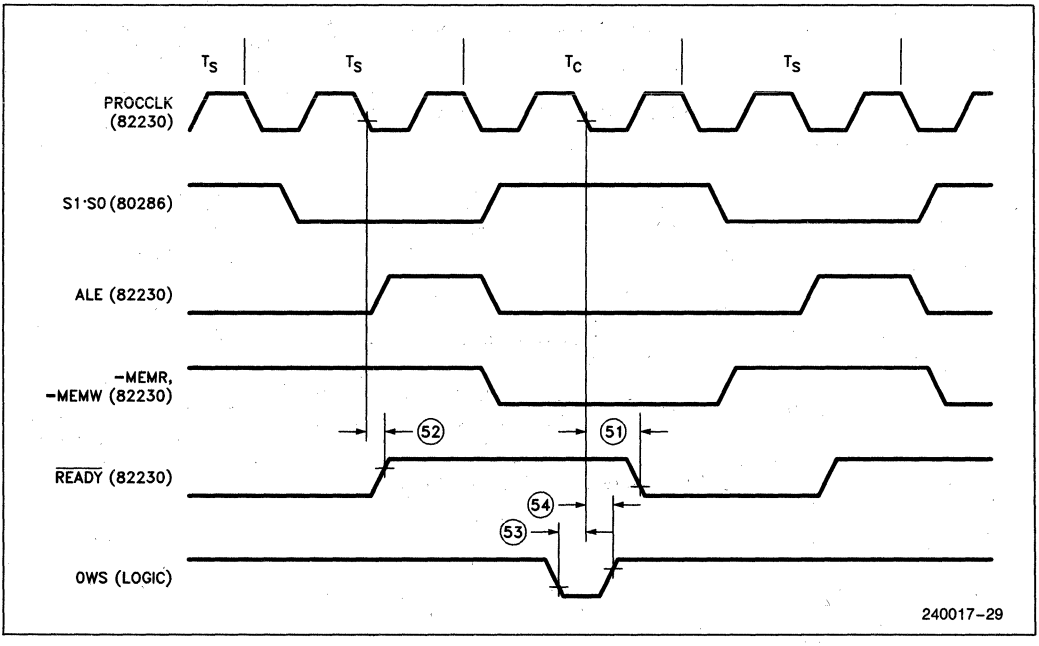


Figure 15. 82230 Zero Wait State Timing

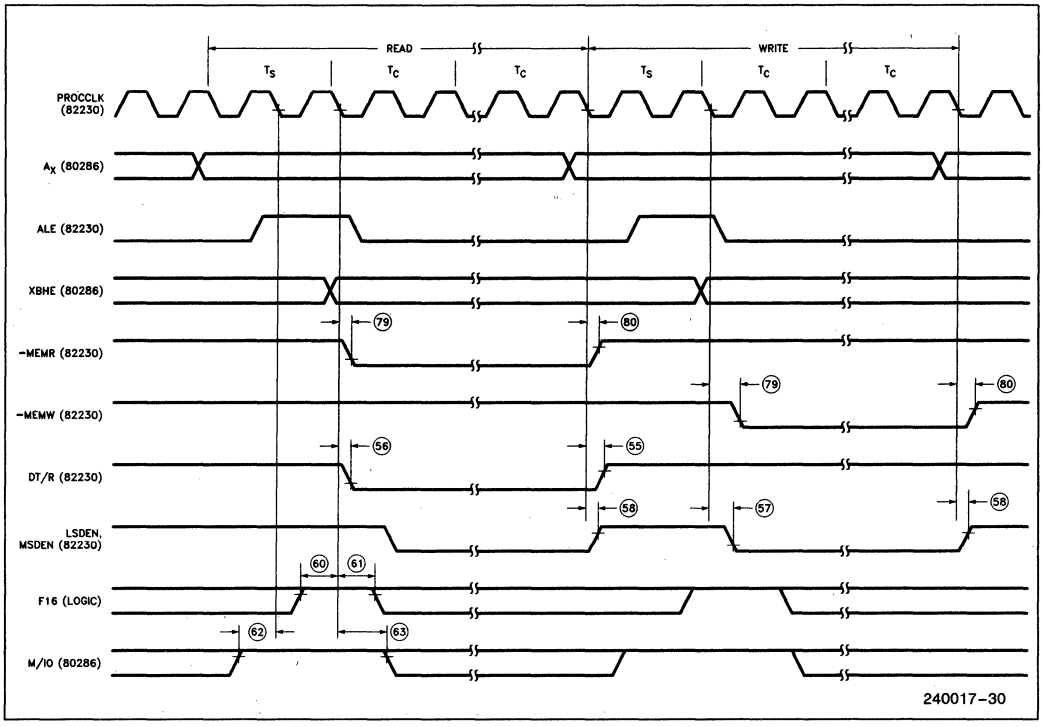


Figure 16. Memory Read/Write Cycles

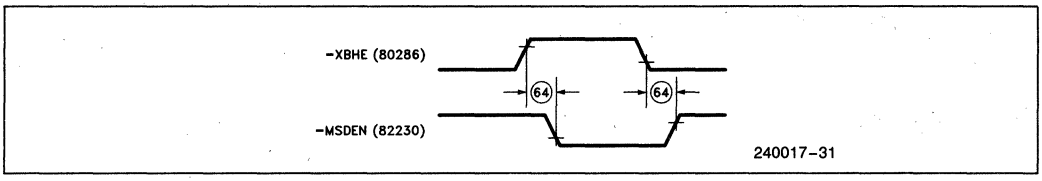


Figure 17. 82230 XHBE Timing

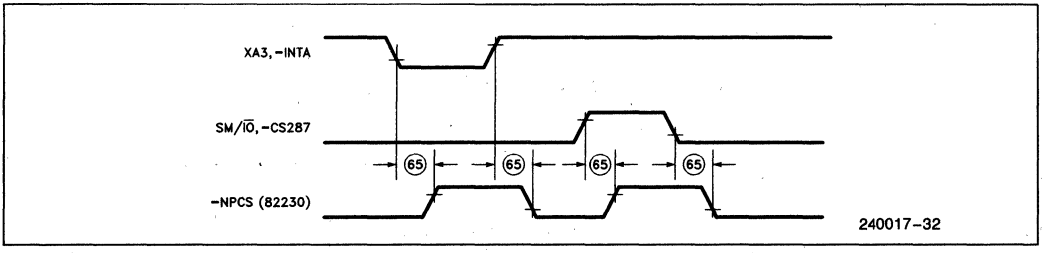


Figure 18. 82230 NPCS Timing

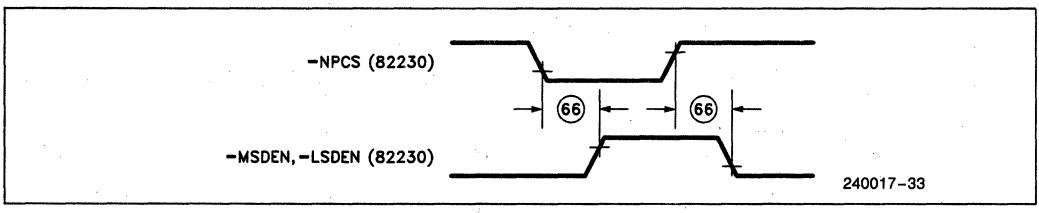


Figure 19. 82230 MSDEN, LSDEN Timing

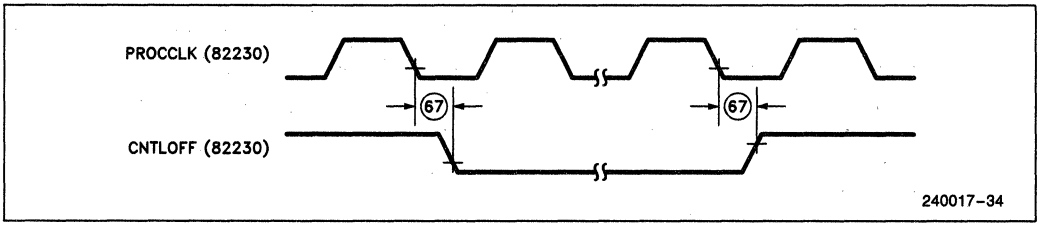


Figure 20. CNTLOFF Timing

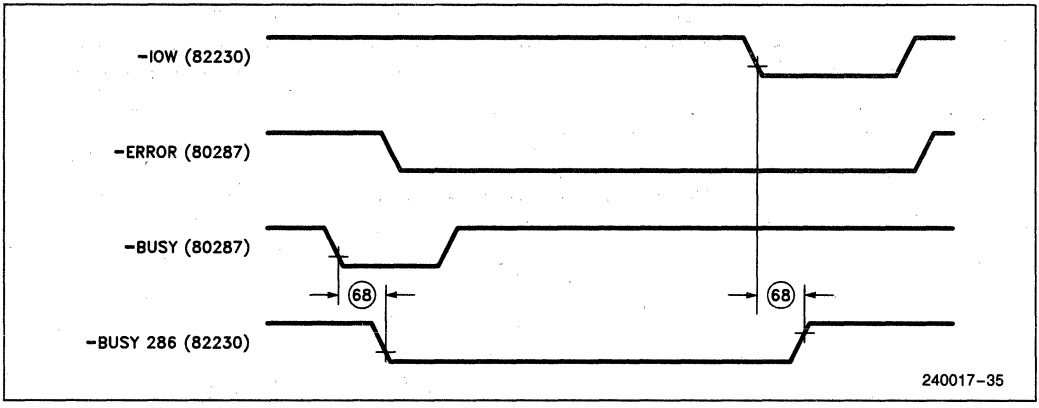


Figure 21. −BUSY286 Timing

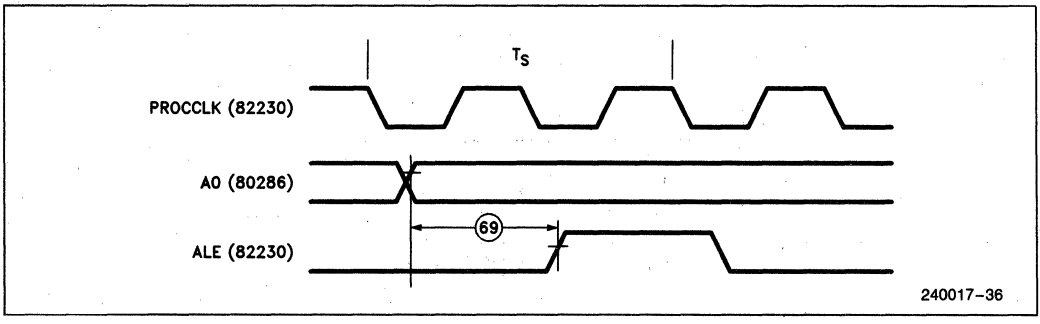


Figure 22. 82230 A0 Timing

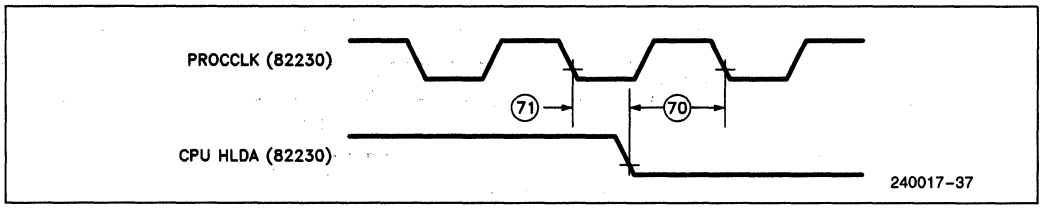


Figure 23. 82230 CPU HLDA Timing

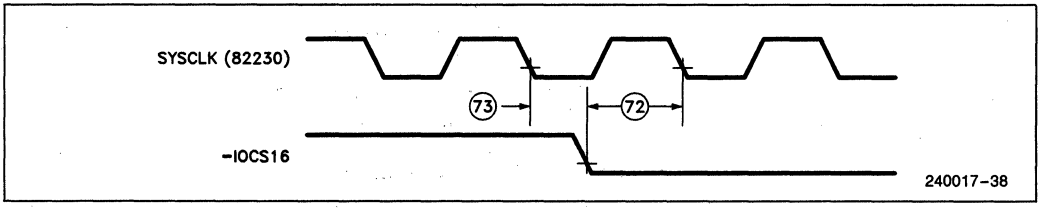


Figure 24. Bus Control Signal Timing

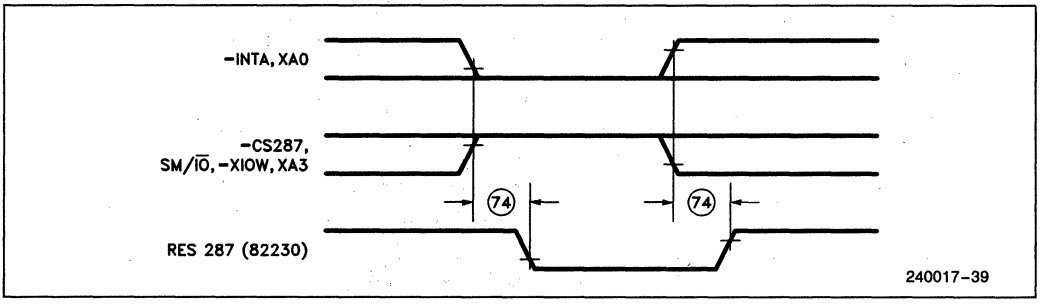


Figure 25. RES 287 Timing

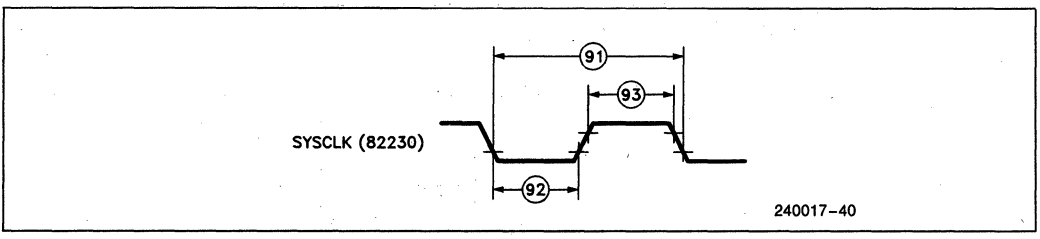


Figure 26. 82231 SYSCLK Timing

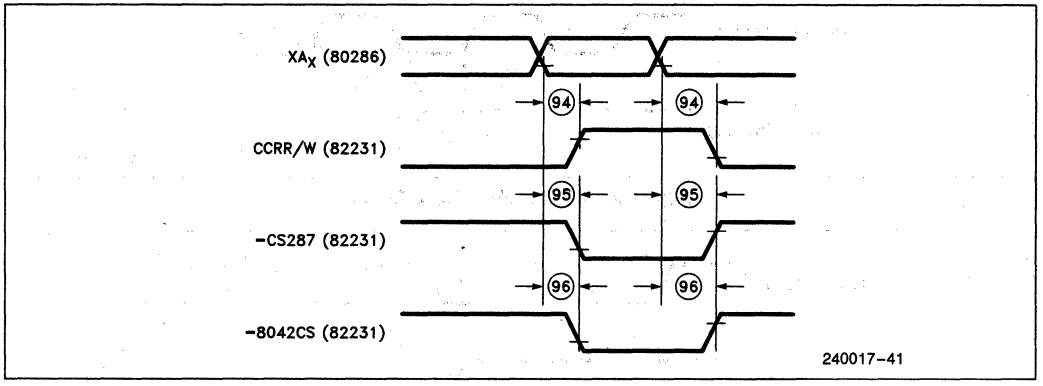


Figure 27. 82231 CCRR/W and CS287 Timing

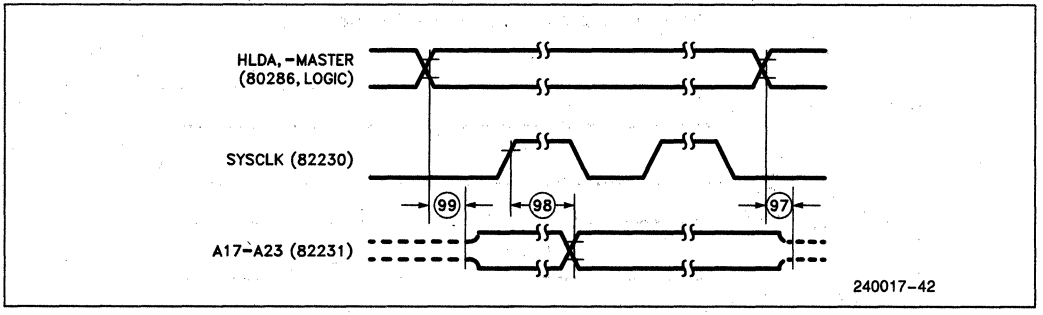


Figure 28. 82231 A17 to A23 Timing

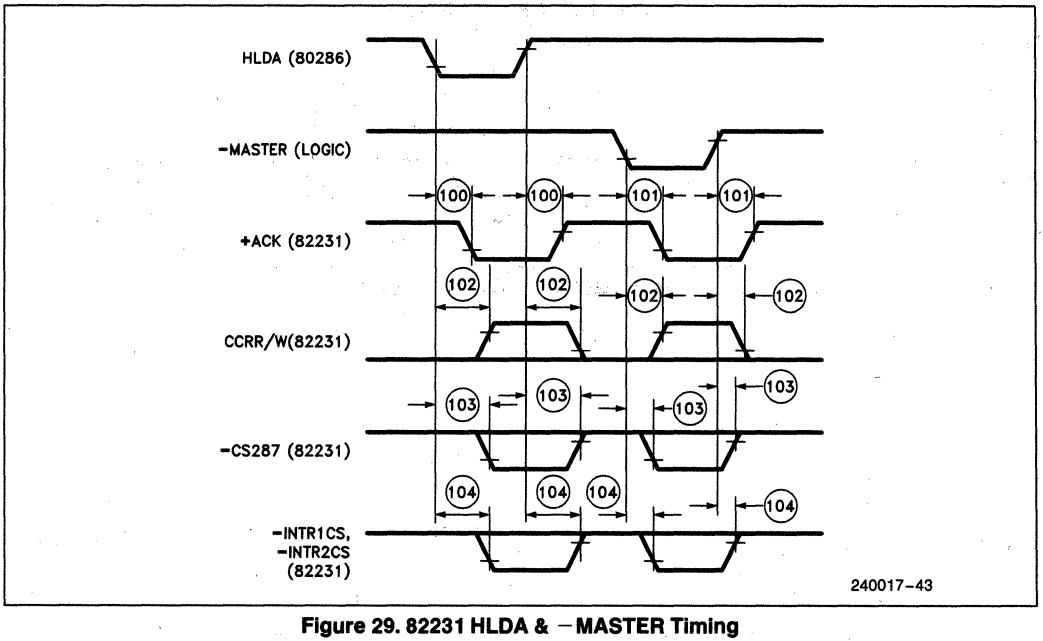


Figure 29. 82231 HLDA & −MASTER Timing

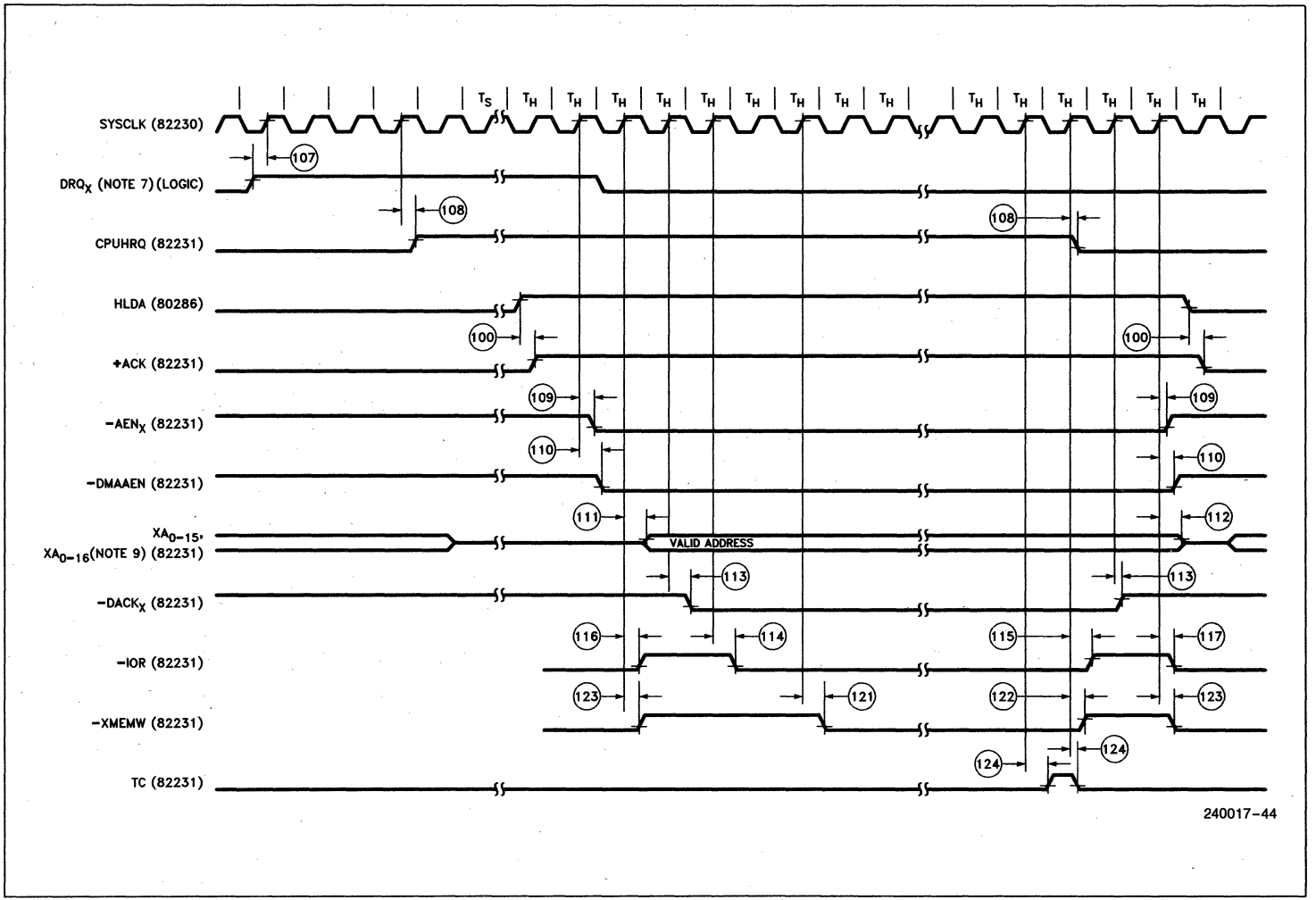


Figure 30. 82231 DMA I/O Read Timing (Single Transfer Shown)

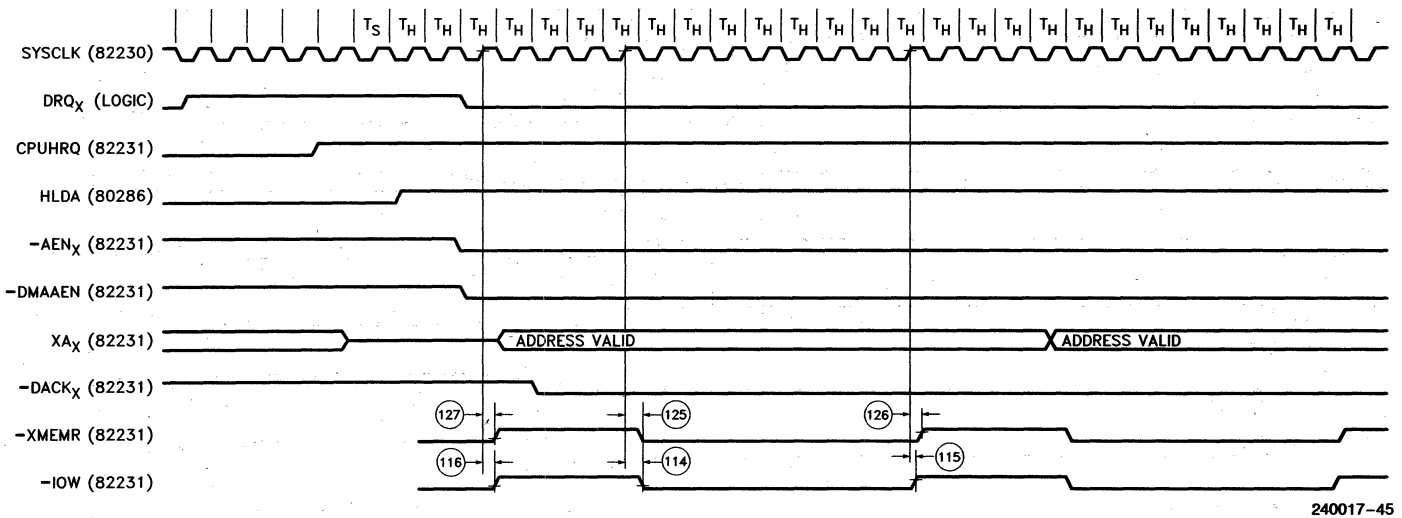


Figure 31. 82231 DMA I/O Write Timing (Block Transfer Shown)

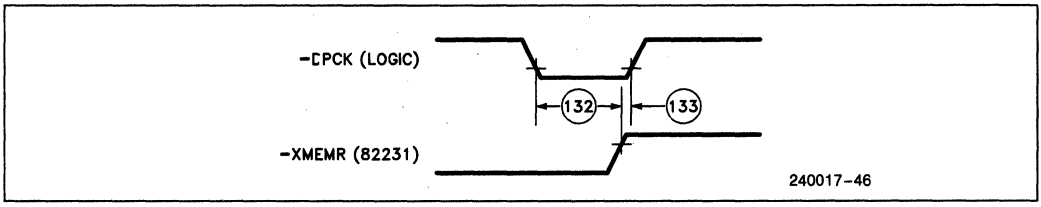


Figure 32. 82231 DPCK Timing

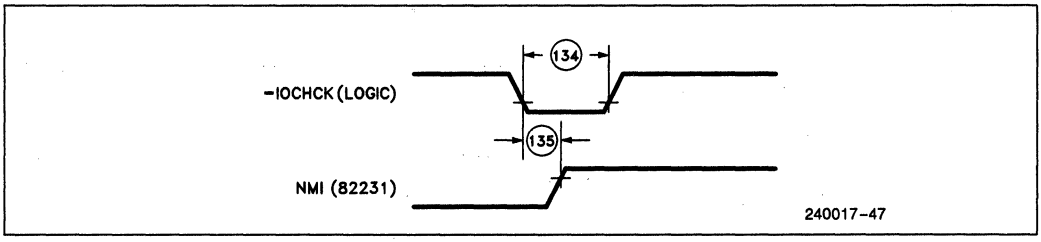


Figure 33. 82231 IOCHCK and NMI Timing

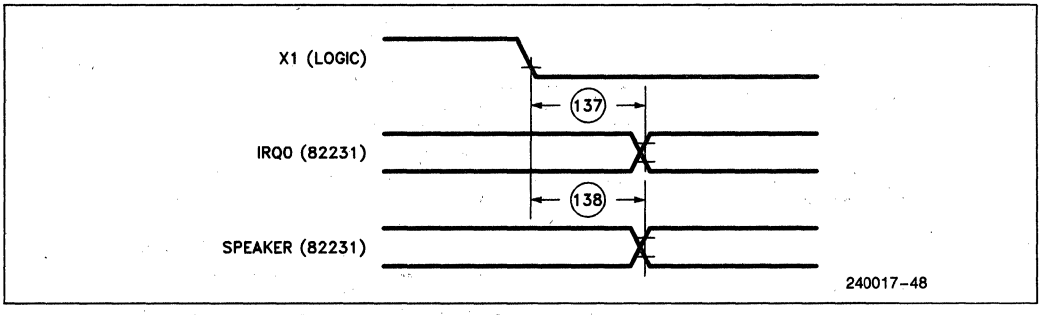


Figure 34. 82231 IRQ0 and SPEAKER Timing

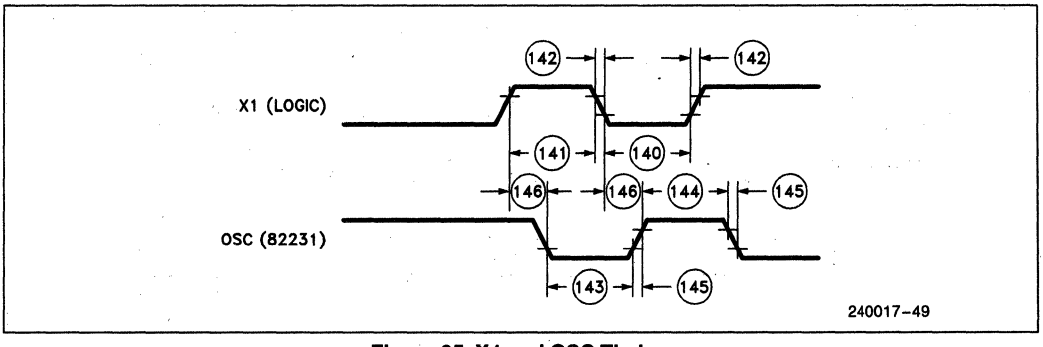


Figure 35. X1 and OSC Timing

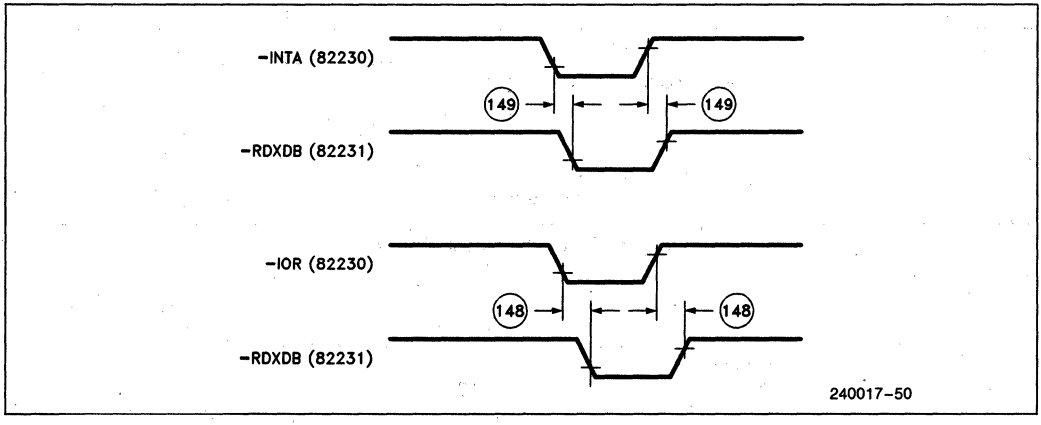


Figure 36. 82231 RDXDB Timing

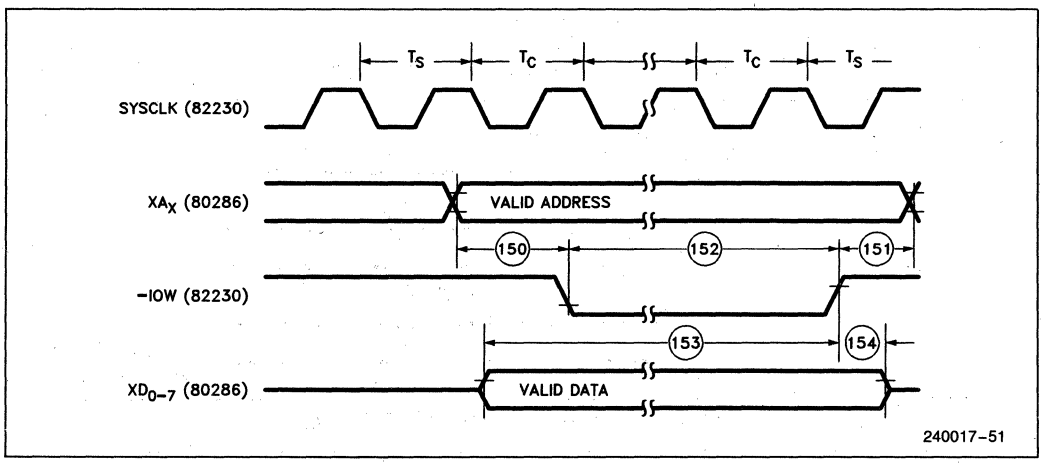


Figure 37. 82231 CPU Write Timing

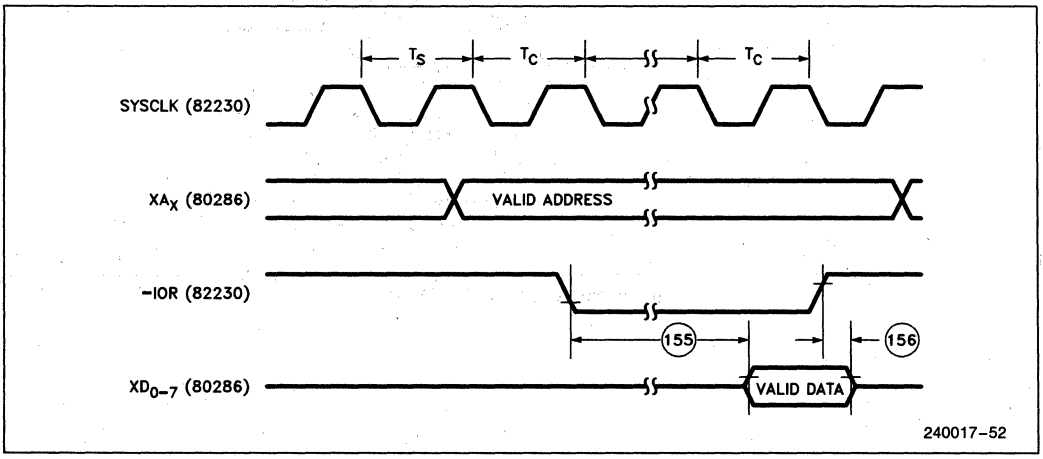


Figure 38. 82231 CPU Read Timing

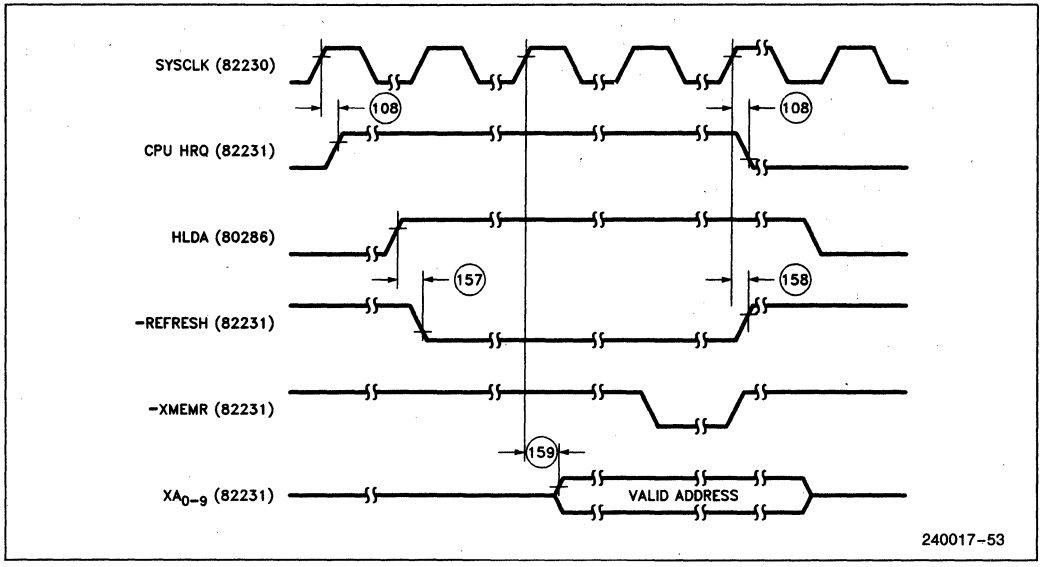
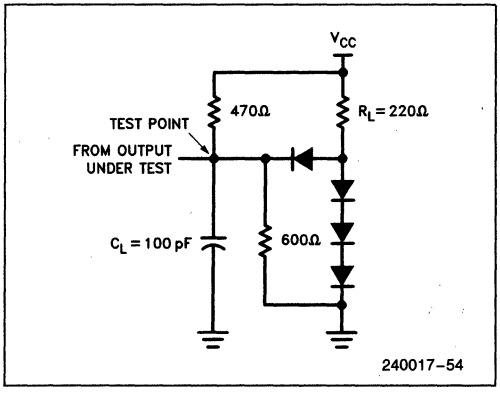


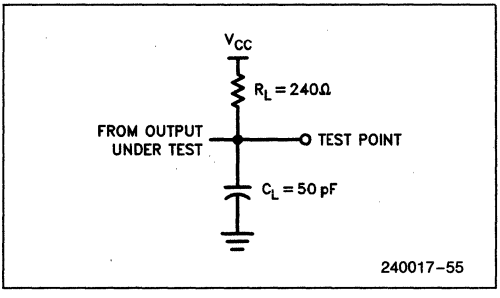
Figure 39. 82231 REFRESH Timing

TEST LOADS

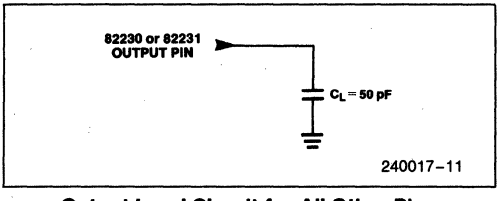


Output Load Circuit

For 82230 pins	SYSCLK
-IOR	-MEMR
-IOW	-MEMW
SA0	
82231 pins -IOR	OSC
-IOW	



Load Circuit for Open-Collector Output



Output Load Circuit for All Other Pins

The following list represents the differences between this and the −001 version of the data sheet.

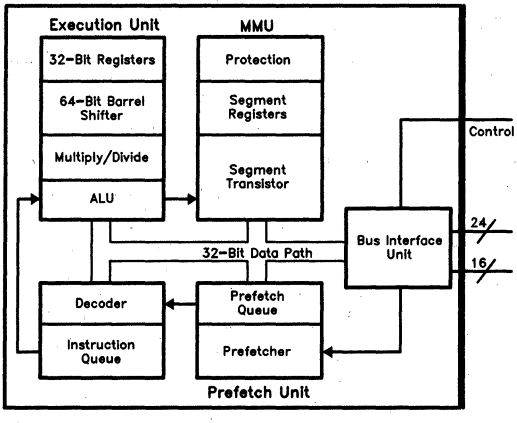
1. Several timing diagrams were added to further detail operating characteristics.
2. The "D.C. CHARACTERISTICS" table was reorganized to clarify the different testing conditions.
3. Extensive changes were made to the "A.C. CHARACTERISTICS" table. These changes include the following:
   - — Timing characteristics were changed to reflect current max and min operating values.
   - — In cases where different specs for the same signal were identical, they were combined into one spec.
   - — In cases where combined specs for signals were found to differ, they were split.
   - — Specs necessary for system design were added.
   - — Unnecessary or irrelevant specs were deleted.
   - — Notes were added, deleted or changed as necessary to correspond to the updated table.
4. The test load diagram was changed.

376™ HIGH PERFORMANCE 32-BIT EMBEDDED PROCESSOR

- **Full 32-Bit Internal Architecture**
  - **8-, 16-, 32-Bit Data Types**
  - **8 General Purpose 32-Bit Registers**
  - **Extensive 32-Bit Instruction Set**
- **High Performance 16-Bit Data Bus**
  - **16 MHz CPU Clock**
  - **Two-Clock Bus Cycles**
  - **16 Mbytes/Sec Bus Bandwidth**
- **16 Mbyte Physical Memory Size**
- **High Speed Numerics Support with the 80387SX**
- **Low System Cost with the 82370 Integrated System Peripheral**
- **On-Chip Debugging Support Including Break Point Registers**
- **Complete Intel Development Support**
  - **C, PL/M, Assembler Translators**
  - **ICE™-376, In-Circuit Emulator**
  - **iRMK Real Time Kernel**
- **Extensive Third-Party Support:**
  - **Software: C, Pascal, FORTRAN, BASIC and ADA***
  - **Hosts: VMS*, UNIX*, MS-DOS*, and Others**
  - **Real-Time Kernels**
- **High Speed CHMOS Technology**
- **Available in 100 Pin Plastic Quad Flat-Pack Package and 88-Pin Pin Grid Array**
  (See Packaging Outlines and Dimensions #231369)

INTRODUCTION

The 376 32-bit embedded processor is designed for high performance embedded systems. It provides the performance benefits of a highly pipelined 32-bit internal architecture with the low system cost associated with 16-bit hardware systems. The 80376 is based on the 80386 and offers a high degree of compatibility with the 80386. All 80386 32-bit programs not dependent on paging can be executed on the 80376 and all 80376 programs can be executed on the 80386. All 32-bit 80386 language translators can be used for software development. With proper support software, any 80386-based computer can be used to develop and test 80376 programs. In addition, any 80386-based PC-AT* compatible computer can be used for hardware prototyping for designs based on the 80376 and its companion product the 82370.



240182-48

80376 Microarchitecture

*UNIX is a registered trademark of AT&T.
ADA is a registered trademark of the U.S. Government, Ada Joint Program Office.
PC-AT is a registered trademark of IBM Corporation.
VMS is a trademark of Digital Equipment Corporation.
MS-DOS is a trademark of MicroSoft Corporation.

August 1988
Order Number: 240182-002

1.0 PIN DESCRIPTION

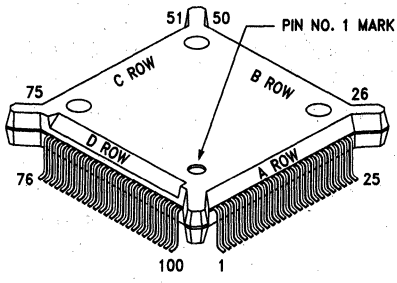


Figure 1.1. 80376 100-Pin Quad Flat-Pack Pin Out (Top View)

Table 1.1. 100-Pin Plastic Quad Flat-Pack Pin Assignments

A Row		B Row		C Row		D Row	
Pin	**Label**	**Pin**	**Label**	**Pin**	**Label**	**Pin**	**Label**
1	D_0	26	LOCK#	51	A_2	76	A_{21}
2	V_{SS}	27	N/C	52	A_3	77	V_{SS}
3	HLDA	28	N/C	53	A_4	78	V_{SS}
4	HOLD	29	N/C	54	A_5	79	A_{22}
5	V_{SS}	30	N/C	55	A_6	80	A_{23}
6	NA#	31	N/C	56	A_7	81	D_{15}
7	READY#	32	V_{CC}	57	V_{CC}	82	D_{14}
8	V_{CC}	33	RESET	58	A_8	83	D_{13}
9	V_{CC}	34	BUSY#	59	A_9	84	V_{CC}
10	V_{CC}	35	V_{SS}	60	A_{10}	85	V_{SS}
11	V_{SS}	36	ERROR#	61	A_{11}	86	D_{12}
12	V_{SS}	37	PEREQ	62	A_{12}	87	D_{11}
13	V_{SS}	38	NMI	63	V_{SS}	88	D_{10}
14	V_{SS}	39	V_{CC}	64	A_{13}	89	D_9
15	CLK2	40	INTR	65	A_{14}	90	D_8
16	ADS#	41	V_{SS}	66	A_{15}	91	V_{CC}
17	BLE#	42	V_{CC}	67	V_{SS}	92	D_7
18	A_1	43	N/C	68	V_{SS}	93	D_6
19	BHE#	44	N/C	69	V_{CC}	94	D_5
20	N/C	45	N/C	70	A_{16}	95	D_4
21	V_{CC}	46	N/C	71	V_{CC}	96	D_3
22	V_{SS}	47	N/C	72	A_{17}	97	V_{CC}
23	M/IO#	48	V_{CC}	73	A_{18}	98	V_{SS}
24	D/C#	49	V_{SS}	74	A_{19}	99	D_2
25	W/R#	50	V_{SS}	75	A_{20}	100	D_1

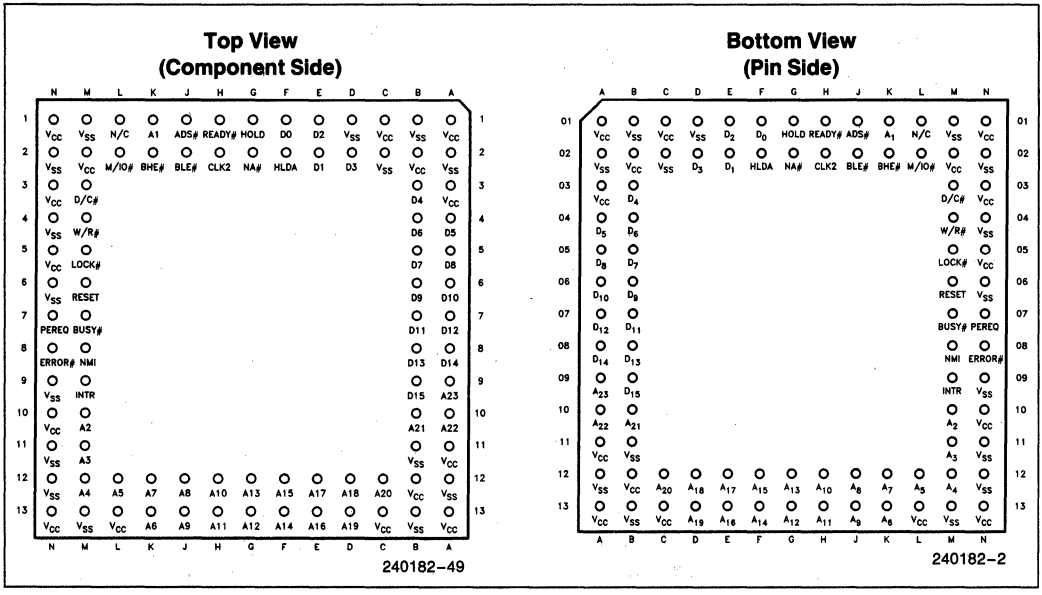


Figure 1.2. 80376 88-Pin Grid Array Pin Out

Table 1.2. 88-Pin Grid Array Pin Assignments

Pin	Label	Pin	Label	Pin	Label	Pin	Label
2H	CLK2	12D	A_{18}	2L	M/IO#	11A	V_{CC}
9B	D_{15}	12E	A_{17}	5M	LOCK#	13A	V_{CC}
8A	D_{14}	13E	A_{16}	1J	ADS#	13C	V_{CC}
8B	D_{13}	12F	A_{15}	1H	READY#	13L	V_{CC}
7A	D_{12}	13F	A_{14}	2G	NA#	1N	V_{CC}
7B	D_{11}	12G	A_{13}	1G	HOLD	13N	V_{CC}
6A	D_{10}	13G	A_{12}	2F	HLDA	11B	V_{SS}
6B	D_9	13H	A_{11}	7N	PEREQ	2C	V_{SS}
5A	D_8	12H	A_{10}	7M	BUSY#	1D	V_{SS}
5B	D_7	13J	A_9	8N	ERROR#	1M	V_{SS}
4B	D_6	12J	A_8	9M	INTR	4N	V_{SS}
4A	D_5	12K	A_7	8M	NMI	9N	V_{SS}
3B	D_4	13K	A_6	6M	RESET	11N	V_{SS}
2D	D_3	12L	A_5	2B	V_{CC}	2A	V_{SS}
1E	D_2	12M	A_4	12B	V_{CC}	12A	V_{SS}
2E	D_1	11M	A_3	1C	V_{CC}	1B	V_{SS}
1F	D_0	10M	A_2	2M	V_{CC}	13B	V_{SS}
9A	A_{23}	1K	A_1	3N	V_{CC}	13M	V_{SS}
10A	A_{22}	2J	BLE#	5N	V_{CC}	2N	V_{SS}
10B	A_{21}	2K	BHE#	10N	V_{CC}	6N	V_{SS}
12C	A_{20}	4M	W/R#	1A	V_{CC}	12N	V_{SS}
13D	A_{19}	3M	D/C#	3A	V_{CC}	1L	N/C

The following table lists a brief description of each pin on the 80376. The following definitions are used in these descriptions:

- # The named signal is active LOW.
- I Input signal.
- O Output signal.
- I/O Input and Output signal.
- — No electrical connection.

Symbol	Type	Name and Function
CLK2	I	**CLK2** provides the fundamental timing for the 80376. For additional information see **Clock** (page 33).
RESET	I	**RESET** suspends any operation in progress and places the 80376 in a known reset state. See **Interrupt Signals** (page 38) for additional information.
D_{15}–D_0	I/O	**DATA BUS** inputs data during memory, I/O and interrupt acknowledge read cycles and outputs data during memory and I/O write cycles. See **Data Bus** (page 34) for additional information.
A_{23}–A_1	O	**ADDRESS BUS** outputs physical memory or port I/O addresses. See **Address Bus** (page 34) for additional information.
W/R#	O	**WRITE/READ** is a bus cycle definition pin that distinguishes write cycles from read cycles. See **Bus Cycle Definition Signals** (page 35) for additional information.
D/C#	O	**DATA/CONTROL** is a bus cycle definition pin that distinguishes data cycles, either memory or I/O, from control cycles which are: interrupt acknowledge, halt, and instruction fetching. See **Bus Cycle Definition Signals** (page 35) for additional information.
M/IO#	O	**MEMORY I/O** is a bus cycle definition pin that distinguishes memory cycles from input/output cycles. See **Bus Cycle Definition Signals** (page 35) for additional information.
LOCK#	O	**BUS LOCK** is a bus cycle definition pin that indicates that other system bus masters are denied access to the system bus while it is active. See **Bus Cycle Definition Signals** (page 35) for additional information.
ADS#	O	**ADDRESS STATUS** indicates that a valid bus cycle definition and address (W/R#, D/C#, M/IO#, BHE#, BLE# and A_{23}–A_1) are being driven at the 80376 pins. See **Bus Control Signals** (page 35) for additional information.
NA#	I	**NEXT ADDRESS** is used to request address pipelining. See **Bus Control Signals** (page 35) for additional information.
READY#	I	**BUS READY** terminates the bus cycle. See **Bus Control Signals** (page 35) for additional information.
BHE#, BLE#	O	**BYTE ENABLES** indicate which data bytes of the data bus take part in a bus cycle. See **Address Bus** (page 34) for additional information.
HOLD	I	**BUS HOLD REQUEST** input allows another bus master to request control of the local bus. See **Bus Arbitration Signals** (page 36) for additional information.

Symbol	Type	Name and Function
HLDA	O	**BUS HOLD ACKNOWLEDGE** output indicates that the 80376 has surrendered control of its local bus to another bus master. See **Bus Arbitration Signals** (page 36) for additional information.
INTR	I	**INTERRUPT REQUEST** is a maskable input that signals the 80376 to suspend execution of the current program and execute an interrupt acknowledge function. See **Interrupt Signals** (page 38) for additional information.
NMI	I	**NON-MASKABLE INTERRUPT REQUEST** is a non-maskable input that signals the 80376 to suspend execution of the current program and execute an interrupt acknowledge function. See **Interrupt Signals** (page 38) for additional information.
BUSY#	I	**BUSY** signals a busy condition from a processor extension. See **Coprocessor Interface Signals** (page 37) for additional information.
ERROR#	I	**ERROR** signals an error condition from a processor extension. See **Coprocessor Interface Signals** (page 37) for additional information.
PEREQ	I	**PROCESSOR EXTENSION REQUEST** indicates that the processor extension has data to be transferred by the 80376. See **Coprocessor Interface Signals** (page 37) for additional information.
N/C	—	**NO CONNECT** should always remain unconnected. Connection of a N/C pin may cause the processor to malfunction or be incompatible with future steppings of the 80376.
V_{CC}	I	**SYSTEM POWER** provides the +5V nominal D.C. supply input.
V_{SS}	I	**SYSTEM GROUND** provides 0V connection from which all inputs and outputs are measured.

2.0 ARCHITECTURE OVERVIEW

The 80376 supports the protection mechanisms needed by sophisticated multitasking embedded systems and real-time operating systems. The use of these protection mechanisms is completely optional. For embedded applications not needing protection, the 80376 can easily be configured to provide a 16 Mbyte physical address space.

Instruction pipelining, high bus bandwidth, and a very high performance ALU ensure short average instruction execution times and high system throughput. The 80376 is capable of execution at sustained rates of 2.5–3.0 million instructions per second.

The 80376 offers on-chip testability and debugging features. Four break point registers allow conditional or unconditional break point traps on code execution or data accesses for powerful debugging of even ROM based systems. Other testability features include self-test and tri-stating of output buffers during RESET.

The Intel 80376 embedded processor consists of a central processing unit, a memory management unit and a bus interface. The central processing unit consists of the execution unit and instruction unit. The execution unit contains the eight 32-bit general registers which are used for both address calculation and data operations and a 64-bit barrel shifter used to speed shift, rotate, multiply, and divide operations. The instruction unit decodes the instruction opcodes and stores them in the decoded instruction queue for immediate use by the execution unit.

The Memory Management Unit (MMU) consists of a segmentation and protection unit. Segmentation allows the managing of the logical address space by providing an extra addressing component, one that allows easy code and data relocatability, and efficient sharing.

The protection unit provides four levels of protection for isolating and protecting applications and the operating system from each other. The hardware enforced protection allows the design of systems with a high degree of integrity and simplifies debugging.

Finally, to facilitate high performance system hardware designs, the 80376 bus interface offers address pipelining and direct Byte Enable signals for each byte of the data bus.

2.1 Register Set

The 80376 has twenty-nine registers as shown in Figure 2.1. These registers are grouped into the following six categories:

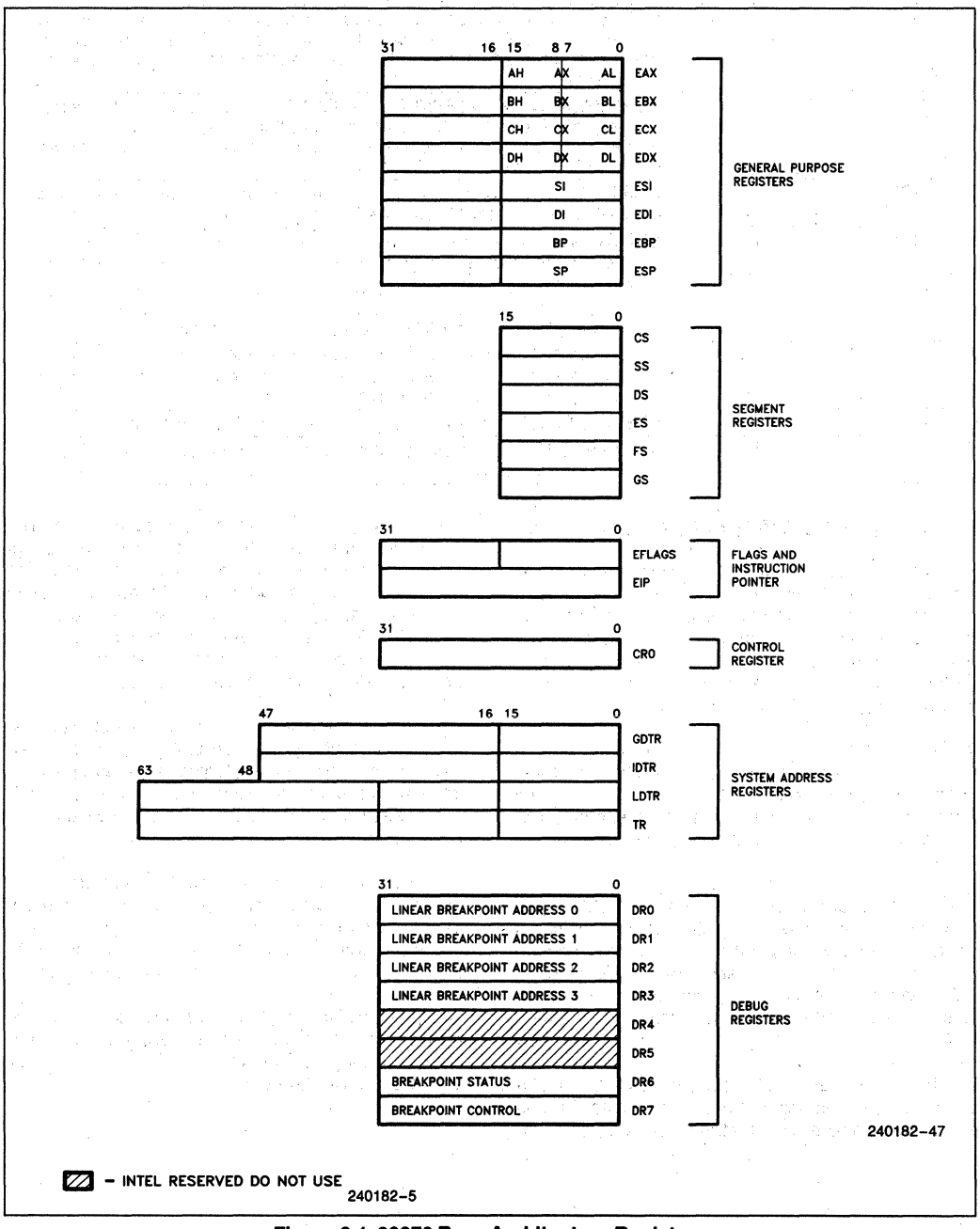


Figure 2.1. 80376 Base Architecture Registers

General Registers: The eight 32-bit general purpose registers are used to contain arithmetic and logical operands. Four of these (EAX, EBX, ECX and EDX) can be used either in their entirety as 32-bit registers, as 16-bit registers, or split into pairs of separate 8-bit registers.

Segment Registers: Six 16-bit special purpose registers select, at any given time, the segments of memory that are immediately addressable for code, stack, and data.

Flags and Instruction Pointer Registers: These two 32-bit special purpose registers in Figure 2.1 record or control certain aspects of the 80376 processor state. The EFLAGS register includes status and control bits that are used to reflect the outcome of many instructions and modify the semantics of some instructions. The Instruction Pointer, called EIP, is 32 bits wide. The Instruction Pointer controls instruction fetching and the processor automatically increments it after executing an instruction.

Control Register: The 32-bit control register, CR0, is used to control Coprocessor Emulation.

System Address Registers: These four special registers reference the tables or segments supported by the 80376/80386 protection model. These tables or segments are:

GDTR (Global Descriptor Table Register),
IDTR (Interrupt Descriptor Table Register),
LDTR (Local Descriptor Table Register),
TR (Task State Segment Register).

Debug Registers: The six programmer accessible debug registers provide on-chip support for debugging. The use of the debug registers is described in Section 2.11 **Debugging Support**.

EFLAGS REGISTER

The flag Register is a 32-bit register named EFLAGS. The defined bits and bit fields within EFLAGS, shown in Figure 2.2, control certain operations and indicate the status of the 80376 processor. The function of the flag bits is given in Table 2.1.

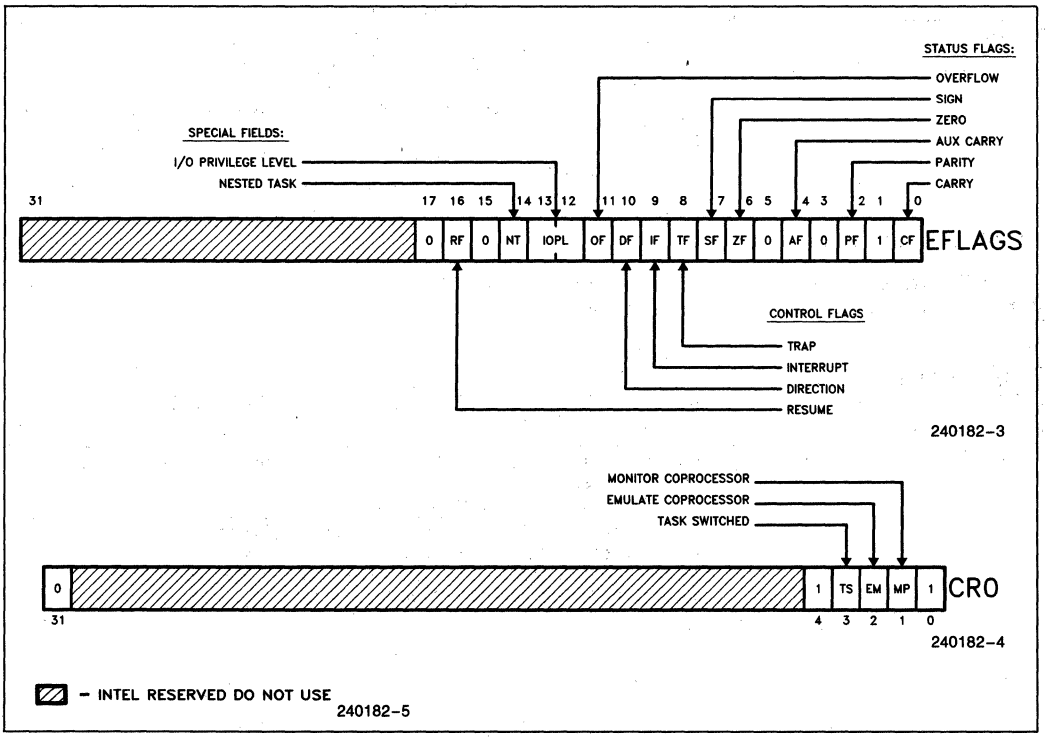


Figure 2.2. Status and Control Register Bit Functions

Table 2.1. Flag Definitions

Bit Position	Name	Function
0	CF	**Carry Flag**—Set on high-order bit carry or borrow; cleared otherwise.
2	PF	**Parity Flag**—Set if low-order 8 bits of result contain an even number of 1-bits; cleared otherwise.
4	AF	**Auxiliary Carry Flag**—Set on carry from or borrow to the low order four bits of AL; cleared otherwise.
6	ZF	**Zero Flag**—Set if result is zero; cleared otherwise.
7	SF	**Sign Flag**—Set equal to high-order bit of result (0 if positive, 1 if negative).
8	TF	**Single Step Flag**—Once set, a single step interrupt occurs after the next instruction executes. TF is cleared by the single step interrupt.
9	IF	**Interrupt-Enable Flag**—When set, external interrupts signaled on the INTR pin will cause the CPU to transfer control to an interrupt vector specified location.
10	DF	**Direction Flag**—Causes string instructions to auto-increment (default) the appropriate index registers when cleared. Setting DF causes auto-decrement.
11	OF	**Overflow Flag**—Set if the operation resulted in a carry/borrow into the sign bit (high-order bit) of the result but did not result in a carry/borrow out of the high-order bit or vice-versa.
12, 13	IOPL	**I/O Privilege Level**—Indicates the maximum CPL permitted to execute I/O instructions without generating an exception 13 fault or consulting the I/O permission bit map. It also indicates the maximum CPL value allowing alteration of the IF bit.
14	NT	**Nested Task**—Indicates that the execution of the current task is nested within another task (see **Task Switching**).
16	RF	**Resume Flag**—Used in conjunction with debug register breakpoints. It is checked at instruction boundaries before breakpoint processing. If set, any debug fault is ignored on the next instruction. It is reset at the successful completion of any instruction except IRET, POPF, and those instructions causing task switches.

CONTROL REGISTER

The 80376 has a 32-bit control register called CR0 that is used to control coprocessor emulation. This register is shown in Figures 2.1 and 2.2. The defined CR0 bits are described in Table 2.2.

Table 2.2. CR0 Definitions

Bit Position	Name	Function
1	MP	**Monitor Coprocessor Extension**—Allows WAIT instructions to cause a processor extension not present exception (number 7).
2	EM	**Emulate Processor Extension**—When set, this bit causes a processor extension not present exception (number 7) on ESC instructions to allow processor extension emulation.
3	TS	**Task Switched**—When set, this bit indicates the next instruction using a processor extension will cause exception 7, allowing software to test whether the current processor extension context belongs to the current task (see **Task Switching**).

2.2 Instruction Set

The instruction set is divided into nine categories of operations:

Data Transfer
Arithmetic
Shift/Rotate
String Manipulation
Bit Manipulation
Control Transfer
High Level Language Support
Operating System Support
Processor Control

These 80376 processor instructions are listed in Table 8.1 **80376 Instruction Set and Clock Count Summary**.

All 80376 processor instructions operate on either 0, 1, 2 or 3 operands; an operand resides in a register, in the instruction itself, or in memory. Most zero operand instructions (e.g. CLI, STI) take only one byte. One operand instructions generally are two bytes long. The average instruction is 3.2 bytes long. Since the 80376 has a 16-byte prefetch instruction queue an average of 5 instructions can be prefetched. The use of two operands permits the following types of common instructions:

Register to Register
Memory to Register
Immediate to Register
Memory to Memory
Register to Memory
Immediate to Memory

The operands are either 8-, 16- or 32-bit long.

2.3 Memory Organization

Memory on the 80376 is divided into 8-bit quantities (bytes), 16-bit quantities (words), and 32-bit quantities (dwords). Words are stored in two consecutive bytes in memory with the low-order byte at the lowest address. Dwords are stored in four consecutive bytes in memory with the low-order byte at the lowest address. The address of a word or Dword is the byte address of the low-order byte.

In addition to these basic data types the 80376 processor supports segments. Memory can be divided up into one or more variable length segments, which can be shared between programs.

ADDRESS SPACES

The 80376 has three types of address spaces: **logical, linear,** and **physical.** A **logical** address (also known as a **virtual** address) consists of a selector and an offset. A selector is the contents of a segment register. An offset is formed by summing all of the addressing components (BASE, INDEX, and DISPLACEMENT), discussed in Section 2.4 **Addressing Modes,** into an effective address.

Every selector has a **logical base** address associated with it that can be up to 32 bits in length. This 32-bit **logical base** address is added to either a 32-bit offset address or a 16-bit offset address (by using the ***address length prefix***)to form a final 32-bit **linear** address. This final **linear** address is then truncated so that only the lower 24 bits of this address are used to address the 16 Mbytes physical memory address space. The **logical base** address is stored in one of two operating system tables (i.e. the Local Descriptor Table or Global Descriptor Table).

Figure 2.3 shows the relationship between the various address spaces.

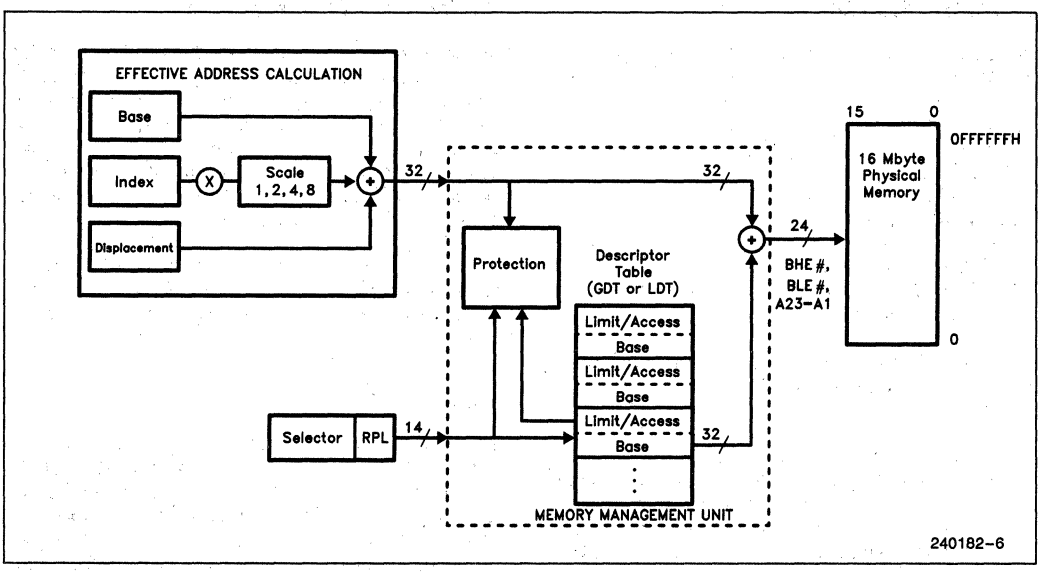


Figure 2.3. Address Translation

SEGMENT REGISTER USAGE

The main data structure used to organize memory is the segment. On the 80376, segments are variable sized blocks of linear addresses which have certain attributes associated with them. There are two main types of segments, code and data. The simplest use of segments is to have one code and data segment. Each segment is 16 Mbytes in size overlapping each other. This allows code and data to be directly addressed by the same offset.

In order to provide compact instruction encoding and increase processor performance, instructions do not need to explicitly specify which segment register is used. The segment register is automatically chosen according to the rules of Table 2.3 (Segment Register Selection Rules). In general, data references use the selector contained in the DS register, stack references use the SS register and instruction fetches use the CS register. The contents of the Instruction Pointer provide the offset. Special segment override prefixes allow the explicit use of a given segment register, and override the implicit rules listed in Table 2.3. The override prefixes also allow the use of the ES, FS and GS segment registers.

There are no restrictions regarding the overlapping of the base addresses of any segments. Thus, all 6 segments could have the base address set to zero. Further details of segmentation are discussed in Section 3.0 Architecture.

Table 2.3. Segment Register Selection Rules

Type of Memory Reference	Implied (Default) Segment Use	Segment Override Prefixes Possible
Code Fetch	CS	None
Destination of PUSH, PUSHF, INT, CALL, PUSHA Instructions	SS	None
Source of POP, POPA, POPF, IRET, RET Instructions	SS	None
Destination of STOS, MOVS, REP STOS, REP MOVS Instructions (DI is Base Register)	ES	None
Other Data References, with Effective Address Using Base Register of:		
[EAX]	DS	CS, SS, ES, FS, GS
[EBX]	DS	CS, SS, ES, FS, GS
[ECX]	DS	CS, SS, ES, FS, GS
[EDX]	DS	CS, SS, ES, FS, GS
[ESI]	DS	CS, SS, ES, FS, GS
[EDI]	DS	CS, SS, ES, FS, GS
[EBP]	SS	CS, SS, ES, FS, GS
[ESP]	SS	CS, SS, ES, FS, GS

2.4 Addressing Modes

The 80376 provides a total of 8 addressing modes for instructions to specify operands. The addressing modes are optimized to allow the efficient execution of high level languages such as C and FORTRAN, and they cover the vast majority of data references needed by high-level languages.

Two of the addressing modes provide for instructions that operate on register or immediate operands:

Register Operand Mode: The operand is located in one of the 8-, 16- or 32-bit general registers.

Immediate Operand Mode: The operand is included in the instruction as part of the opcode.

The remaining 6 modes provide a mechanism for specifying the effective address of an operand. The linear address consists of two components: the segment base address and an effective address. The effective address is calculated by summing any combination of the following three address elements (see Figure 2.3):

DISPLACEMENT: an 8-, 16- or 32-bit immediate value following the instruction.

BASE: The contents of any general purpose register. The base registers are generally used by compilers to point to the start of the local variable area. Note that if the ***Address Length Prefix*** is used, only BX and BP can be used as a BASE register.

INDEX: The contents of any general purpose register except for ESP. The index registers are used to access the elements of an array, or a string of characters. The index register's value can be multiplied by a scale factor, either 1, 2, 4 or 8. The scaled index is especially useful for accessing arrays or structures. Note that if the ***Address Length Prefix*** is used, no Scaling is available and only the registers SI and DI can be used to INDEX.

Combinations of these 3 components make up the 6 additional addressing modes. There is no performance penalty for using any of these addressing combinations, since the effective address calculation is pipelined with the execution of other instructions. The one exception is the simultaneous use of BASE and INDEX components which requires one additional clock.

As shown in Figure 2.4, the effective address (EA) of an operand is calculated according to the following formula:

$$EA = BASE_{Register} + (INDEX_{Register} \times scaling) + DISPLACEMENT$$

1. **Direct Mode:** The operand's offset is contained as part of the instruction as an 8-, 16- or 32-bit DISPLACEMENT.
2. **Register Indirect Mode:** A BASE register contains the address of the operand.
3. **Based Mode:** A BASE register's contents is added to a DISPLACEMENT to form the operand's offset.
4. **Scaled Index Mode:** An INDEX register's contents is multiplied by a SCALING factor which is added to a DISPLACEMENT to form the operand's offset.
5. **Based Scaled Index Mode:** The contents of an INDEX register is multiplied by a SCALING factor and the result is added to the contents of a BASE register to obtain the operand's offset.
6. **Based Scaled Index Mode with Displacement:** The contents of an INDEX register are multiplied by a SCALING factor, and the result is added to the contents of a BASE register and a DISPLACEMENT to form the operand's offset.

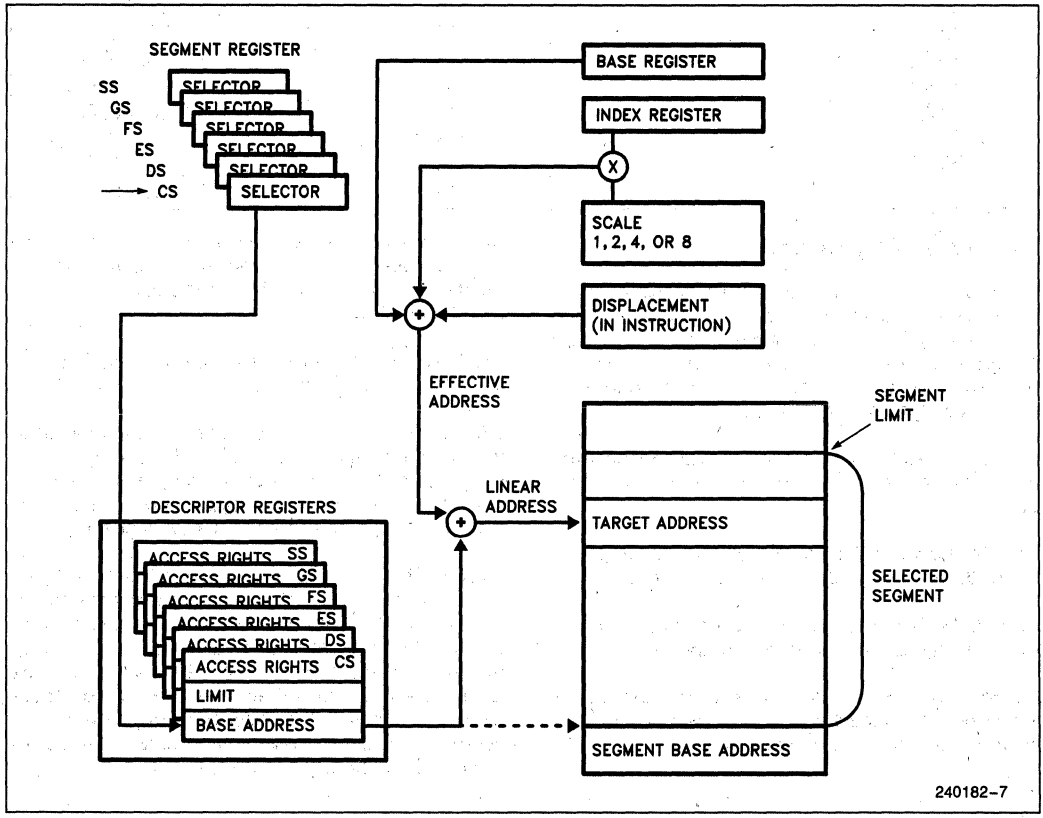


Figure 2.4. Addressing Mode Calculations

GENERATING 16-BIT ADDRESSES

The 80376 executes code with a default length for operands and addresses of 32 bits. The 80376 is also able to execute operands and addresses of 16 bits. This is specified through the use of override prefixes. Two prefixes, the **Operand Length Prefix** and the **Address Length Prefix,** override the default 32-bit length on an individual instruction basis. These prefixes are automatically added by assemblers. The Operand Length and Address Length Prefixes can be applied separately or in combination to any instruction.

The 80376 normally executes 32-bit code and uses either 8- or 32-bit displacements, and any register can be used as based or index registers. When executing 16-bit code (by prefix overrides), the displacements are either 8 or 16 bits, and the base and index register conform to the 16-bit model. Table 2.4 illustrates the differences.

Table 2.4. BASE and INDEX Registers for 16- and 32-Bit Addresses

	16-Bit Addressing	32-Bit Addressing
BASE REGISTER	BX, BP	Any 32-Bit GP Register
INDEX REGISTER	SI, DI	Any 32-Bit GP Register except ESP
SCALE FACTOR	None	1, 2, 4, 8
DISPLACMENT	0, 8, 16 Bits	0, 8, 32 Bits

2.5 Data Types

The 80376 supports all of the data types commonly used in high level languages:

Bit:	A single bit quantity.
Bit Field:	A group of up to 32 contiguous bits, which spans a maximum of four bytes.
Bit String:	A set of contiguous bits, on the 80376 bit strings can be up to 16 Mbits long.
Byte:	A signed 8-bit quantity.
Unsigned Byte:	An unsigned 8-bit quantity.
Integer (Word):	A signed 16-bit quantity.
Long Integer (Double Word):	A signed 32-bit quantity. All operations assume a 2's complement representation.
Unsigned Integer (Word):	An unsigned 16-bit quantity.
Unsigned Long Integer (Double Word):	An unsigned 32-bit quantity.
Signed Quad Word:	A signed 64-bit quantity.
Unsigned Quad Word:	An unsigned 64-bit quantity.
Pointer:	A 16- or 32-bit offset only quantity which indirectly references another memory location.
Long Pointer:	A full pointer which consists of a 16-bit segment selector and either a 16- or 32-bit offset.
Char:	A byte representation of an ASCII Alphanumeric or control character.
String:	A contiguous sequence of bytes, words or dwords. A string may contain between 1 byte and 16 Mbytes.
BCD:	A byte (unpacked) representation of decimal digits 0–9.
Packed BCD:	A byte (packed) representation of two decimal digits 0–9 storing one digit in each nibble.

When the 80376 is coupled with a numerics Coprocessor such as the 80387SX then the following common Floating Point types are supported.

Floating Point: A signed 32-, 64- or 80-bit real number representation. Floating point numbers are supported by the 80387SX numerics coprocessor.

Figure 2.5 illustrates the data types supported by the 80376 processor and the 80387SX coprocessor.

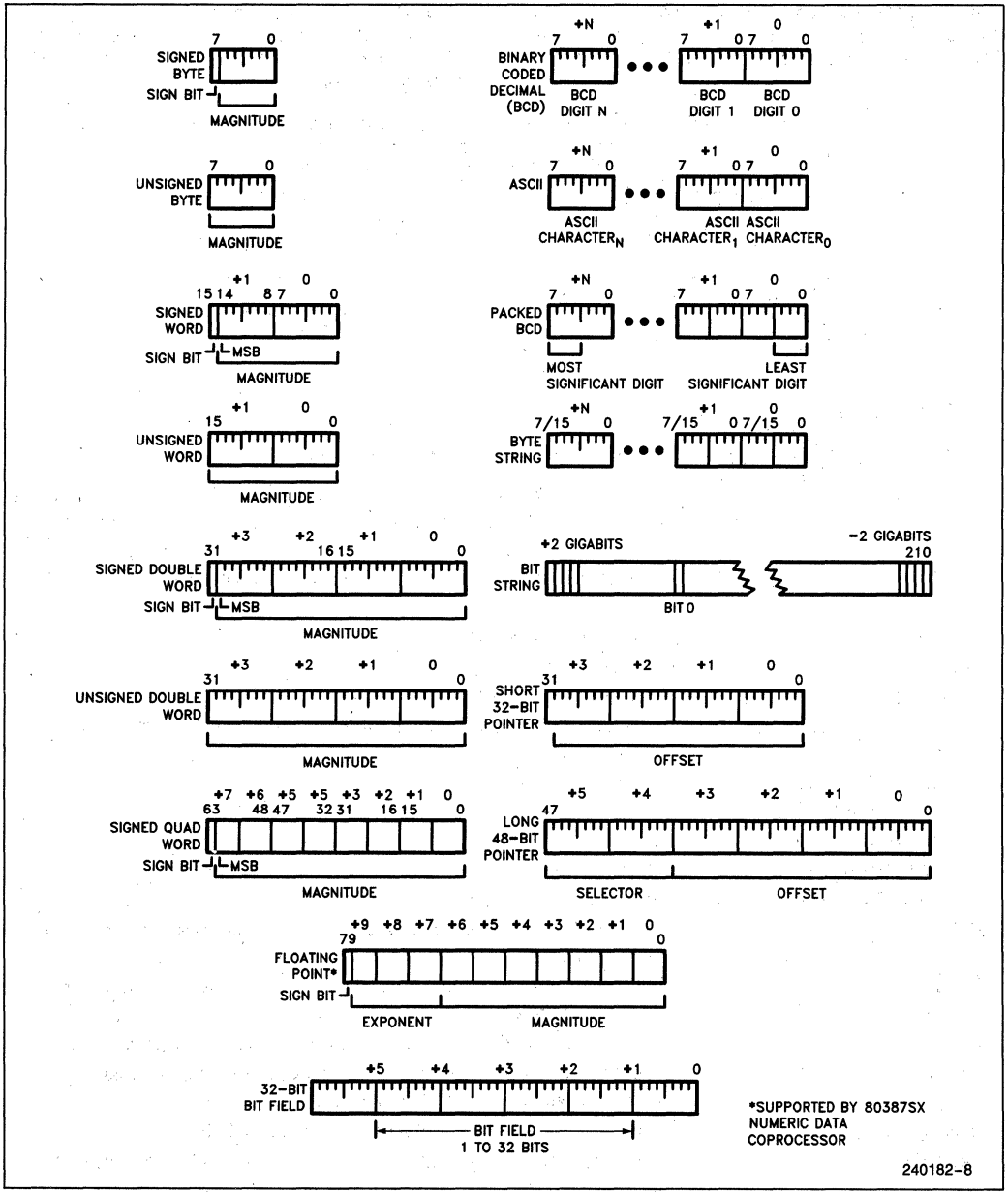


Figure 2.5. 80376 Supported Data Types

2.6 I/O Space

The 80376 has two distinct physical address spaces: physical memory and I/O. Generally, peripherals are placed in I/O space although the 80376 also supports memory-mapped peripherals. The I/O space consists of 64 Kbytes which can be divided into 64K 8-bit ports, 32K 16-bit ports, or any combination of ports which add to no more than 64 Kbytes. The M/IO# pin acts as an additional address line, thus allowing the system designer to easily determine which address space the processor is accessing. Note that the I/O address refers to a physical address.

The I/O ports are accessed by the IN and OUT instructions, with the port address supplied as an immediate 8-bit constant in the instruction or in the DX register. All 8-bit and 16-bit port addresses are zero extended on the upper address lines. The I/O instructions cause the M/IO# pin to be driven LOW. I/O port addresses 00F8H through 00FFH are reserved for use by Intel.

2.7 Interrupts and Exceptions

Interrupts and exceptions alter the normal program flow in order to handle external events, report errors or exceptional conditons. The difference between interrupts and exceptions is that interrupts are used to handle asynchronous external events while exceptions handle instruction faults. Although a program can generate a software interrupt via an INT N instruction, the processor treats software interrupts as exceptions.

Hardware interrupts occur as the result of an external event and are classified into two types: maskable or non-maskable. Interrupts are serviced after the execution of the current instruction. After the interrupt handler is finished servicing the interrupt, execution proceeds with the instruction immediately **after** the interrupted instruction.

Exceptions are classified as faults, traps, or aborts depending on the way they are reported, and whether or not restart of the instruction causing the exception is suported. **Faults** are exceptions that are detected and serviced **before** the execution of the faulting instruction. **Traps** are exceptions that are reported immediately **after** the execution of the instruction which caused the problem. **Aborts** are exceptions which do not permit the precise location of the instruction causing the exception to be determined. Thus, when an interrupt service routine has been completed, execution proceeds from the instruction immediately following the interrupted instruction. On the other hand the return address from an exception/fault routine will always point at the instruction causing the exception and include any leading instruction prefixes. Table 2.5 summarizes the possible interrupts for the 80376 and shows where the return address points to.

The 80376 has the ability to handle up to 256 different interrupts/exceptions. In order to service the interrupts, a table with up to 256 interrupt vectors must be defined. The interrupt vectors are simply pointers to the appropriate interrupt service routine. The interrupt vectors are 8-byte quantities, which are put in an Interrupt Descriptor Table. Of the 256 possible interrupts, 32 are reserved for use by Intel and the remaining 224 are free to be used by the system designer.

INTERRUPT PROCESSING

When an interrupt occurs the following actions happen. First, the current program address and the Flags are saved on the stack to allow resumption of the interrupted program. Next, an 8-bit vector is supplied to the 80376 which identifies the appropriate entry in the interrupt table. The table contains either an Interrupt Gate, a Trap Gate or a Task Gate that will point to an interrupt procedure or task. The user supplied interrupt service routine is executed. Finally, when an IRET instruction is executed the old processor state is restored and program execution resumes at the appropriate instruction.

The 8-bit interrupt vector is supplied to the 80376 in several different ways: exceptions supply the interrupt vector internally; software INT instructions contain or imply the vector; maskable hardware interrupts supply the 8-bit vector via the interrupt acknowledge bus sequence. Non-Maskable hardware interrupts are assigned to interrupt vector 2.

Maskable Interrupt

Maskable interrupts are the most common way to respond to asynchronous external hardware events. A hardware interrupt occurs when the INTR is pulled HIGH and the Interrupt Flag bit (IF) is enabled. The processor only responds to interrupts between instructions (string instructions have an "interrupt window" between memory moves which allows interrupts during long string moves). When an interrupt occurs the processor reads an 8-bit vector supplied by the hardware which identifies the source of the interrupt (one of 224 user defined interrupts).

Table 2.5. Interrupt Vector Assignments

Function	Interrupt Number	Instruction Which Can Cause Exception	Return Address Points to Faulting Instruction	Type
Divide Error	0	DIV, IDIV	Yes	FAULT
Debug Exception	1	Any Instruction	Yes	TRAP*
NMI Interrupt	2	INT 2 or NMI	No	NMI
One-Byte Interrupt	3	INT	No	TRAP
Interrupt on Overflow	4	INTO	No	TRAP
Array Bounds Check	5	BOUND	Yes	FAULT
Invalid OP-Code	6	Any Illegal Instruction	Yes	FAULT
Device Not Available	7	ESC, WAIT	Yes	FAULT
Double Fault	8	Any Instruction That Can Generate an Exception		ABORT
Coprocessor Segment Overrun	9	ESC	No	ABORT
Invalid TSS	10	JMP, CALL, IRET, INT	Yes	FAULT
Segment Not Present	11	Segment Register Instructions	Yes	FAULT
Stack Fault	12	Stack References	Yes	FAULT
General Protection Fault	13	Any Memory Reference	Yes	FAULT
Intel Reserved	14–15	—	—	—
Coprocessor Error	16	ESC, WAIT	Yes	FAULT
Intel Reserved	17–32			
Two-Byte Interrupt	0–255	INT n	No	TRAP

*Some debug exceptions may report both traps on the previous instruction, and faults on the next instruction.

Interrupts through Interrupt Gates automatically reset IF, disabling INTR requests. Interrupts through Trap Gates leave the state of the IF bit unchanged. Interrupts through a Task Gate change the IF bit according to the image of the EFLAGs register in the task's Task State Segment (TSS). When an IRET instruction is executed, the original state of the IF bit is restored.

Non-Maskable Interrupt

Non-maskable interrupts provide a method of servicing very high priority interrupts. When the NMI input is pulled HIGH it causes an interrupt with an internally supplied vector value of 2. Unlike a normal hardware interrupt no interrupt acknowledgement sequence is performed for an NMI.

While executing the NMI servicing procedure, the 80376 will not service any further NMI request, or INT requests, until an interrupt return (IRET) instruction is executed or the processor is reset. If NMI occurs while currently servicing an NMI, its presence will be saved for servicing after executing the first IRET instruction. The disabling of INTR requests depends on the gate in IDT location 2.

Software Interrupts

A third type of interrupt/exception for the 80376 is the software interrupt. An INT n instruction causes the processor to execute the interrupt service routine pointed to by the n^{th} vector in the interrupt table.

A special case of the two byte software interrupt INT n is the one byte INT 3, or breakpoint interrupt. By inserting this one byte instruction in a program, the user can set breakpoints in his program as a debugging tool.

A final type of software interrupt, is the single step interrupt. It is discussed in **Single-Step Trap** (page 22).

INTERRUPT AND EXCEPTION PRIORITIES

Interrupts are externally-generated events. Maskable Interrupts (on the INTR input) and Non-Maskable Interrupts (on the NMI input) are recognized at instruction boundaries. When NMI and maskable INTR are **both** recognized at the **same** instruction boundary, the 80376 invokes the NMI service routine first. If, after the NMI service routine has been invoked, maskable interrupts are still enabled, then the 80376 will invoke the appropriate interrupt service routine.

As the 80376 executes instructions, it follows a consistent cycle in checking for exceptions, as shown in Table 2.6. This cycle is repeated as each instruction is executed, and occurs in parallel with instruction decoding and execution.

INSTRUCTION RESTART

The 80376 fully supports restarting all instructions after faults. If an exception is detected in the instruction to be executed (exception categories 4 through 9 in Table 2.6), the 80376 device invokes the appropriate exception service routine. The 80376 is in a state that permits restart of the instruction.

DOUBLE FAULT

A Double fault (exception 8) results when the processor attempts to invoke an exception service routine for the segment exceptions (10, 11, 12 or 13), but in the process of doing so, detects an exception.

2.8 Reset and Initialization

When the processor is Reset the registers have the values shown in Table 2.7. The 80376 will then start executing instructions near the top of physical memory, at location 0FFFFF0H. A short JMP should be executed within the segment defined for power-up (see Table 2.7). The GDT should then be initialized for a start-up data and code segment followed by a far JMP that will load the segment descriptor cache with the new descriptor values. The IDT table, after reset, is located at physical address 0H, with a limit of 256 entries.

RESET forces the 80376 to terminate all execution and local bus activity. No instruction execution or bus activity will occur as long as Reset is active. Between 350 and 450 CLK2 periods after Reset becomes inactive, the 80376 will start executing instructions at the top of physical memory.

Table 2.6. Sequence of Exception Checking

Consider the case of the 80376 having just completed an instruction. It then performs the following checks before reaching the point where the next instruction is completed:

1. Check for Exception 1 Traps from the instruction just completed (single-step via Trap Flag, or Data Breakpoints set in the Debug Registers).
2. Check for external NMI and INTR.
3. Check for Exception 1 Faults in the next instruction (Instruction Execution Breakpoint set in the Debug Registers for the next instruction).
4. Check for Segmentation Faults that prevented fetching the entire next instruction (exceptions 11 or 13).
5. Check for Faults decoding the next instruction (exception 6 if illegal opcode; or exception 13 if instruction is longer than 15 bytes, or privilege violation (i.e. not at IOPL or at CPL = 0).
6. If WAIT opcode, check if TS = 1 and MP = 1 (exception 7 if both are 1).
7. If ESCape opcode for numeric coprocessor, check if EM = 1 or TS = 1 (exception 7 if either are 1).
8. If WAIT opcode or ESCape opcode for numeric coprocessor, check ERROR# input signal (exception 16 if ERROR# input is asserted).
9. Check for Segmentation Faults that prevent transferring the entire memory quantity (exceptions 11, 12, 13).

Table 2.7. Register Values after Reset

Flag Word (EFLAGS)	uuuu0002H	(Note 1)
Machine Status Word (CR0)	uuuuuuu1H	(Note 2)
Instruction Pointer (EIP)	0000FFF0H	
Code Segment (CS)	F000H	(Note 3)
Data Segment (DS)	0000H	(Note 4)
Stack Segment (SS)	0000H	
Extra Segment (ES)	0000H	(Note 4)
Extra Segment (FS)	0000H	
Extra Segment (GS)	0000H	
EAX Register	0000H	(Note 5)
EDX Register	Component and Stepping ID	(Note 6)
All Other Registers	Undefined	(Note 7)

NOTES:
1. EFLAG Register. The upper 14 bits of the EFLAGS register are undefined, all defined flag bits are zero.
2. CR0: The defined 4 bits in the CR0 is equal to 1H.
3. The Code Segment Register (CS) will have its Base Address set to 0FFFF0000H and Limit set to 0FFFFH.
4. The Data and Extra Segment Registers (DS and ES) will have their Base Address set to 000000000H and Limit set to 0FFFFH.
5. If self-test is selected, the EAX should contain a 0 value. If a value of 0 is not found the self-test has detected a flaw in the part.
6. EDX register always holds component and stepping identifier.
7. All unidentified bits are Intel Reserved and should not be used.

2.9 Initialization

Because the 80376 processor starts executing in protected mode, certain precautions need be taken during initialization. Before any far jumps can take place the GDT and/or LDT tables need to be setup and their respective registers loaded. Before interrupts can be initialized the IDT table must be setup and the IDTR must be loaded. The example code is shown below:

```
; ***************************************************************
;
; This is an example of startup code to put either an 80376,
; 80386SX or 80386 into flat mode. All of memory is treated as
; simple linear RAM. There are no interrupt routines. The
; Builder creates the GDT-alias and IDT-alias and places them,
; by default, in GDT[1] and GDT[2].  Other entries in the GDT
; are specified in the Build file.  After initialization it jumps
; to a C startup routine. To use this template, change this jmp
; address to that of your code, or make the label of your code
; "c_startup".
;
; This code was assembled and built using version 1.2 of the
; Intel RLL utilities and Intel 386ASM assembler.
;
;            ***     This code was tested     ***
;
; ***************************************************************
```

```
NAME FLAT                ; name of the object module

EXTRN    c_startup:near ; this is the label jmped to after init

pe_flag       equ 1
data_selc     equ 20h    ; assume code is GDT[3], data GDT[4]

INIT_CODE    SEGMENT ER PUBLIC USE32     ; Segment base at 0ffffff80h

PUBLIC GDT_DESC

gdt_desc    dq  ?

PUBLIC    START

start:
    cld                          ; clear direction flag
    smsw bx                      ; check for processor (80376) at reset
    test bl,1                    ; use SMSW rather than MOV for speed
    jnz pestart
realstart                        ; is an 80386 and in real mode
    db 66h                       ; force the next operand into 32-bit mode.
    mov eax,offset gdt_desc      ; move address of the GDT descriptor into eax
    xor ebx,ebx                  ; clear ebx
    mov bh,ah                    ; load 8 bits of address into bh
    move bl,al                   ; load 8 bits of address into bl
    db 67h
    db 66h                       ; use the 32-bit form of LGDT to load
    lgdt cs:[ebx]                ; the 32-bits of address into the GDTR
    smsw ax                      ; go into protected mode (set PE bit)
    or al,pe_flag
    lmsw ax
    jmp next                     ; flush prefetch queue
pestart:
    mov ebx,offset gdt_desc
    xor eax,eax
    mov ax,bx                    ; lower portion of address only
    lgdt cs:[eax]
    xor ebx,ebx                  ; initialize data selectors
    mov bl,data_selc             ; GDT[3]
    mov ds,bx
    mov ss,bx
    mov es,bx
    mov fs,bx
    mov gs,bx
    jmp pejump
next:
    xor ebx,ebx                  ; initialize data selectors
    mov bl,data_selc             ; GDT[3]
    mov ds,bx
    mov ss,bx
    mov es,bx
    mov fs,bx
    mov gs,bx
    db 66h                       ; for the 80386, need to make a 32-bit jump
pejump:
    jmp far ptr c_startup        ; but the 80376 is already 32-bit.

    org 70h                      ; only if segment base is at 0ffffff80h
    jmp short start
INIT_CODE ENDS
END
```

This code should be linked into your application for boot loadable code. The following build file illustrates how this is accomplished.

```
FLAT; -- build program id

SEGMENT
    *segments (dpl=0),              -- Give all user segments a DPL of 0.
    _phantom_code_ (dpl=0),         -- These two segments are created by
    _phantom_data_ (dpl=0),         -- the builder when the FLAT control is used.
    init_code  (base=0ffffff80h); -- Put startup code at the reset vector area.

GATE
    g13 (entry=13, dpl=0, trap),        -- trap gate disables interrupts
    i32 (entry=32, dpl=0, interrupt),  -- interrupt gates doesn't

TABLE
    -- create GDT

    GDT (LOCATION = GDT_DESC,       -- In a buffer starting at GDT_DESC,
                                    -- BLD386 places the GDT base and
                                    -- GDT limit values. Buffer must be
                                    -- 6 bytes long. The base and limit
                                    -- values are places in this buffer
                                    -- as two bytes of limit plus
                                    -- four bytes of base in the format
                                    -- required for use by the LGDT
                                    -- instruction.
        ENTRY = (3:_phantom_code_,      -- Explicitly place segment
                 4:_phantom_data_,      -- entries into the GDT.
                 5:code32,
                 6:data,
                 7:init_code)
        );
TASK

    MAIN_TASK
        (
         DPL  = 0,                  -- Task privilege level is 0.
         DATA = DATA,               -- Points to a segment that
                                    -- indicates initial DS value.
         CODE = main,               -- Entry point is main, which
                                    -- must be a public id.

         STACKS = (DATA),           -- Segment id points to stack
                                    -- segment. Sets the initial SS:ESP.
         NO INTENABLED,             -- Disable interrupts.
         PRESENT                    -- Present bit in TSS set to 1.
        );

    MEMORY
        (RANGE = (EPROM = ROM(0ffff8000h..0ffffffffh),
                  DRAM = RAM(0..0ffffh)),
        ALLOCATE = (EPROM = (MAIN_TASK)));

END
```

```
asm386 flatsim.a38 debug
asm386 application.a38 debug
bnd386 application.obj,flatsim.obj nolo debug oj (application.bnd)
bld386 application.bnd bf (flatsim.bld) bl flat
```

Commands to assemble and build a boot-loadable application named "application.a38". The initialization code is called "flatsim.a38", and build file is called "application.bld".

2.10 Self-Test

The 80376, like the 80386, has the capability to perform a self-test. The self-test checks the function of all of the Control ROM and most of the non-random logic of the part. Approximately one-half of the 80376 can be tested during self-test.

Self-Test is initiated on the 80376 when the RESET pin transitions from HIGH to LOW, and the BUSY# pin is LOW. The self-test takes about 2^{20} clocks, or approximately 33 ms with a 16 MHz 80376 processor. At the completion of self-test the processor performs reset and begins normal operation. The part has successfully passed self-test if the contents of the EAX register is zero. If the EAX register is not zero then the self-test has detected a flaw in the part. If self-test is not selected after reset, EAX may be non-zero after reset.

2.11 Debugging Support

The 80376 provides several features which simplify the debugging process. The three categories of on-chip debugging aids are:

1. The code execution breakpoint opcode (0CCH).
2. The single-step capability provided by the TF bit in the flag register, and
3. The code and data breakpoint capability provided by the Debug Registers DR0–3, DR6, and DR7.

BREAKPOINT INSTRUCTION

A single-byte software interrupt (Int 3) breakpoint instruction is available for use by software debuggers. The breakpoint opcode is 0CCh, and generates an exception 3 trap when executed.

DEBUG REGISTERS

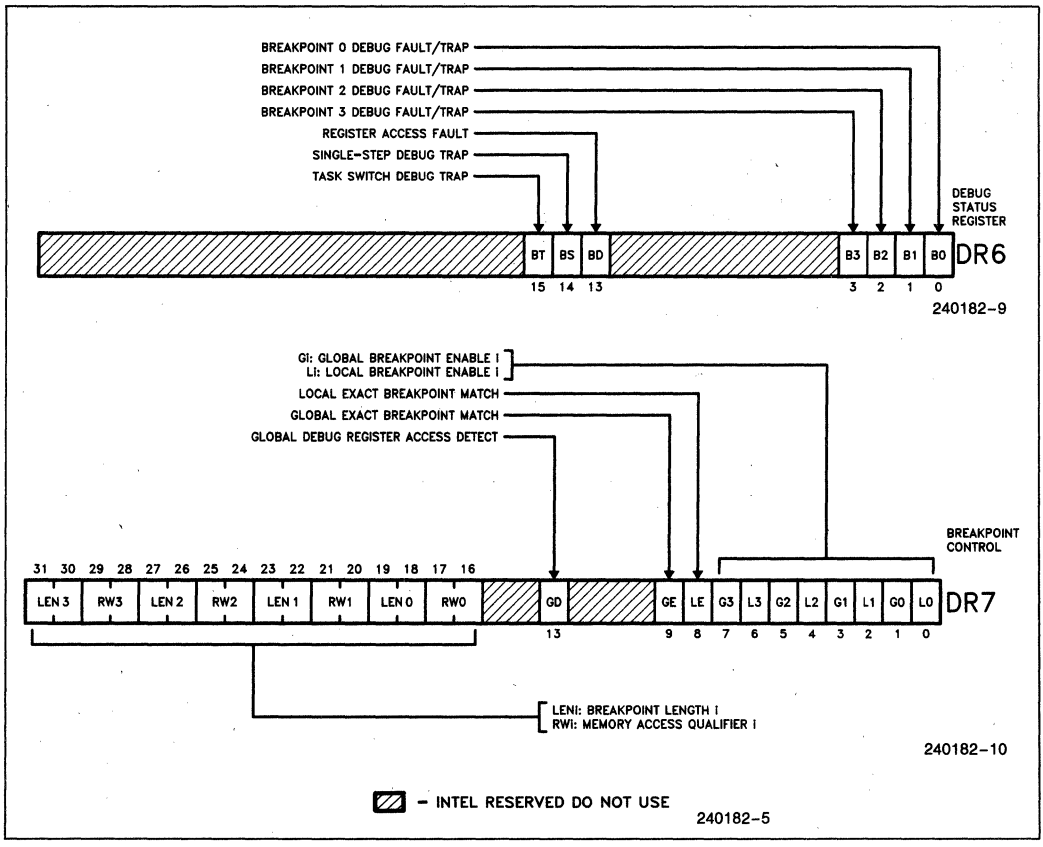


Figure 2.6. Debug Registers

SINGLE-STEP TRAP

If the single-step flag (TF, bit 8) in the EFLAG register is found to be set at the end of an instruction, a single-step exception occurs. The single-step exception is auto vectored to exception number 1.

The Debug Registers are an advanced debugging feature of the 80376. They allow data access breakpoints as well as code execution breakpoints. Since the breakpoints are indicated by on-chip registers, an instruction execution breakpoint can be placed in ROM code or in code shared by several tasks, neither of which can be supported by the INT 3 breakpoint opcode.

The 80376 contains six Debug Registers, consisting of four breakpoint address registers and two breakpoint control registers. Initially after reset, breakpoints are in the disabled state; therefore, no breakpoints will occur unless the debug registers are programmed. Breakpoints set up in the Debug Registers are auto-vectored to exception 1. Figure 2.6 shows the breakpoint status and control registers.

3.0 ARCHITECTURE

The Intel 80376 Embedded Processor has a physical address space of 16 Mbytes (2^{24} bytes) and allows the running of virtual memory programs of almost unlimited size (16 Kbytes × 16 Mbytes or 256 Gbytes (2^{38} bytes)). In addition the 80376 provides a sophisticated memory management and a hardware-assisted protection mechanism.

3.1 Addressing Mechanism

The 80376 uses two components to form the logical address, a 16-bit selector which determines the linear base address of a segment, and a 32-bit effective address. The selector is used to specify an index into an operating system defined table (see Figure 3.1). The table contains the 32-bit base address of a given segment. The linear address is formed by adding the base address obtained from the table to the 32-bit effective address. This value is truncated to 24 bits to form the physical address, which is then placed on the address bus.

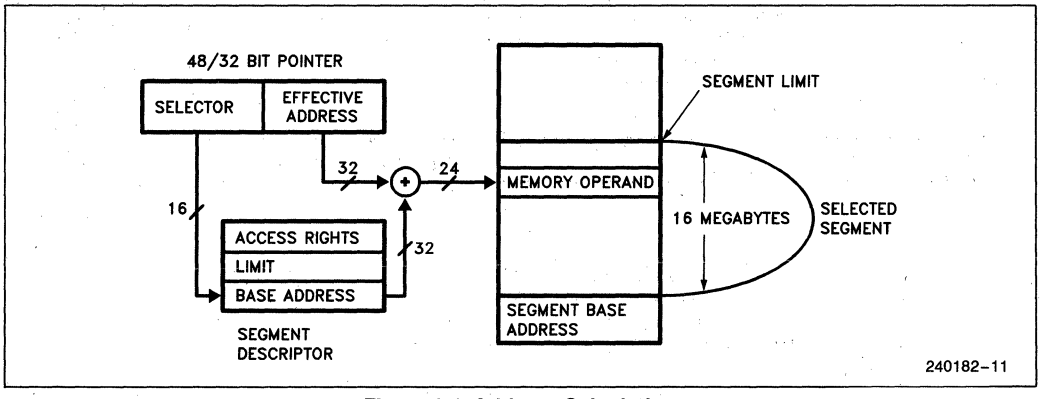


Figure 3.1. Address Calculation

3.2 Segmentation

Segmentation is one method of memory management and provides the basis for protection in the 80376. Segments are used to encapsulate regions of memory which have common attributes. For example, all of the code of a given program could be contained in a segment, or an operating system table may reside in a segment. All information about each segment, is stored in an 8-byte data structure called a descriptor. All of the descriptors in a system are contained in tables recognized by hardware.

TERMINOLOGY

The following terms are used throughout the discussion of descriptors, privilege levels and protection:

PL: **Privilege Level**—One of the four hierarchical privilege levels. Level 0 is the most privileged level and level 3 is the least privileged.

RPL: **Requestor Privilege Level**—The privilege level of the original supplier of the selector. RPL is determined by the least two significant bits of a selector.

DPL: **Descriptor Privilege Level**—This is the least privileged level at which a task may access that descriptor (and the segment associated with that descriptor). Descriptor Privilege Level is determined by bits 6:5 in the Access Right Byte of a descriptor.

CPL: **Current Privilege Level**—The privilege level at which a task is currently executing, which equals the privilege level of the code segment being executed. CPL can also be determined by examining the lowest 2 bits of the CS register, except for conforming code segments.

EPL: **Effective Privilege Level**—The effective privilege level is the least privileged of the RPL and the DPL. EPL is the numerical maximum of RPL and DPL.

Task: One instance of the execution of a program. Tasks are also referred to as processes.

DESCRIPTOR TABLES

The descriptor tables define all of the segments which are used in an 80376 system. There are three types of tables on the 80376 which hold descriptors: the Global Descriptor Table, Local Descriptor Table, and the Interrupt Decriptor Table. All of the tables are variable length memory arrays, they can range in size between 8 bytes and 64 Kbytes. Each table can hold up to 8192 8-byte descriptors. The upper 13 bits of a selector are used as an index into the descriptor table. The tables have registers associated with them which hold the 32-bit linear base address, and the 16-bit limit of each table.

Each of the tables have a register associated with it: GDTR, LDTR and IDTR; see Figure 3.2. The LGDT, LLDT and LIDT instructions load the base and limit of the Global, Local and Interrupt Descriptor Tables into the appropriate register. The SGDT, SLDT and SIDT store these base and limit values. These are privileged instructions.

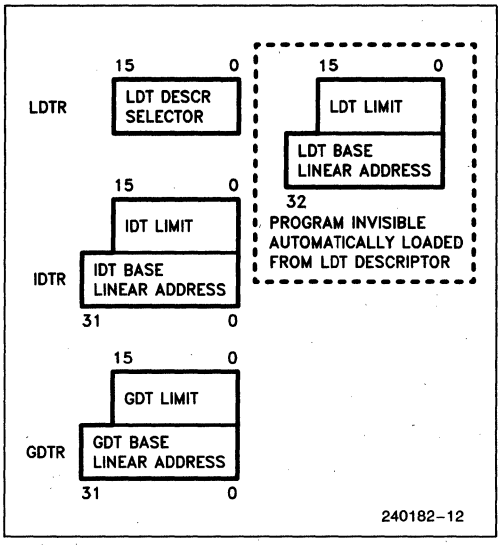


Figure 3.2. Descriptor Table Registers

Global Descriptor Table

The Global Descriptor Table (GDT) contains descriptors which are possibly available to all of the tasks in a system. The GDT can contain any type of segment descriptor except for interrupt and trap descriptors. Every 80376 system contains a GDT. A simple 80376 system contains only 2 entries in the GDT; a code and a data descriptor.

The first slot of the Global Descriptor Table corresponds to the null selector and is not used. The null selector defines a null pointer value.

Local Descriptor Table

LDTs contain descriptors which are associated with a given task. Generally, operating systems are designed so that each task has a separate LDT. The LDT may contain only code, data, stack, task gate, and call gate descriptors. LDTs provide a mechanism for isolating a given task's code and data segments from the rest of the operating system, while the GDT contains descriptors for segments which are common to all tasks. A segment cannot be accessed by a task if its segment descriptor does not exist in either the current LDT or the GDT. This pro-

vides both isolation and protection for a task's segments, while still allowing global data to be shared among tasks.

Unlike the 6-byte GDT or IDT registers which contain a base address and limit, the visible portion of the LDT register contains only a 16-bit selector. This selector refers to a Local Descriptor Table descriptor in the GDT (see Figure 2.1).

INTERRUPT DESCRIPTOR TABLE

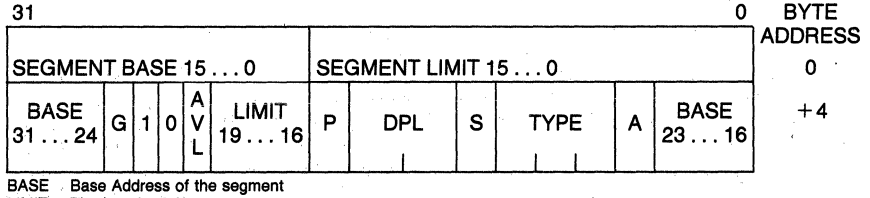
The third table needed for 80376 systems is the Interrupt Descriptor Table. The IDT contains the descriptors which point to the location of up to 256 interrupt service routines. The IDT may contain only task gates, interrupt gates and trap gates. The IDT should be at least 256 bytes in size in order to hold the descriptors for the 32 Intel Reserved Interrupts. Every interrupt used by a system must have an entry in the IDT. The IDT entries are referenced by INT instructions, external interrupt vectors, and exceptions.

DESCRIPTORS

The object to which the segment selector points to is called a descriptor. Descriptors are eight-byte quantities which contain attributes about a given region of linear address space. These attributes include the 32-bit logical base address of the segment, the 20-bit length and granularity of the segment, the protection level, read, write or execute privileges, and the type of segment. All of the attribute information about a segment is contained in 12 bits in the segment descriptor. Figure 3.3 shows the general format of a descriptor. All segments on the the 80376 have three attribute fields in common: the Present bit (P), the Descriptor Privilege Level bits (DPL) and the Segment bit (S). P=1 if the segment is loaded in physical memory, if P = 0 then any attempt to access the segment causes a not present exception (exception 11). The DPL is a two-bit field which specifies the protection level, 0–3, associated with a segment.

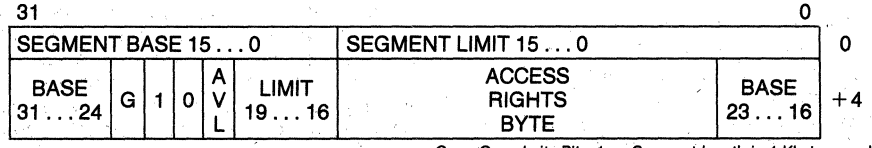
The 80376 has two main categories of segments: system segments, and non-system segments (for code and data). The segment bit, S, determines if a given segment is a system segment, a code segment or a data segment. If the S bit is 1 then the segment is either a code or data segment, if it is 0 then the segment is a system segment.

Note that although the 80376 is limited to a 16-Mbyte Physical address space (2^{24}), its base address allows a segment to be placed anywhere in a 4-Gbyte linear address space. When writing code for the 80376, users should keep code protability to an 80386 processor (or other processors with a larger physical address space) in mind. A segment base address can be placed anywhere in this 4-Gbyte linear address space, but a physical address will be



BASE Base Address of the segment
LIMIT The length of the segment
P Present Bit 1 = Present 0 = Not Present
DPL Descriptor Privilege Level 0–3
S Segment Descriptor: 0 = System Descriptor, 1 = Code or Data Descriptor
TYPE Type of Segment
A Accessed Bit
G Granularity Bit 1 = Segment length is 4 Kbyte Granular
0 = Segment length is byte granular
0 Bit must be zero (0) for compatibility with future processors
AVL Available field for user or OS

Figure 3.3. Segment Descriptors



G Granularity Bit 1 = Segment length is 4 Kbyte granular
0 = Segment length is byte granular
0 Bit must be zero (0) for compatibility with future processors
AVL Available field for user or OS

Figure 3.4. Code and Data Descriptors

Table 3.1. Access Rights Byte Definition for Code and Data Descriptors

Bit Position	Name	Function	
7	Present (P)	P = 1 Segment is mapped into physical memory. P = 0 No mapping to physical memory exits	
6–5	Descriptor Privilege Level (DPL)	Segment privilege attribute used in privilege tests.	
4	Segment Descriptor (S)	S = 1 Code or Data (includes stacks) segment descriptor S = 0 System Segment Descriptor or Gate Descriptor	
3	Executable (E)	E = 0 Descriptor type is data segment:	If Data Segment (S = 1, E = 0)
2	Expansion Direction (ED)	ED = 0 Expand up segment, offsets must be ≤ limit. ED = 1 Expand down segment, offsets must be > limit.	
1	Writable (W)	W = 0 Data segment may not be written into. W = 1 Data segment may be written into.	
3	Executable (E)	E = 1 Descriptor type is code segment:	If Code Segment (S = 1, E = 1)
2	Conforming (C)	C = 1 Code segment may only be executed when CPL ≥ DPL and CPL remains unchanged.	
1	Readable (R)	R = 0 Code segment may not be read. R = 1 Code segment may be read.	
0	Accessed (A)	A = 0 Segment has not been accessed. A = 1 Segment selector has been loaded into segment register or used by selector test instructions.	

generated that is a truncated version of this linear address. Truncation will be to the maximum number of address bits. It is recommended to place EPROM at the highest physical address and DRAM at the lowest physical addresses.

Code and Data Descriptors (S = 1)

Figure 3.4 shows the general format of a code and data descriptor and Table 3.1 illustrates how the bits in the Access Right Byte are interpreted.

Code and data segments have several descriptor fields in common. The accessed bit, A, is set whenever the processor accesses a descriptor. The granularity bit, G, specifies if a segment length is 1-byte-granular or 4-Kbyte-granular. Base address bits 31–24, which are normally found in 80386 descriptors, are not made externally available on the 80376. They do not affect the operation of the 80376. The A_{31}–A_{24} field should be set to allow an 80386 to correctly execute with EPROM at the upper 4096 Mbytes of physical memory.

System Descriptor Formats (S = 0)

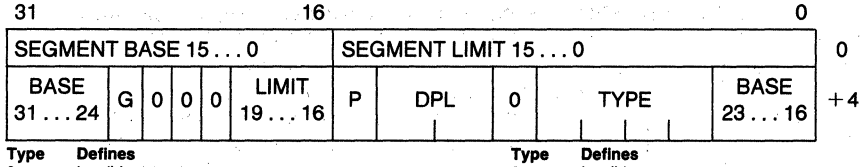
System segments describe information about operating system tables, tasks, and gates. Figure 3.5 shows the general format of system segment descriptors, and the various types of system segments. 80376 system descriptors (which are the same as 80386 descriptor types 2, 5, 9, B, C, E and F) contain a 32-bit logical base address and a 20-bit segment limit.

Selector Fields

A selector has three fields: Local or Global Descriptor Table Indicator (TI), Descriptor Entry Index (Index), and Requestor (the selector's) Privilege Level (RPL) as shown in Figure 3.6. The TI bit selects either the Global Descriptor Table or the Local Descriptor Table. The Index selects one of 8K descriptors in the appropriate descriptor table. The RPL bits allow high speed testing of the selector's privilege attributes.

Segment Descriptor Cache

In addition to the selector value, every segment register has a segment descriptor cache register associated with it. Whenever a segment register's contents are changed, the 8-byte descriptor associated with that selector is automatically loaded (cached) on the chip. Once loaded, all references to that segment use the cached descriptor information instead of reaccessing the descriptor. The contents of the descriptor cache are not visible to the programmer. Since descriptor caches only change when a segment register is changed, programs which modify the descriptor tables must reload the appropriate segment registers after changing a descriptor's value.



Type	Defines
0	Invalid
1	Reserved
2	LDT
3	Reserved
4	Reserved
5	Task Gate (80376/80386 Task)
6	Reserved
7	Reserved

Type	Defines
8	Invalid
9	Available 80376/80386 TSS
A	Undefined (Intel Reserved)
B	Busy 80376/80386 TSS
C	80376/80386 Call Gate
D	Undefined (Intel Reserved)
E	80376/80386 Interrupt Gate
F	80376/80386 Trap Gate

Figure 3.5. System Descriptors

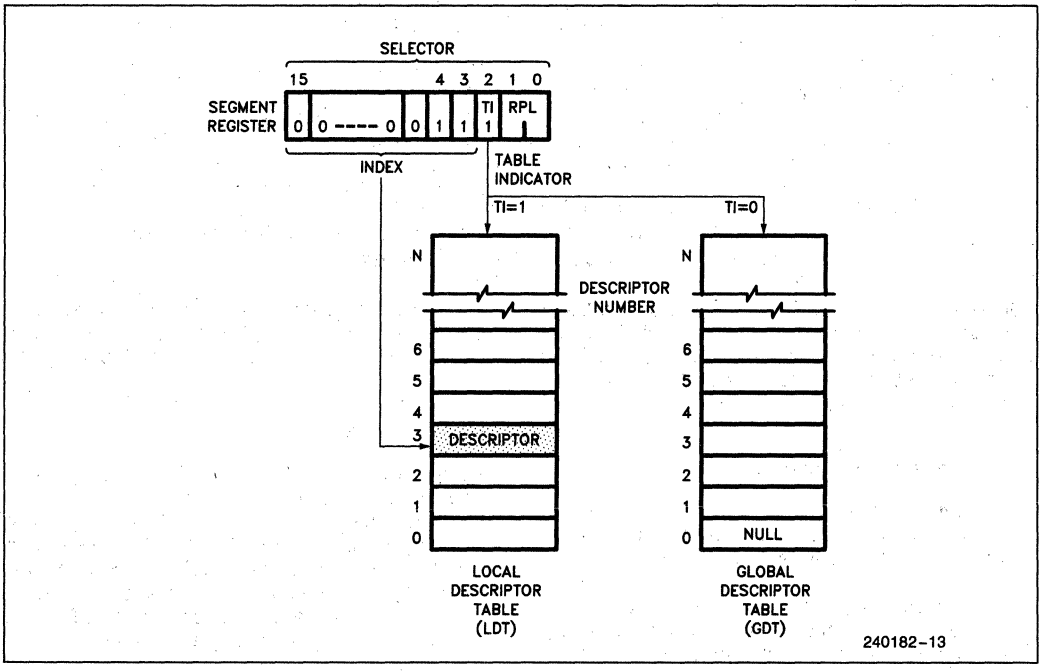


Figure 3.6. Example Descriptor Selection

3.3 Protection

The 80376 offers extensive protection features. These protection features are particularly useful in sophisticated embedded applications which use multitasking real-time operating systems. For simpler embedded applications these protection capabilities can be easily bypassed by making all applications run at privilege level (PL) 0.

RULES OF PRIVILEGE

The 80376 controls access to both data and procedures between levels of a task, according to the following rules.

—Data stored in a segment with privilege level **p** can be accessed only by code executing at a privilege level at least as privileged as **p.**

—A code segment/procedure with privilege level **p** can only be called by a task executing at the same or a lesser privilege level than **p.**

PRIVILEGE LEVELS

At any point in time, a task on the 80376 always executes at one of the four privilege levels. The Current Privilege Level (CPL) specifies what the task's privilege level is. A task's CPL may only be changed

by control transfers through gate descriptors to a code segment with a different privilege level. Thus, an application program running at PL = 3 may call an operating system routine at PL = 1 (via a gate) which would cause the task's CPL to be set to 1 until the operating system routine was finished.

Selector Privilege (RPL)

The privilege level of a selector is specified by the RPL field. The selector's RPL is only used to establish a less trusted privilege level than the current privilege level of the task for the use of a segment. This level is called the task's effective privilege level (EPL). The EPL is defined as being the least privileged (numerically larger) level of a task's CPL and a selector's RPL. The RPL is most commonly used to verify that pointers passed to an operating system procedure do not access data that is of higher privilege than the procedure that originated the pointer. Since the originator of a selector can specify any RPL value, the Adjust RPL (ARPL) instruction is provided to force the RPL bits to the originator's CPL.

I/O Privilege

The I/O privilege level (IOPL) lets the operating system code executing at CPL = 0 define the least privileged level at which I/O instructions can be used. An exception 13 (General Protection Violation) is generated if an I/O instruction is attempted when the CPL of the task is less privileged than the IOPL. The IOPL is stored in bits 13 and 14 of the EFLAGS register. The following instructions cause an exception 13 if the CPL is greater than IOPL: IN, INS, OUT, OUTS, STI, CLI and LOCK prefix.

Descriptor Access

There are basically two types of segment accesses: those involving code segments such as control transfers, and those involving data accesses. Determining the ability of a task to access a segment involves the type of segment to be accessed, the instruction used, the type of descriptor used and CPL, RPL, and DPL as described above.

Any time an instruction loads a data segment register (DS, ES, FS, GS) the 80376 makes protection validation checks. Selectors loaded in the DS, ES, FS, GS registers must refer only to data segment or readable code segments.

Finally the privilege validation checks are performed. The CPL is compared to the EPL and if the EPL is more privileged than the CPL, an exception 13 (general protection fault) is generated.

The rules regarding the stack segment are slightly different than those involving data segments. Instructions that load selectors into SS must refer to data segment descriptors for writeable data segments. The DPL and RPL must equal the CPL of all other descriptor types or a privilege level violation will cause an exception 13. A stack not present fault causes an exception 12.

PRIVILEGE LEVEL TRANSFERS

Inter-segment control transfers occur when a selector is loaded in the CS register. For a typical system most of these transfers are simply the result of a call or a jump to another routine. There are five types of control transfers which are summarized in Table 3.2. Many of these transfers result in a privilege level transfer. Changing privilege levels is done only by control transfers, using gates, task switches, and interrupt or trap gates.

Control transfers can only occur if the operation which loaded the selector references the correct descriptor type. Any violation of these descriptor usage rules will cause an exception 13.

CALL GATES

Gates provide protected indirect CALLs. One of the major uses of gates is to provide a secure method of privilege transfers within a task. Since the operating system defines all of the gates in a system, it can ensure that all gates only allow entry into a few trusted procedures.

Table 3.2. Descriptor Types Used for Control Transfer

Control Transfer Types	Operation Types	Descriptor Referenced	Descriptor Table
Intersegment within the same privilege level	JMP, CALL, RET, IRET*	Code Segment	GDT/LDT
Intersegment to the same or higher privilege level Interrupt within task may change CPL	CALL	Call Gate	GDT/LDT
	Interrupt Instruction, Exception, External Interrupt	Trap or Interrupt Gate	IDT
Intersegment to a lower privilege level (changes task CPL)	RET, IRET*	Code Segment	GDT/LDT
	CALL, JMP	Task State Segment	GDT
Task Switch	CALL, JMP	Task Gate	GDT/LDT
	IRET** Interrupt Instruction, Exception, External Interrupt	Task Gate	IDT

*NT (Nested Task bit of flag register) = 0
**NT (Nested Task bit of flag register) = 1

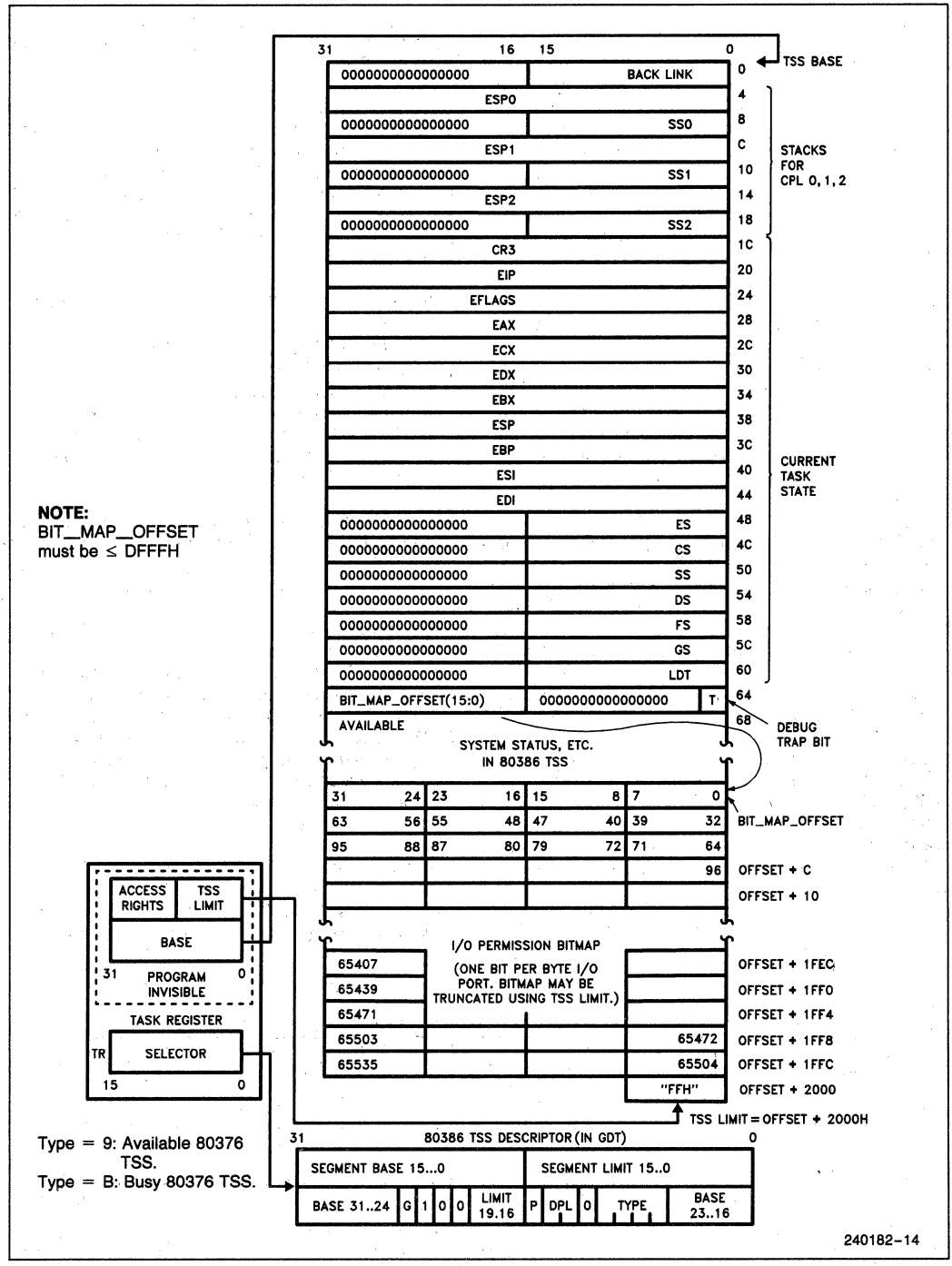


Figure 3.7. 80376 TSS And TSS Registers

TASK SWITCHING

A very important attribute of any multi-tasking operating system is its ability to rapidly switch between tasks or processes. The 80376 directly supports this operation by providing a task switch instruction in hardware. The 80376 task switch operation saves the entire state of the machine (all of the registers, address space, and a link to the previous task), loads a new execution state, performs protection checks, and commences execution in the new task. Like transfer of control by gates, the task switch operation is invoked by executing an inter-segment JMP or CALL instruction which refers to a Task State Segment (TSS), or a task gate descriptor in the GDT or LDT. An INT n instruction, exception, trap or external interrupt may also invoke the task switch operation if there is a task gate descriptor in the associated IDT descriptor slot. For simple applications, the TSS and task switching may not be used. The TSS or task switch will not be used or occur if no task gates are present in the GDT, LDT or IDT.

The TSS descriptor points to a segment (see Figure 3.7) containing the entire 80376 execution state. A task gate descriptor contains a TSS selector. The limit of an 80376 TSS must be greater than 64H, and can be as large as 16 Mbytes. In the additional TSS space, the operating system is free to store additional information as the reason the task is inactive, the time the task has spent running, and open files belonging to the task.

Each Task must have a TSS associated with it. The current TSS is identified by a special register in the 80376 called the Task State Segment Register (TR). This register contains a selector referring to the task state segment descriptor that defines the current TSS. A hidden base and limit register associated with the TSS descriptor is loaded whenever TR is loaded with a new selector. Returning from a task is accomplished by the IRET instruction. When IRET is executed, control is returned to the task which was interrupted. The current executing task's state is saved in the TSS and the old task state is restored from its TSS.

Several bits in the flag register and CR0 register give information about the state of a task which is useful to the operating system. The Nested Task bit, NT, controls the function of the IRET instruction. If NT = 0 the IRET instruction performs the regular return. If NT = 1, IRET performs a task switch operation back to the previous task. The NT bit is set or reset in the following fashion:

When a CALL or INT instruction initiates a task switch, the new TSS will be marked busy and the back link field of the new TSS set to the old TSS selector. The NT bit of the new task is set by CALL or INT initiated task switches. An interrupt that does not cause a task switch will clear NT (The NT bit will be restored after execution of the interrupt handler). NT may also be set or cleared by POPF or IRET instructions.

The 80376 task state segment is marked busy by changing the descriptor type field from TYPE 9 to TYPE 0BH. Use of a selector that references a busy task state segment causes an exception 13.

The coprocessor's state is not automatically saved when a task switch occurs. The Task Switched Bit, TS, in the CR0 register helps deal with the coprocessor's state in a multi-tasking environment. Whenever the 80376 switches tasks, it sets the TS bit. The 80376 detects the first use of a processor extension instruction after a task switch and causes the processor extension not available exception 7. The exception handler for exception 7 may then decide whether to save the state of the coprocessor.

The T bit in the 80376 TSS indicates that the processor should generate a debug exception when switching to a task. If T = 1 then upon entry to a new task a debug exception 1 will be generated.

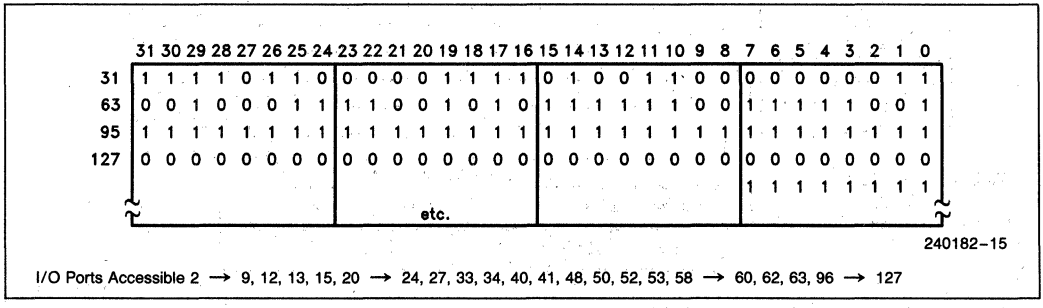


Figure 3.8. Sample I/O Permission Bit Map

PROTECTION AND I/O PERMISSION BIT MAP

The I/O instructions that directly refer to addresses in the processor's I/O space are IN, INS, OUT and OUTS. The 80376 has the ability to selectively trap references to specific I/O addresses. The structure that enables selective trapping is the ***I/O Permission Bit Map*** in the TSS segment (see Figures 3.7 and 3.8). The I/O permission map is a bit vector. The size of the map and its location in the TSS segment are variable. The processor locates the I/O permission map by means of the **I/O map base** field in the fixed portion of the TSS. The **I/O map base** field is 16 bits wide and contains the offset of the beginning of the I/O permission map.

If an I/O instruction (IN, INS, OUT or OUTS) is encountered, the processor first checks whether CPL $\leq$ IOPL. If this condition is true, the I/O operation may proceed. If not true, the processor checks the I/O permission map.

Each bit in the map corresponds to an I/O port byte address; for example, the bit for port 41 is found at **I/O map base** +5 linearly, (5 $\times$ 8 = 40), bit offset 1. The processor tests all the bits that correspond to the I/O addresses spanned by an I/O operation; for example, a double word operation tests four bits corresponding to four adjacent byte addresses. If any tested bit is set, the processor signals a general protection exception. If all the tested bits are zero, the I/O operations may proceed.

It is not necessary for the I/O permission map to represent all the I/O addresses. I/O addresses not spanned by the map are treated as if they had one-bits in the map. The **I/O map base** should be at least one byte less than the TSS limit and the last byte beyond the I/O mapping information must contain all 1's.

Because the I/O permission map is in the TSS segment, different tasks can have different maps. Thus, the operating system can allocate ports to a task by changing the I/O permission map in the task's TSS.

IMPORTANT IMPLEMENTATION NOTE:
Beyond the last byte of I/O mapping information in the I/O permission bit map **must** be a byte containing all 1's. The byte of all 1's must be within the limit of the 80376's TSS segment (see Figure 3.7).

4.0 FUNCTIONAL DATA

The Intel 80376 embedded processor features a straightforward functional interface to the external hardware. The 80376 has separate parallel buses for data and address. The data bus is 16 bits in width, and bidirectional. The address bus outputs 24-bit address values using 23 address lines and two-byte enable signals.

The 80376 has two selectable address bus cycles: pipelined and non-pipelined. The pipelining option allows as much time as possible for data access by

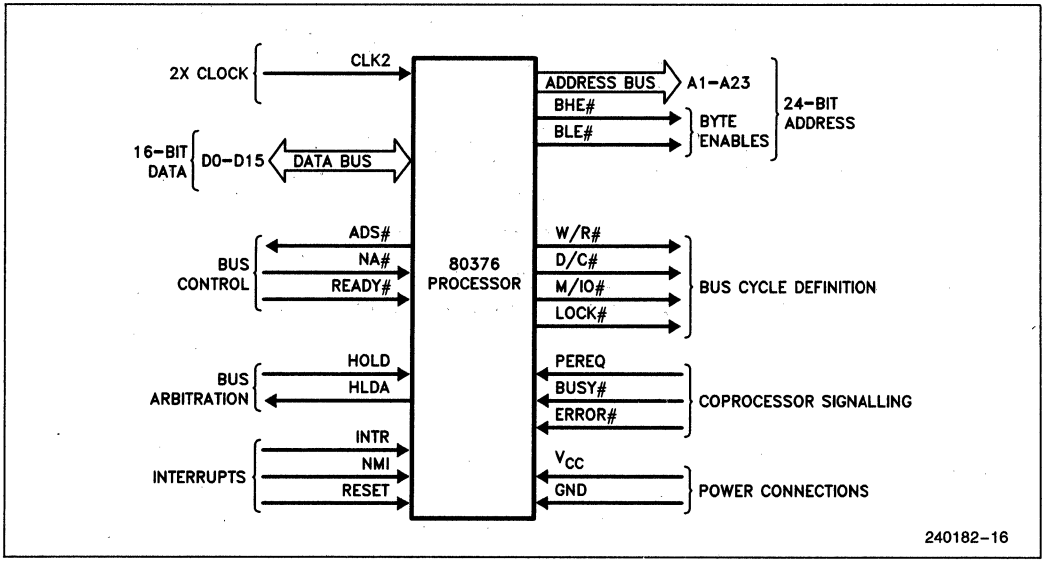


Figure 4.1. Functional Signal Groups

starting the pending bus cycle before the present bus cycle is finished. A non-pipelined bus cycle gives the highest bus performance by executing every bus cycle in two processor clock cycles. For maximum design flexibility, the address pipelining option is selectable on a cycle-by-cycle basis.

The processor's bus cycle is the basic mechanism for information transfer, either from system to processor, or from processor to system. 80376 bus cycles perform data transfer in a minimum of only two clock periods. On a 16-bit data bus, the maximum 80376 transfer bandwidth at 16 MHz is therefore 16 Mbytes/sec. However, any bus cycle will be extended for more than two clock periods if external hardware withholds acknowledgement of the cycle.

The 80376 can relinquish control of its local buses to allow mastership by other devices, such as direct memory access (DMA) channels. When relinquished, HLDA is the only output pin driven by the 80376, providing near-complete isolation of the processor from its system (all other output pins are in a float condition).

4.1 Signal Description Overview

Ahead is a brief description of the 80376 input and output signals arranged by functional groups. Note the # symbol at the end of a signal name indicates the active, or asserted, state occurs when the signal is at a LOW voltage. When no # is present after the signal name, the signal is asserted when at the HIGH voltage level.

Example signal: M/IO#—HIGH voltage indicates Memory selected

—LOW voltage indicates I/O selected

The signal descriptions sometimes refer to A.C. timing parameters, such as "t_{25} Reset Setup Time" and "t_{26} Reset Hold Time." The values of these parameters can be found in Table 6.4.

CLOCK (CLK2)

CLK2 provides the fundamental timing for the 80376. It is divided by two internally to generate the internal processor clock used for instruction execution. The internal clock is comprised of two

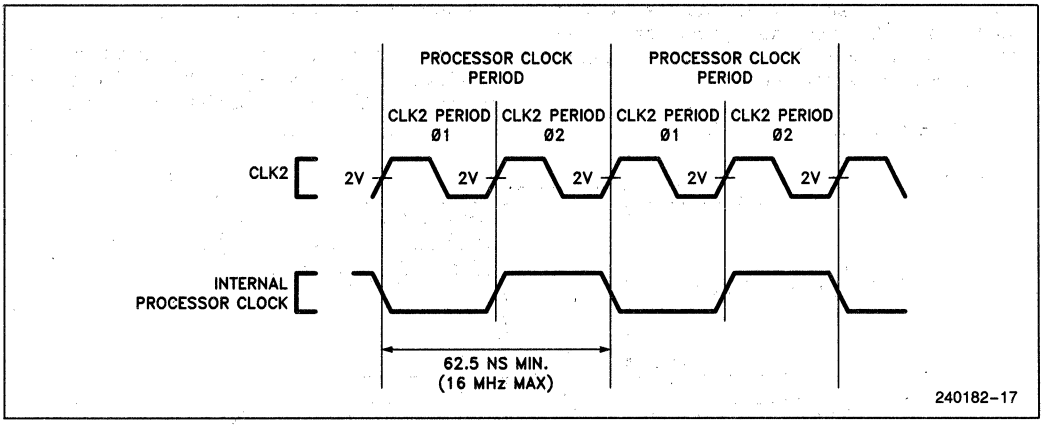


Figure 4.2. CLK2 Signal and Internal Processor Clock

phases, "phase one" and "phase two". Each CLK2 period is a phase of the internal clock. Figure 4.2 illustrates the relationship. If desired, the phase of the internal processor clock can be synchronized to a known phase by ensuring the falling edge of the RESET signal meets the applicable setup and hold times t_{25} and t_{26}.

DATA BUS (D_{15}–D_0)

These three-state bidirectional signals provide the general purpose data path between the 80376 and other devices. The data bus outputs are active HIGH and will float during bus hold acknowledge. Data bus reads require that read-data setup and hold times t_{21} and t_{22} be met relative to CLK2 for correct operation.

ADDRESS BUS (BHE#, BLE#, A_{23}–A_1)

These three-state outputs provide physical memory addresses or I/O port addresses. A_{23}–A_{16} are LOW during I/O transfers except for I/O transfers automatically generated by coprocessor instructions. During coprocessor I/O transfers, A_{22}–A_{16} are driven LOW, and A_{23} is driven HIGH so that this address line can be used by external logic to generate the coprocessor select signal. Thus, the I/O address driven by the 80376 for coprocessor commands is 8000F8H, and the I/O address driven by the 80376 processor for coprocessor data is 8000FCH or 8000FEH.

The address bus is capable of addressing 16 Mbytes of physical memory space (000000H through 0FFFFFFH), and 64 Kbytes of I/O address space (000000H through 00FFFFH) for programmed I/O. The address bus is active HIGH and will float during bus hold acknowledge.

The Byte Enable outputs BHE# and BLE# directly indicate which bytes of the 16-bit data bus are involved with the current transfer. BHE# applies to D_{15}–D_8 and BLE# applies to D_7–D_0. If both BHE# and BLE# are asserted, then 16 bits of data are being transferred. See Table 4.1 for a complete decoding of these signals. The byte enables are active LOW and will float during bus hold acknowledge.

Table 4.1. Byte Enable Definitions

BHE#	BLE#	Function
0	0	Word Transfer
0	1	Byte Transfer on Upper Byte of the Data Bus, D_{15}–D_8
1	0	Byte Transfer on Lower Byte of the Data Bus, D_7–D_0
1	1	Never Occurs

BUS CYCLE DEFINITION SIGNALS (W/R#, D/C#, M/IO#, LOCK#)

These three-state outputs define the type of bus cycle being performed: W/R# distinguishes between write and read cycles, D/C# distinguishes between data and control cycles, M/IO# distinguishes between memory and I/O cycles, and LOCK# distinguishes between locked and unlocked bus cycles. All of these signals are active LOW and will float during bus acknowledge.

The primary bus cycle definition signals are W/R#, D/C# and M/IO#, since these are the signals driven valid as ADS# (Address Status output) becomes active. The LOCK# signal is driven valid at the same time the bus cycle begins, which due to address pipelining, could be after ADS# becomes active. Exact bus cycle definitions, as a function of W/R#, D/C# and M/IO# are given in Table 4.2.

LOCK# indicates that other system bus masters are not to gain control of the system bus while it is active. LOCK# is activated on the CLK2 edge that begins the first locked bus cycle (i.e., it is not active at the same time as the other bus cycle definition pins) and is deactivated when ready is returned to the end of the last bus cycle which is to be locked. The beginning of a bus cycle is determined when READY# is returned in a previous bus cycle and another is pending (ADS# is active) or the clock in which ADS# is driven active if the bus was idle. This means that it follows more closely with the write data rules when it is valid, but may cause the bus to be locked longer than desired. The LOCK# signal may be explicitly activated by the LOCK prefix on certain instructions. LOCK# is always asserted when executing the XCHG instruction, during descriptor updates, and during the interrupt acknowledge sequence.

BUS CONTROL SIGNALS (ADS#, READY#, NA#)

The following signals allow the processor to indicate when a bus cycle has begun, and allow other system hardware to control address pipelining and bus cycle termination.

Address Status (ADS#)

This three-state output indicates that a valid bus cycle definition and address (W/R#, D/C#, M/IO#, BHE#, BLE# and A_{23}–A_1) are being driven at the 80376 pins. ADS# is an active LOW output. Once ADS# is driven active, valid address, byte enables, and definition signals will not change. In addition, ADS# will remain active until its associated bus cycle begins (when READY# is returned for the previous bus cycle when running pipelined bus cycles). ADS# will float during bus hold acknowledge. See sections **Non-Pipelined Bus Cycles** (page 43) and **Pipelined Bus Cycles** (page 45) for additional information on how ADS# is asserted for different bus states.

Transfer Acknowledge (READY#)

This input indicates the current bus cycle is complete, and the active bytes indicated by BHE# and BLE# are accepted or provided. When READY# is sampled active during a read cycle or interrupt acknowledge cycle, the 80376 latches the input data and terminates the cycle. When READY# is sampled active during a write cycle, the processor terminates the bus cycle.

Table 4.2. Bus Cycle Definition

M/IO#	D/C#	W/R#	Bus Cycle Type	Locked?
0	0	0	INTERRUPT ACKNOWLEDGE	Yes
0	0	1	Does Not Occur	—
0	1	0	I/O DATA READ	No
0	1	1	I/O DATA WRITE	No
1	0	0	MEMORY CODE READ	No
1	0	1	HALT: Address = 2, BHE# = 1, BLE# = 0 / SHUTDOWN: Address = 0, BHE# = 1, BLE# = 0	No
1	1	0	MEMORY DATA READ	Some Cycles
1	1	1	MEMORY DATA WRITE	Some Cycles

READY# is ignored on the first bus state of all bus cycles, and sampled each bus state thereafter until asserted. READY# must eventually be asserted to acknowledge every bus cycle, including Halt Indication and Shutdown Indication bus cycles. When being sampled, READY# must always meet setup and hold times t_{19} and t_{20} for correct operation.

Next Address Request (NA#)

This is used to request pipelining. This input indicates the system is prepared to accept new values of BHE#, BLE#, A_{23}–A_1, W/R#, D/C# and M/IO# from the 80376 even if the end of the current cycle is not being acknowledged on READY#. If this input is active when sampled, the next bus cycle's address and status signals are driven onto the bus, provided the next bus request is already pending internally. NA# is ignored in clock cycles in which ADS# or READY# is activated. This signal is active LOW and must satisfy setup and hold times t_{15} and t_{16} for correct operation. See **Pipelined Bus Cycles** (page 45) and **Read and Write Cycles** (page 42) for additional information.

BUS ARBITRATION SIGNALS (HOLD, HLDA)

This section describes the mechanism by which the processor relinquishes control of its local buses when requested by another bus master device. See **Entering and Exiting Hold Acknowledge** (page 52) for additional information.

Bus Hold Request (HOLD)

This input indicates some device other than the 80376 requires bus mastership. When control is granted, the 80376 floats A_{23}–A_1, BHE#, BLE#, D_{15}–D_0, LOCK#, M/IO#, D/C#, W/R# and ADS#, and then activates HLDA, thus entering the bus hold acknowledge state. The local bus will remain granted to the requesting master until HOLD becomes inactive. When HOLD becomes inactive, the 80376 will deactivate HLDA and drive the local bus (at the same time), thus terminating the hold acknowledge condition.

HOLD must remain asserted as long as any other device is a local bus master. External pull-up resistors may be required when in the hold acknowledge state since none of the 80376 floated outputs have internal pull-up resistors. See **Resistor Recommendations** (page 59) for additional information. HOLD is not recognized while RESET is active but is recognized during the time between the high-to-low transistion of RESET and the first instruction fetch. If RESET is asserted while HOLD is asserted, RESET has priority and places the bus into an idle state, rather than the hold acknowledge (high-impedance) state.

HOLD is a level-sensitive, active HIGH, synchronous input. HOLD signals must always meet setup and hold times t_{23} and t_{24} for correct operation.

Bus Hold Acknowledge (HLDA)

When active (HIGH), this output indicates the 80376 has relinquished control of its local bus in response to an asserted HOLD signal, and is in the bus Hold Acknowledge state.

The Bus Hold Acknowledge state offers near-complete signal isolation. In the Hold Acknowledge state, HLDA is the only signal being driven by the 80376. The other output signals or bidirectional signals (D_{15}–D_0, BHE#, BLE#, A_{23}–A_1, W/R#, D/C#, M/IO#, LOCK# and ADS#) are in a high-impedance state so the requesting bus master may control them. These pins remain OFF throughout the time that HLDA remains active (see Table 4.3). Pull-up resistors may be desired on several signals to avoid spurious activity when no bus master is driving them. See **Resistor Recommendations** (page 59) for additional information.

When the HOLD signal is made inactive, the 80376 will deactivate HLDA and drive the bus. One rising edge on the NMI input is remembered for processing after the HOLD input is negated.

Table 4.3. Output Pin State during HOLD

Pin Value	Pin Names
1	HLDA
Float	LOCK#, M/IO#, D/C#, W/R#, ADS#, A_{23}–A_1, BHE#, BLE#, D_{15}–D_0

In addition to the normal usage of Hold Acknowledge with DMA controllers or master peripherals, the near-complete isolation has particular attractiveness during system test when test equipment drives the system, and in hardware-fault-tolerant applications.

Hold Latencies

The maximum possible HOLD latency depends on the software being executed. The actual HOLD latency at any time depends on the current bus activity, the state of the LOCK# signal (internal to the CPU) activated by the LOCK# prefix, and interrupts. The 80376 will not honor a HOLD request until the current bus operation is complete. Table 4.4 shows the types of bus operations that can affect HOLD latency, and indicates the types of delays that

these operations may introduce. When considering maximum HOLD latencies, designers must select which of these bus operations are possible, and then select the maximum latency form among them.

The 80376 breaks 32-bit data or I/O accesses into 2 internally locked 16-bit bus cycles; the LOCK# signal is not asserted. The 80376 breaks unaligned 16-bit or 32-bit data or I/O accesses into 2 or 3 internally locked 16-bit bus cycles. Again the LOCK# signal is not asserted but a HOLD request will not be recognized until the end of the entire transfer.

As indicated in Table 4.4, wait states affect HOLD latency. The 80376 will not honor a HOLD request until the end of the current bus operation, no matter how many wait states are required. Systems with DMA where data transfer is critical must insure that READY# returns sufficiently soon.

Table 4.4. Locked Bus Operations Affecting HOLD Latency in Systems Clocks

}***Not Available At This Time***{

COPROCESSOR INTERFACE SIGNALS (PEREQ, BUSY#, ERROR#)

In the following sections are descriptions of signals dedicated to the numeric coprocessor interface. In addition to the data bus, address bus, and bus cycle definition signals, these following signals control communication between the 80376 and the 80387SX processor extension.

Coprocessor Request (PEREQ)

When asserted (HIGH), this input signal indicates a coprocessor request for a data operand to be transferred to/from memory by the 80376. In response, the 80376 transfers information between the coprocessor and memory. Because the 80376 has internally stored the coprocessor opcode being executed, it performs the requested data transfer with the correct direction and memory address.

PEREQ is a level-sensitive active HIGH asynchronous signal. Setup and hold times, t_{29} and t_{30}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. This signal is provided with a weak internal pull-down resistor of around 20 KΩ to ground so that it will not float active when left unconnected.

Coprocessor Busy (BUSY#)

When asserted (LOW), this input indicates the coprocessor is still executing an instruction, and is not yet able to accept another. When the 80376 encounters any coprocessor instruction which operates on the numerics stack (e.g. load, pop, or arithmetic operation), or the WAIT instruction, this input is first automatically sampled until it is seen to be inactive. This sampling of the BUSY# input prevents overrunning the execution of a previous coprocessor instruction.

The F(N)INIT, F(N)CLEX coprocessor instructions are allowed to execute even if BUSY# is active, since these instructions are used for coprocessor initialization and exception-clearing.

BUSY# is an active LOW, level-sensitive asynchronous signal. Setup and hold times, t_{29} and t_{30}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. This pin is provided with a weak internal pull-up resistor of around 20 KΩ to V_{CC} so that it will not float active when left unconnected.

BUSY# serves an additional function. If BUSY# is sampled LOW at the falling edge of RESET, the 80376 processor performs an internal self-test (see **Bus Activity During and Following Reset** on page 54). If BUSY# is sampled HIGH, no self-test is performed.

Coprocessor Error (ERROR#)

When asserted (LOW), this input signal indicates that the previous coprocessor instruction generated a coprocessor error of a type not masked by the coprocessor's control register. This input is automatically sampled by the 80376 when a coprocessor instruction is encountered, and if active, the 80376 generates exception 16 to access the error-handling software.

Several coprocessor instructions, generally those which clear the numeric error flags in the coprocessor or save coprocessor state, do execute without the 80376 generating exception 16 even if ERROR# is active. These instructions are FNINIT, FNCLEX, FNSTSW, FNSTSWAX, FNSTCW, FNSTENV and FNSAVE.

ERROR# is an active LOW, level-sensitive asynchronous signal. Setup and hold times t_{29} and t_{30}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. This pin is provided with a weak internal pull-up resistor of around 20 KΩ to V_{CC} so that it will not float active when left unconnected.

INTERRUPT SIGNALS (INTR, NMI, RESET)

The following descriptions cover inputs that can interrupt or suspend execution of the processor's current instruction stream.

Maskable Interrupt Request (INTR)

When asserted, this input indicates a request for interrupt service, which can be masked by the 80376 Flag Register IF bit. When the 80376 responds to the INTR input, it performs two interrupt acknowledge bus cycles and, at the end of the second, latches an 8-bit interrupt vector on D_7–D_0 to identify the source of the interrupt.

INTR is an active HIGH, level-sensitive asynchronous signal. Setup and hold times, t_{27} and t_{28}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. To assure recognition of an INTR request, INTR should remain active until the first interrupt acknowledge bus cycle begins. INTR is sampled at the beginning of every instruction. In order to be recognized at a particular instruction boundary, INTR must be active at least eight CLK2 clock periods before the beginning of the execution of the instruction. If recognized, the 80376 will begin execution of the interrupt.

Non-Maskable Interrupt Request (NMI)

This input indicates a request for interrupt service which cannot be masked by software. The non-maskable interrupt request is always processed according to the pointer or gate in slot 2 of the interrupt table. Because of the fixed NMI slot assignment, no interrupt acknowledge cycles are performed when processing NMI.

NMI is an active HIGH, rising edge-sensitive asynchronous signal. Setup and hold times, t_{27} and t_{28}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. To assure recognition of NMI, it must be inactive for at least eight CLK2 periods, and then be active for at least eight CLK2 periods before the beginning of the execution of an instruction.

Once NMI processing has begun, no additional NMI's are processed until after the next IRET instruction, which is typically the end of the NMI service routine. If NMI is re-asserted prior to that time, however, one rising edge on NMI will be remembered for processing after executing the next IRET instruction.

Interrupt Latency

The time that elapses before an interrupt request is serviced (interrupt latency) varies according to several factors. This delay must be taken into account by the interrupt source. Any of the following factors can affect interrupt latency:

1. If interrupts are masked, and INTR request will not be recognized until interrupts are reenabled.
2. If an NMI is currently being serviced, an incoming NMI request will not be recognized until the 80376 encounters the IRET instruction.
3. An interrupt request is recognized only on an instruction boundary of the 80376 ***Execution Unit*** except for the following cases:
   - — Repeat string instructions can be interrupted after each iteration.
   - — If the instruction loads the Stack Segment register, an interrupt is not processed until after the following instruction, which should be an ESP load. This allows the entire stack pointer to be loaded without interruption.
   - — If an instruction sets the interrupt flag (enabling interrupts), an interrupt is not processed until after the next instruction.

   The longest latency occurs when the interrupt request arrives while the 80376 processor is executing a long instruction such as multiplication, division or a task-switch.
4. Saving the Flags register and CS:EIP registers.
5. If interrupt service routine requires a task switch, time must be allowed for the task switch.
6. If the interrupt service routine saves registers that are not automatically saved by the 80376.

RESET

This input signal suspends any operation in progress and places the 80376 in a known reset state. The 80376 is reset by asserting RESET for 15 or more CLK2 periods (80 or more CLK2 periods before requesting self-test). When RESET is active, all other input pins are ignored, and all other bus pins are driven to an idle bus state as shown in Table 4.5. If RESET and HOLD are both active at a point in time, RESET takes priority even if the 80376 was in a Hold Acknowledge state prior to RESET active.

RESET is an active HIGH, level-sensitive synchronous signal. Setup and hold times, t_{25} and t_{26}, must be met in order to assure proper operation of the 80376.

Table 4.5. Pin State (Bus Idle) during RESET

Pin Name	Signal Level during RESET
ADS#	1
D_{15}–D_0	Float
BHE#, BLE#	0
A_{23}–A_1	1
W/R#	0
D/C#	1
M/IO#	0
LOCK#	1
HLDA	0

4.2 Bus Transfer Mechanism

All data transfers occur as a result of one or more bus cycles. Logical data operands of byte and word lengths may be transferred without restrictions on physical address alignment. Any byte boundary may be used, although two physical bus cycles are performed as required for unaligned operand transfers.

The 80376 processor address signals are designed to simplify external system hardware. BHE# and BLE# provide linear selects for the two bytes of the 16-bit data bus.

Byte Enable outputs BHE# and BLE# are asserted when their associated data bus bytes are involved with the present bus cycle, as listed in Table 4.6.

Table 4.6. Byte Enables and Associated Data and Operand Bytes

Byte Enable	Associated Data Bus Signals
BHE# BLE#	D_{15}–D_8 (Byte 1—Most Significant) D_7–D_0 (Byte 0—Least Significant)

Each bus cycle is composed of at least two bus states. Each bus state requires one processor clock period. Additional bus states added to a single bus cycle are called wait states. See **Bus Functional Description** (page 39) for additional information.

4.3 Memory and I/O Spaces

Bus cycles may access physical memory space or I/O space. Peripheral devices in the system may either be memory-mapped, or I/O-mapped, or both. As shown in Figure 4.3, physical memory addresses range from 000000H to 0FFFFFFH (16 Mbytes) and I/O addresses from 000000H to 00FFFFH (64 Kbytes). Note the I/O addresses used by the automatic I/O cycles for coprocessor communication are 8000F8H to 8000FFH, beyond the address range of programmed I/O, to allow easy generation of a coprocessor chip select signal using the A_{23} and M/IO# signals.

OPERAND ALIGNMENT

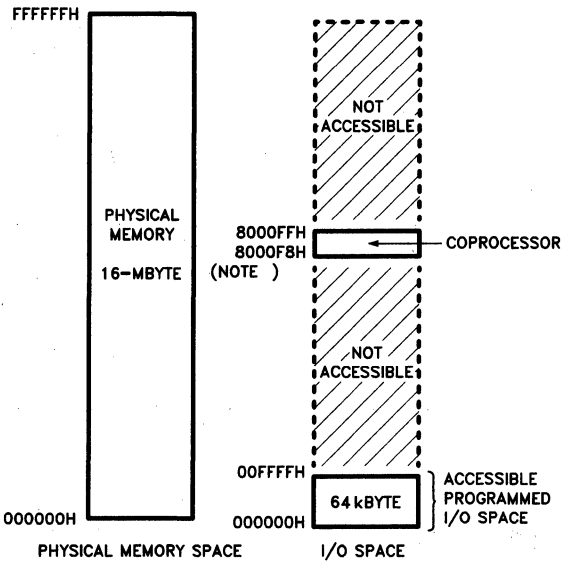
With the flexibility of memory addressing on the 80376, it is possible to transfer a logical operand that spans more than one physical Dword or word of memory or I/O. Examples are 32-bit Dword or 16-bit word operands beginning at addresses not evenly divisible by 2.

Operand alignment and size dictate when multiple bus cycles are required. Table 4.6a describes the transfer cycles generated for all combinations of logical operand lengths and alignment.

Table 4.6a. Transfer Bus Cycles for Bytes, Words and Dwords

	Byte-Length of Logical Operand								
	1	2				4			
Physical Byte Address in Memory (Low-Order Bits)	xx	00	01	10	11	00	01	10	11
Transfer Cycles	b	w	lb, hb	w	hb, l,b	lw, hw	hb, lb, mw	hw, lw	mw, hb, lb

Key: b = byte transfer
w = word transfer
l = low-order portion
m = mid-order portion
x = don't care
h = high-order portion



NOTE:
Since A_{23} is HIGH during automatic communication with coprocessor, A_{23} HIGH and M/IO# LOW can be used to easily generate a coprocessor select signal.

Figure 4.3. Physical Memory and I/O Spaces

4.4 Bus Functional Description

The 80376 has separate, parallel buses for data and address. The data bus is 16 bits in width, and bidirectional. The address bus provides a 24-bit value using 23 signals for the 23 upper-order address bits and 2 Byte Enable signals to directly indicate the active bytes. These buses are interpreted and controlled by several definition signals.

The definition of each bus cycle is given by three signals: M/IO#, W/R# and D/C#. At the same time, a valid address is present on the byte enable signals, BHE# and BLE#, and the other address signals A_{23}–A_1. A status signal, ADS#, indicates when the 80376 issues a new bus cycle definition and address.

Collectively, the address bus, data bus and all associated control signals are referred to simply as "the bus". When active, the bus performs one of the bus cycles below:

1. Read from memory space
2. Locked read from memory space
3. Write to memory space
4. Locked write to memory space
5. Read from I/O space (or coprocessor)
6. Write to I/O space (or coprocessor)
7. Interrupt acknowledge (always locked)
8. Indicate halt, or indicate shutdown

Table 4.2 shows the encoding of the bus cycle definition signals for each bus cycle. See **Bus Cycle Definition Signals** (page 35) for additonal information.

When the 80376 bus is not performing one of the activities listed above, it is either Idle or in the Hold Acknowledge state, which may be detected by external circuitry. The idle state can be identified by the 80376 giving no further assertions on its address strobe output (ADS#) since the beginning of its most recent bus cycle, and the most recent bus cycle having been terminated. The hold acknowledge state is identified by the 80376 asserting its hold acknowledge (HLDA) output.

The shortest time unit of bus activity is a bus state. A bus state is one processor clock period (two CLK2 periods) in duration. A complete data transfer occurs during a bus cycle, composed of two or more bus states.

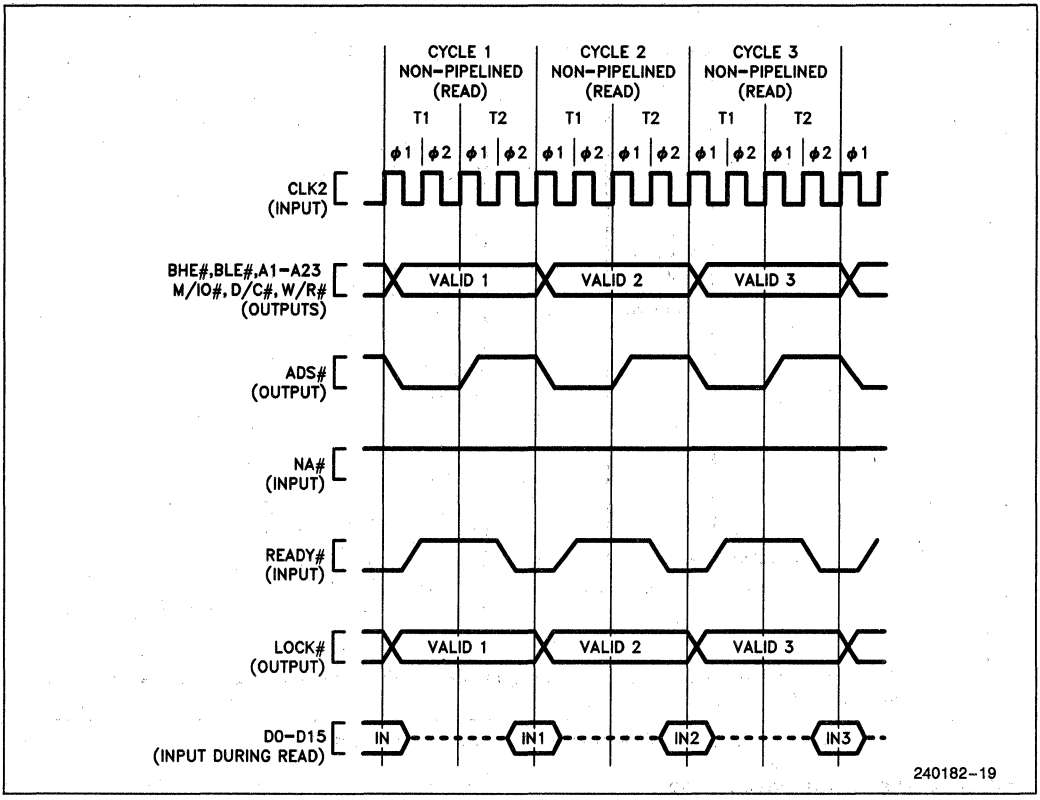


Figure 4.4. Fastest Read Cycles with Non-Pipelined Timing

The fastest 80376 bus cycle requires only two bus states. For example, three consecutive bus read cycles, each consisting of two bus states, are shown by Figure 4.4. The bus states in each cycle are named T1 and T2. Any memory or I/O address may be accessed by such a two-state bus cycle, if the external hardware is fast enough.

Every bus cycle continues until it is acknowledged by the external system hardware, using the 80376 READY# input. Acknowledging the bus cycle at the end of the first T2 results in the shortest bus cycle, requiring only T1 and T2. If READY# is not immediately asserted however, T2 states are repeated indefinitely until the READY# input is sampled active.

The pipelining option provides a choice of bus cycle timings. Pipelined or non-pipelined cycles are selectable on a cycle-by-cycle basis with the Next Address (NA#) input.

When pipelining is selected the address (BHE#, BLE# and A_{23}–A_1) and definition (W/R#, D/C#, M/IO# and LOCK#) of the next cycle are available before the end of the current cycle. To signal their availability, the 80376 address status output (ADS#) is asserted. Figure 4.5 illustrates the fastest read cycles with pipelined timing.

Note from Figure 4.5 the fastest bus cycles using pipelining require only two bus states, named **T1P** and **T2P**. Therefore pipelined cycles allow the same data bandwidth as non-pipelined cycles, but address-to-data access time is increased by one T-state time compared to that of a non-pipelined cycle.

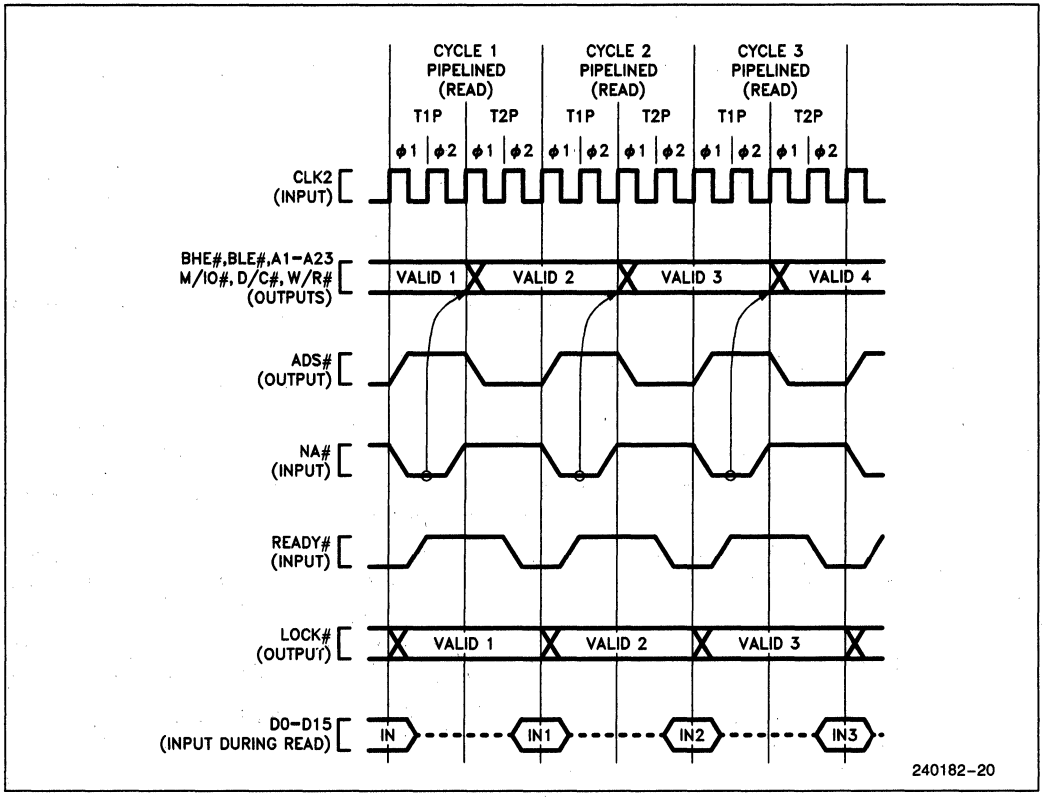


Figure 4.5. Fastest Read Cycles with Pipelined Timing

READ AND WRITE CYCLES

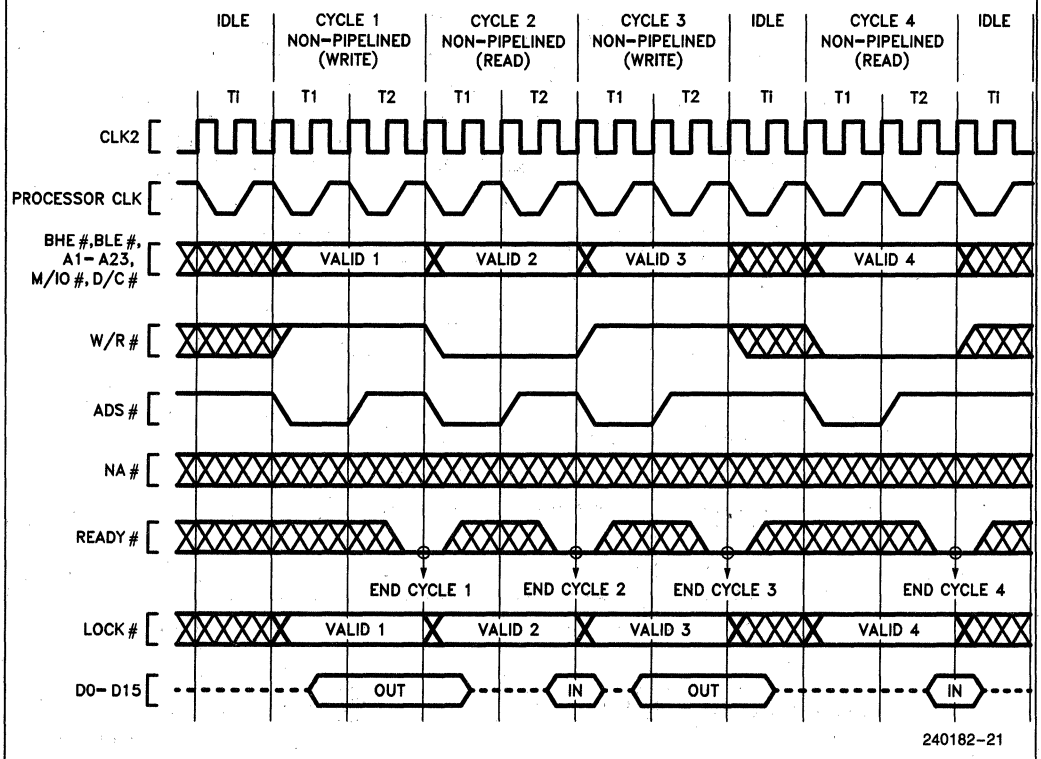
Data transfers occur as a result of bus cycles, classified as read or write cycles. During read cycles, data is transferred from an external device to the processor. During write cycles, data is transferred from the processor to an external device.

Two choices of bus cycle timing are dynamically selectable: non-pipelined or pipelined. After an idle bus state, the processor always uses non-pipelined timing. However the NA# (Next Address) input may be asserted to select pipelined timing for the next bus cycle. When pipelining is selected and the 80376 has a bus request pending internally, the address and definition of the next cycle is made available even before the current bus cycle is acknowledged by READY#.

Terminating a read or write cycle, like any bus cycle, requires acknowledging the cycle by asserting the READY# input. Until acknowledged, the processor inserts wait states into the bus cycle, to allow adjustment for the speed of any external device. External hardware, which has decoded the address and bus cycle type, asserts the READY# input at the appropriate time.

At the end of the second bus state within the bus cycle, READY# is sampled. At that time, if external hardware acknowledges the bus cycle by asserting READY#, the bus cycle terminates as shown in Figure 4.6. If READY# is negated as in Figure 4.7, the 80376 executes another bus state (a wait state) and READY# is sampled again at the end of that state. This continues indefinitely until the cycle is acknowledged by READY# asserted.

When the current cycle is acknowledged, the 80376 terminates it. When a read cycle is acknowledged, the 80376 latches the information present at its data pins. When a write cycle is acknowledged, the write data of the 80376 remains valid throughout phase one of the next bus state, to provide write data hold time.



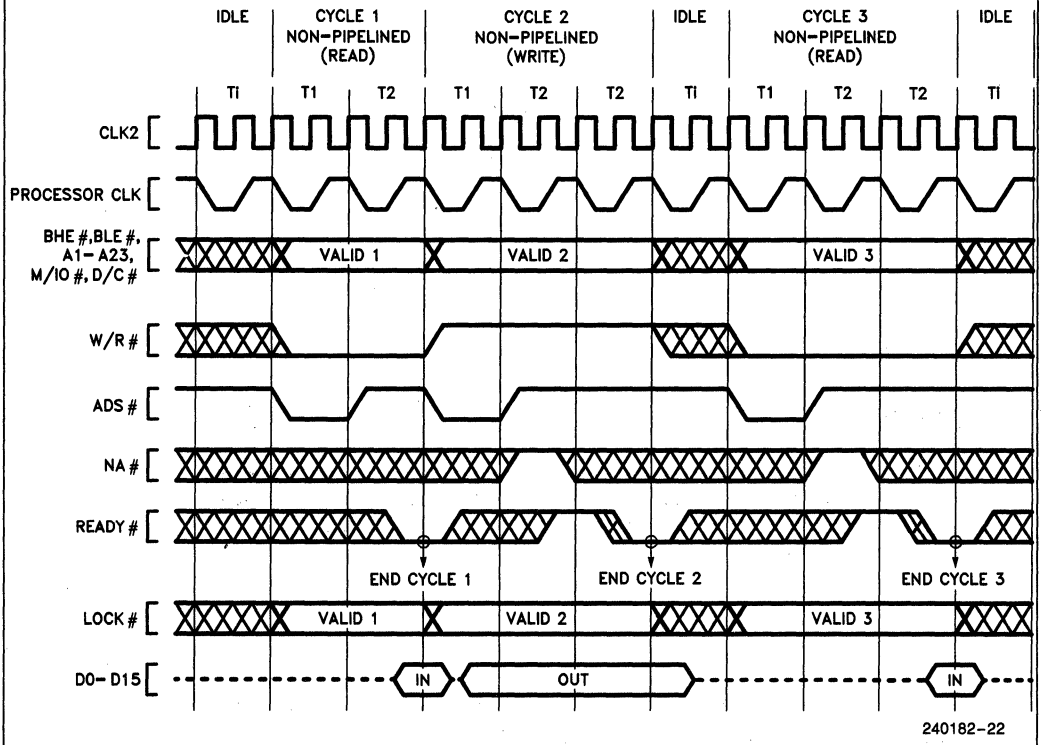
Idle states are shown here for diagram variety only. Write cycles are not always followed by an idle state. An active bus cycle can immediately follow the write cycle.

Figure 4.6. Various Non-Pipelined Bus Cycles (Zero Wait States)

Non-Pipelined Bus Cycles

Any bus cycle may be performed with non-pipelined timing. For example, Figure 4.6 shows a mixture of non-pipelined read and write cycles. Figure 4.6 shows that the fastest possible non-pipelined cycles have two bus states per bus cycle. The states are named T1 and T2. In phase one of T1, the address signals and bus cycle definition signals are driven valid and, to signal their availability, address strobe (ADS#) is simultaneously asserted.

During read or write cycles, the data bus behaves as follows. If the cycle is a read, the 80376 floats its data signals to allow driving by the external device being addressed. **The 80376 requires that all data bus pins be at a valid logic state (HIGH or LOW) at the end of each read cycle, when READY# is asserted. The system MUST be designed to meet this requirement.** If the cycle is a write, data signals are driven by the 80376 beginning in phase two of T1 until phase one of the bus state following cycle acknowledgement.



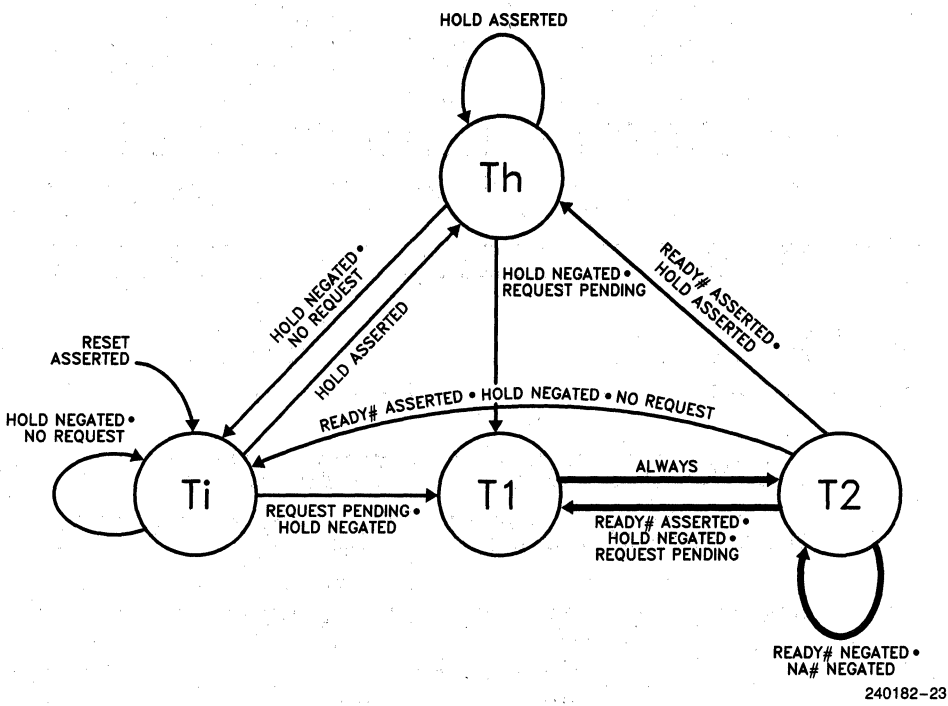
Idle states are shown here for diagram variety only. Write cycles are not always followed by an idle state. An active bus cycle can immediately follow the write cycle.

Figure 4.7. Various Non-Pipelined Bus Cycles (Various Number of Wait States)

Figure 4.7 illustrates non-pipelined bus cycles with one wait state added to Cycles 2 and 3. READY# is sampled inactive at the end of the first T2 in Cycles 2 and 3. Therefore Cycles 2 and 3 have T2 repeated again. At the end of the second T2, READY# is sampled active.

When address pipelining is not used, the address and bus cycle definition remain valid during all wait states. When wait states are added and it is desirable to maintain non-pipelined timing, it is necessary to negate NA# during each T2 state except the last one, as shown in Figure 4.7, Cycles 2 and 3. If NA# is sampled active during a T2 other than the last one, the next state would be T2I or T2P instead of another T2.

When address pipelining is not used, the bus states and transitions are completely illustrated by Figure 4.8. The bus transitions between four possible states, T1, T2, T_i, and T_h. Bus cycles consist of T1 and T2, with T2 being repeated for wait states. Otherwise the bus may be idle, T_i, or in the hold acknowledge state T_h.



Bus States:
T1—first clock of a non-pipelined bus cycle (80376 drives new address and asserts ADS#).
T2—subsequent clocks of a bus cycle when NA# has not been sampled asserted in the current bus cycle.
Ti—idle state.
Th—hold acknowledge state (80376 asserts HLDA).
The fastest bus cycle consists of two states: T1 and T2.
Four basic bus states describe bus operation when not using pipelined address.

Figure 4.8. 80376 Bus States (Not Using Pipelined Address)

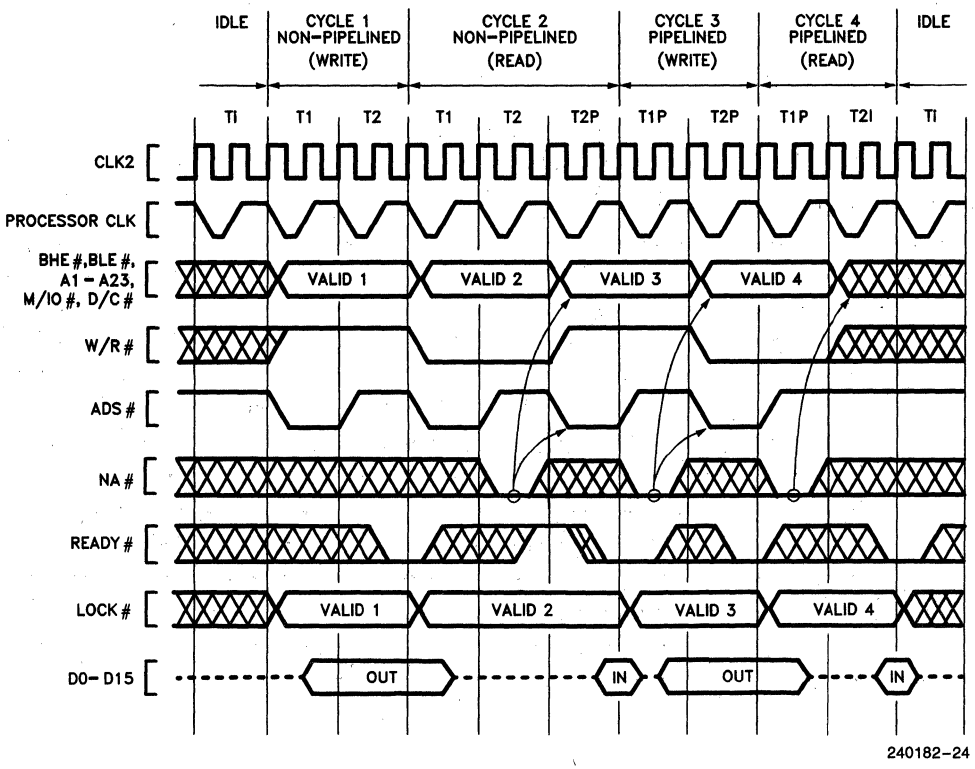
Bus cycles always begin with T1. T1 always leads to T2. If a bus cycle is not acknowledged during T2 and NA# is inactive, T2 is repeated. When a cycle is acknowledged during T2, the following state will be T1 of the next bus cycle if a bus request is pending internally, or T_i if there is no bus request pending, or T_h if the HOLD input is being asserted.

Use of pipelining allows the 80376 to enter three additional bus states not shown in Figure 4.8. Figure 4.12 on page 49 is the complete bus state diagram, including pipelined cycles.

Pipelined Bus Cycles

Pipelining is the option of requesting the address and the bus cycle definition of the next internally pending bus cycle before the current bus cycle is acknowledged with READY# asserted. ADS# is asserted by the 80376 when the next address is issued. The pipelining option is controlled on a cycle-by-cycle basis with the NA# input signal.

Once a bus cycle is in progress and the current address has been valid for at least one entire bus state, the NA# input is sampled at the end of every phase one until the bus cycle is acknowledged. During non-pipelined bus cycles NA# is sampled at the end of phase one in every T2. An example is Cycle 2 in Figure 4.9, during which NA# is sampled at the end of phase one of every T2 (it was asserted once during the first T2 and has no further effect during that bus cycle).



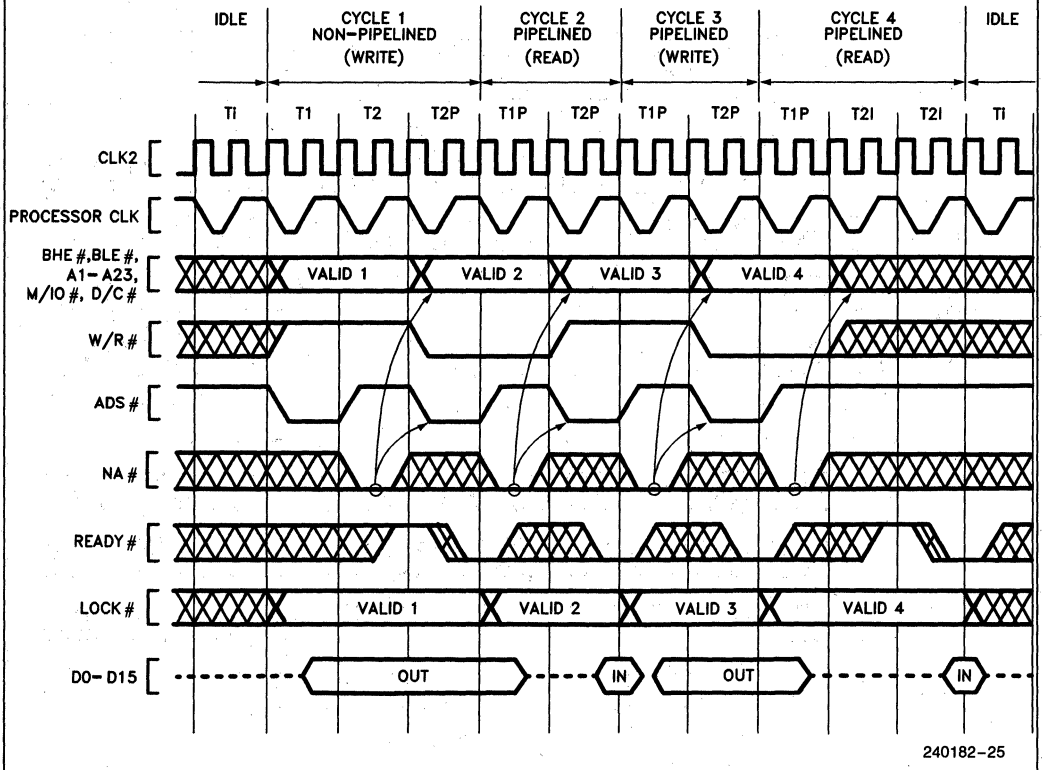
Following any idle bus state (Ti), bus cycles are non-pipelined. Within non-pipelined bus cycles, NA# is only sampled during wait states. Therefore, to begin pipelining during a group of non-pipelined bus cycles requires a non-pipelined cycle with at least one wait state (Cylcle 2 above).

Figure 4.9. Transitioning to Pipelining during Burst of Bus Cycles

If NA# is sampled active, the 80376 is free to drive the address and bus cycle definition of the next bus cycle, and assert ADS#, as soon as it has a bus request internally pending. It may drive the next address as early as the next bus state, whether the current bus cycle is acknowledged at that time or not.

Regarding the details of pipelining, the 80376 has the following characteristics:

1. The next address and status may appear as early as the bus state after NA# was sampled active (see Figures 4.9 or 4.10). In that case, state T2P is entered immediately. However, when there is not an internal bus request already pending, the next address and status will not be available immediately after NA# is asserted and T2I is entered instead of T2P (see Figure 4.11 Cycle 3). Provided the current bus cycle isn't yet acknowledged by READY# asserted, T2P will be entered as soon as the 80376 does drive the next address and status. External hardware should therefore observe the ADS# output as confirmation the next address and status are actually being driven on the bus.
2. Any address and status which are validated by a pulse on the 80376 ADS# output will remain stable on the address pins for at least two processor clock periods. The 80376 cannot produce a new address and status more frequently than every two processor clock periods (see Figures 4.9, 4.10 and 4.11).
3. Only the address and bus cycle definition of the very next bus cycle is available. The pipelining capability cannot look further than one bus cycle ahead (see Figure 4.11, Cycle 1).



Following any idle bus state (Ti) the bus cycle is always non-pipelined and NA# is only sampled during wait states. To start, address pipelining after an idle state requires a non-pipelined cycle with at least one wait state (cycle 1 above). The pipelined cycles (2, 3, 4 above) are shown with various numbers of wait states.

Figure 4.10. Fastest Transition to Pipelined Bus Cycle Following Idle Bus State

The complete bus state transition diagram, including pipelining is given by Figure 4.12. Note it is a superset of the diagram for non-pipelined only, and the three additional bus states for pipelining are drawn in bold.

The fastest bus cycle with pipelining consists of just two bus states, T1P and T2P (recall for non-pipelined it is T1 and T2). T1P is the first bus state of a pipelined cycle.

Initiating and Maintaining Pipelined Bus Cycles

Using the state diagram Figure 4.12, observe the transitions from an idle state, T_i, to the beginning of a pipelined bus cycle T1P. From an idle state, T_i, the first bus cycle must begin with T1, and is therefore a non-pipelined bus cycle. The next bus cycle will be pipelined, however, provided NA# is asserted and the first bus cycle ends in a T2P state (the address and status for the next bus cycle is driven during T2P). The fastest path from an idle state to a pipelined bus cycle is shown in bold below:

T_I, T_I,	**T1-T2-T2P,**	**T1P-T2P,**
idle states	non-pipelined cycle	pipelined cycle

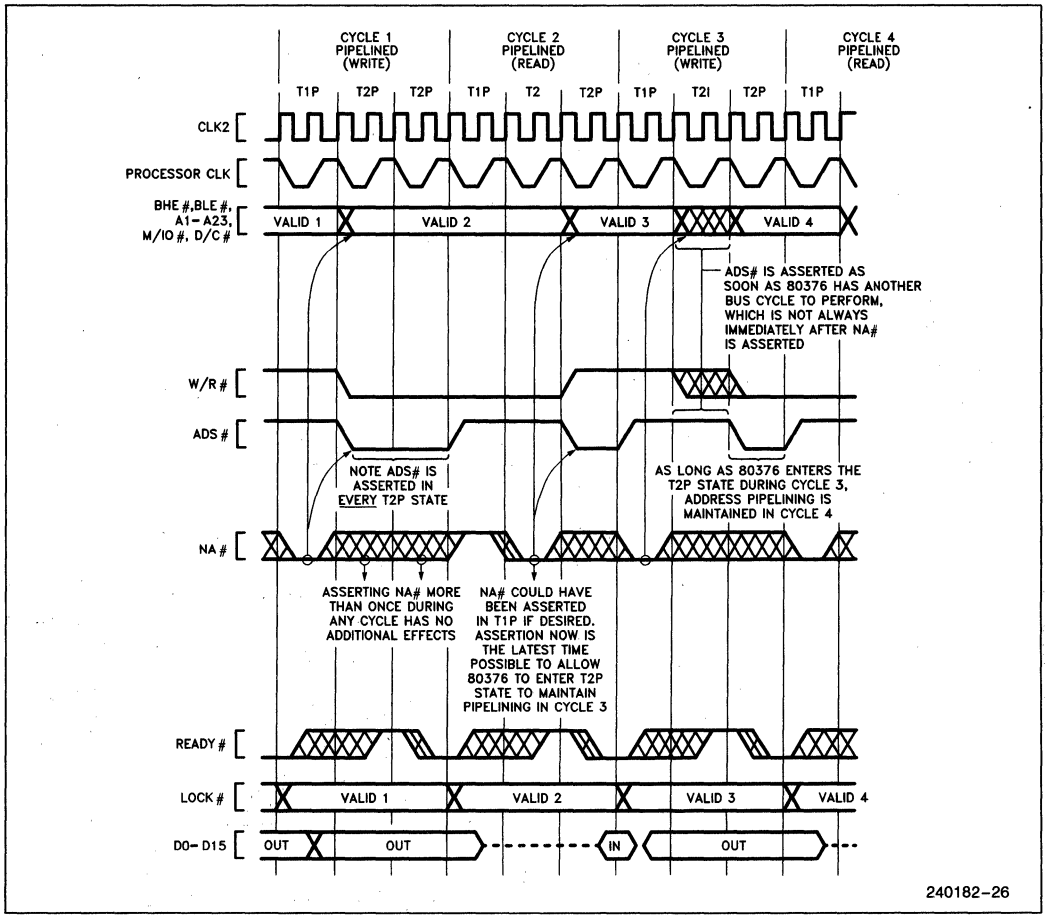


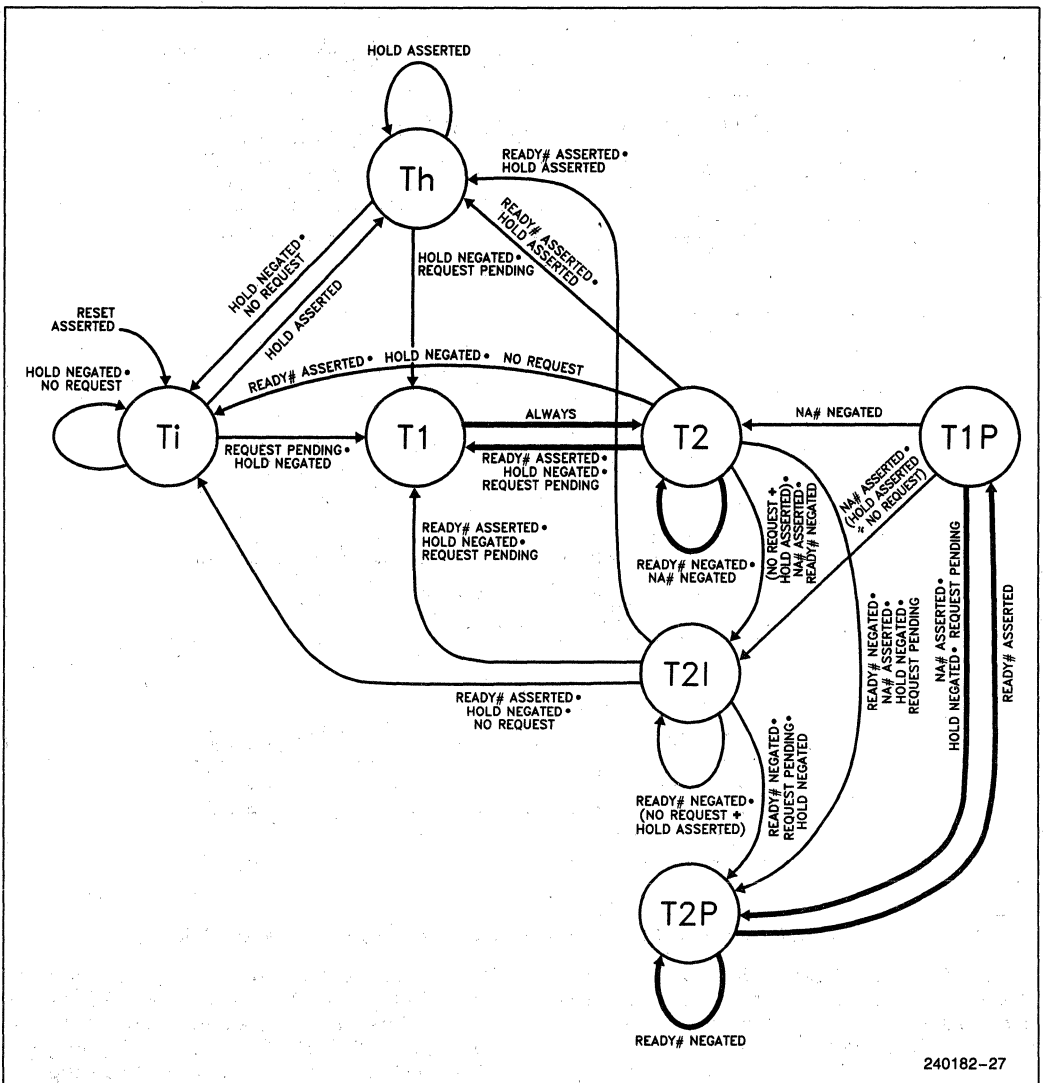
Figure 4.11. Details of Address Pipelining during Cycles with Wait States

T1–T2–T2P are the states of the bus cycle that establishes address pipelining for the next bus cycle, which begins with T1P. The same is true after a bus hold state, shown below:

T_h, T_h, T_h,	T1–T2–T2P,	T1P–T2P,
hold aknowledge states	non-pipelined cycle	pipelined cycle

The transition to pipelined address is shown functionally by Figure 4.10, Cycle 1. Note that Cycle 1 is used to transition into pipelined address timing for the subsequent Cycles 2, 3 and 4, which are pipelined. The NA# input is asserted at the appropriate time to select address pipelining for Cycles 2, 3 and 4.

Once a bus cycle is in progress and the current address and status has been valid for one entire bus state, the NA# input is sampled at the end of every phase one until the bus cycle is acknowledged.



Bus States:

T1—first clock of a non-pipelined bus cycle (80376 drives new address, status and asserts ADS#).
T2—subsequent clocks of a bus cycle when NA# has not been sampled asserted in the current bus cycle.
T2I—subsequent clocks of a bus cycle when NA# has been sampled asserted in the current bus cycle but there is not yet an internal bus request pending (80376 will not drive new address, status or assert ADS#).
T2P—subsequent clocks of a bus cycle when NA# has been sampled asserted in the current bus cycle and there is an internal bus request pending (80376 drives new address, status and asserts ADS#).
T1P—first clock of a pipelined bus cycle.
Ti—idle state.
Th—hold acknowledge state (80376 asserts HLDA).

Asserting NA# for pipelined bus cycles gives access to three more bus states: T2I, T2P and T1P.
Using pipelining the fastest bus cycle consists of T1P and T2P.

Figure 4.12. 80376 Processor Complete Bus States (Including Pipelining)

Sampling begins in T2 during Cycle 1 in Figure 4.10. Once NA# is sampled active during the current cycle, the 80376 is free to drive a new address and bus cycle definition on the bus as early as the next bus state. In Figure 4.10, Cycle 1 for example, the next address and status is driven during state T2P. Thus Cycle 1 makes the transition to pipelined timing, since it begins with T1 but ends with T2P. Because the address for Cycle 2 is available before Cycle 2 begins, Cycle 2 is called a pipelined bus cycle, and it begins with T1P. Cycle 2 begins as soon as READY# asserted terminates Cycle 1.

Examples of transition bus cycles are Figure 4.10, Cycle 1 and Figure 4.9, Cycle 2. Figure 4.10 shows transition during the very first cycle after an idle bus state, which is the fastest possible transition into address pipelining. Figure 4.9, Cycle 2 shows a transition cycle occurring during a burst of bus cycles. In any case, a transition cycle is the same whenever it occurs: it consists at least of T1, T2 (NA# is asserted at that time), and T2P (provided the 80376 has an internal bus request already pending, which it almost always has). T2P states are repeated if wait states are added to the cycle.

Note that only three states (T1, T2 and T2P) are required in a bus cycle performing a **transition** from non-pipelined into pipelined timing, for example Figure 4.10, Cycle 1. Figure 4.10, Cycles 2, 3 and 4 show that pipelining can be maintained with two-state bus cycles consisting only of T1P and T2P.

Once a pipelined bus cycle is in progress, pipelined timing is maintained for the next cycle by asserting NA# and detecting that the 80376 enters T2P during the current bus cycle. The current bus cycle must end in state T2P for pipelining to be maintained in the next cycle. T2P is identified by the assertion of ADS#. Figures 4.9 and 4.10 however, each show pipelining ending after Cycle 4 because Cycle 4 ends in T2I. This indicates the 80376 didn't have an internal bus request prior to the acknowledgement of Cycle 4. If a cycle ends with a T2 or T2I, the next cycle will not be pipelined.

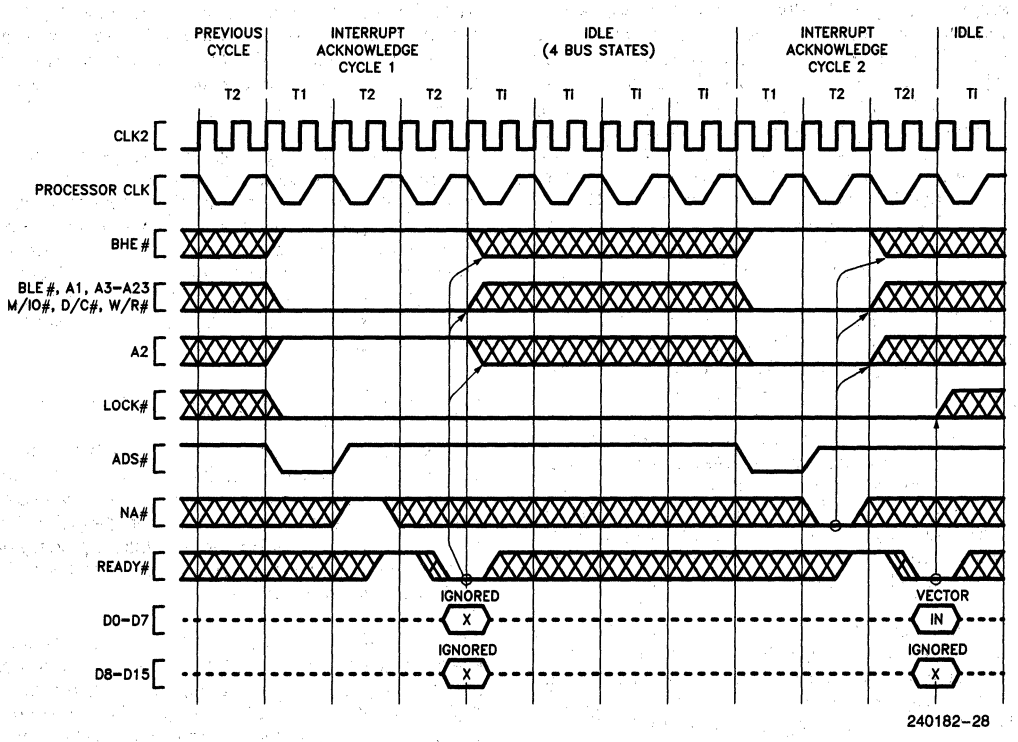
Realistically, pipelining is almost always maintained as long as NA# is sampled asserted. This is so because in the absence of any other request, a code prefetch request is always internally pending until the instruction decoder and code prefetch queue are completely full. Therefore pipelining is maintained for long bursts of bus cycles, if the bus is available (i.e., HOLD inactive) and NA# is sampled active in each of the bus cycles.

INTERRUPT ACKNOWLEDGE (INTA) CYCLES

In repsonse to an interrupt request on the INTR input when interrupts are enabled, the 80376 performs two interrupt acknowledge cycles. These bus cycles are similar to read cycles in that bus definition signals define the type of bus activity taking place, and each cycle continues until acknowledged by READY# sampled active.

The state of A_2 distinguishes the first and second interrupt acknowledge cycles. The byte address driven during the first interrupt acknowledge cycle is 4 (A_{23}–A_3, A_1, BLE# LOW, A_2 and BHE# HIGH). The byte address driven during the second interrupt acknowledge cycle is 0 (A_{23}–A_1, BLE# LOW and BHE# HIGH).

The LOCK# output is asserted from the beginning of the first interrupt acknowledge cycle until the end of the second interrupt acknowledge cycle. Four idle bus states, T_i, are inserted by the 80376 between the two interrupt acknowledge cycles for compatibility with the interrupt specification T_{RHRL} of the 8259A Interrupt Controller and the 82370 Integrated Peripheral.



Interrupt Vector (0–255) is read on D0–D7 at end of second Interrupt Acknowledge bus cycle.
Because each Interrupt Acknowledge bus cycle is followed by idle bus states, asserting NA# has no practical effect. Choose the approach which is simplest for your system hardware design.

Figure 4.13. Interrupt Acknowledge Cycles

During both interrupt acknowledge cycles, D_{15}–D_0 float. No data is read at the end of the first interrupt acknowledge cycle. At the end of the second interrupt acknowledge cycle, the 80376 will read an external interrupt vector from D_7–D_0 of the data bus. The vector indicates the specific interrupt number (from 0–255) requiring service.

HALT INDICATION CYCLE

The 80376 execution unit halts as a result of executing a HLT instruction. Signaling its entrance into the halt state, a halt indication cycle is performed. The halt indication cycle is identified by the state of the bus definition signals shown on page 34, **Bus Cycle Definition Signals,** and a byte address of 2. The halt indication cycle must be acknowledged by READY# asserted. A halted 80376 resumes execution when INTR (if interrupts are enabled), NMI or RESET is asserted.

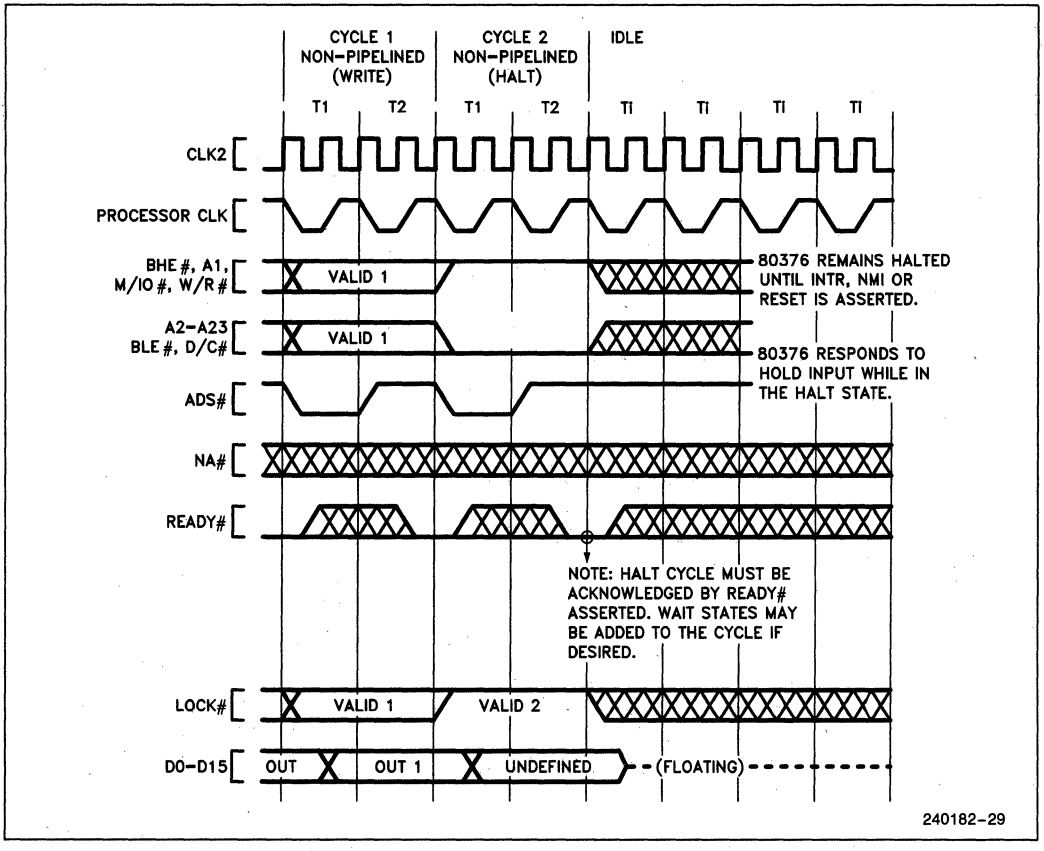


Figure 4.14. Example Halt Indication Cycle from Non-Pipelined Cycle

SHUTDOWN INDICATION CYCLE

The 80376 shuts down as a result of a protection fault while attempting to process a double fault. Signaling its entrance into the shutdown state, a shutdown indication cycle is performed. The shutdown indication cycle is identified by the state of the bus definition signals shown on page 34 **Bus Cycle Definition Signals** and a byte address of 0. The shutdown indication cycle must be acknowledged by READY# asserted. A shutdown 80376 resumes execution when NMI or RESET is asserted.

ENTERING AND EXITING HOLD ACKNOWLEDGE

The bus hold acknowledge state, T_h, is entered in response to the HOLD input being asserted. In the bus hold acknowledge state, the 80376 floats all outputs or bidirectional signals, except for HLDA. HLDA is asserted as long as the 80376 remains in the bus hold acknowledge state. In the bus hold acknowledge state, all inputs except HOLD and RESET are ignored.

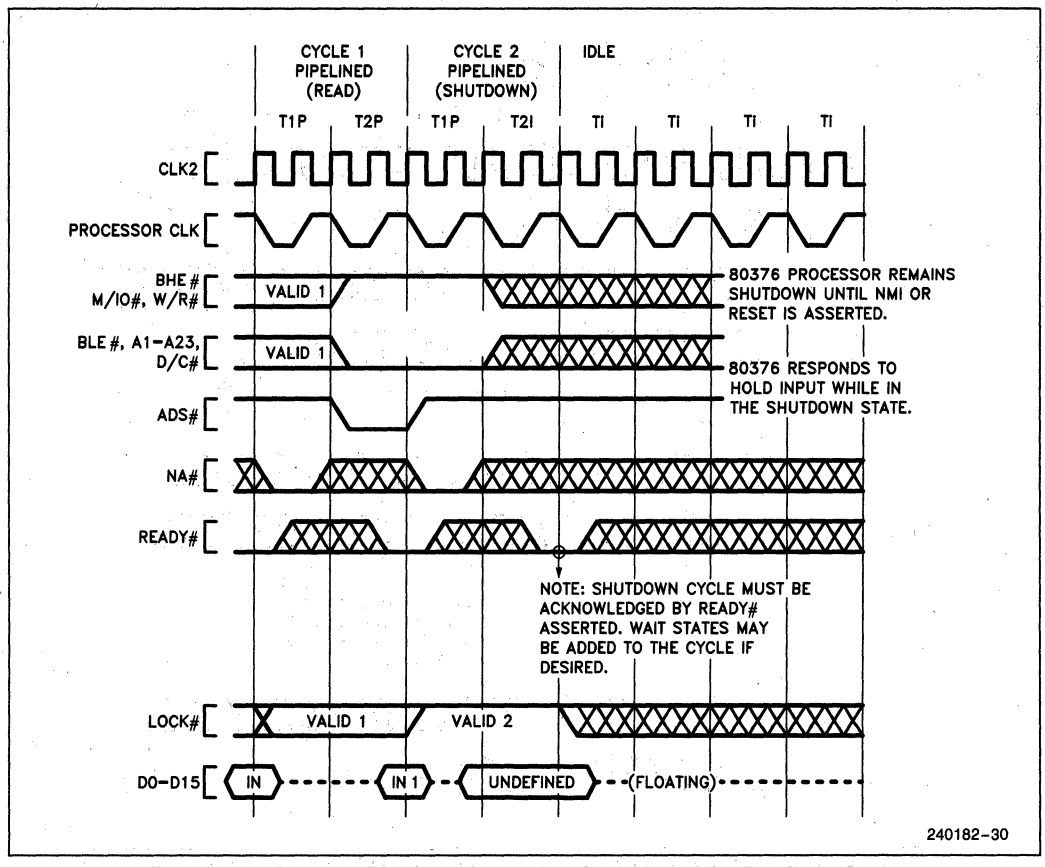
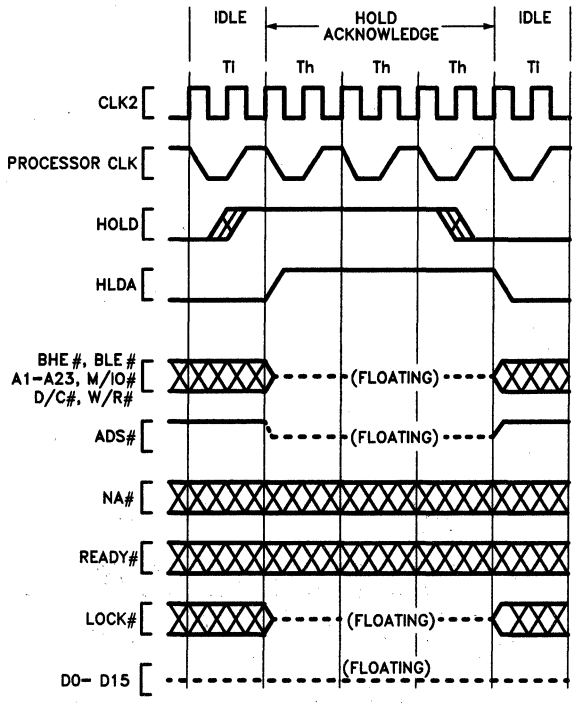


Figure 4.15. Example Shutdown Indication Cycle from Non-Pipelined Cycle

T_h may be entered from a bus idle state as in Figure 4.16 or after the acknowledgement of the current physical bus cycle if the LOCK# signal is not asserted, as in Figures 4.17 and 4.18.

T_h is exited in response to the HOLD input being negated. The following state will be T_i as in Figure 4.16 if no bus request is pending. The following bus state will be T1 if a bus request is internally pending, as in Figures 4.17 and 4.18. T_h is exited in response to RESET being asserted.

If a rising edge occurs on the edge-triggered NMI input while in T_h, the event is remembered as a non-maskable interrupt 2 and is serviced when T_h is exited unless the 80376 is reset before T_h is exited.



NOTE:
For maximum design flexibility the 80376 has no internal pull-up resistors on its outputs. Your design may require an external pullup on ADS# and other 80376 outputs to keep them negated during float periods.

Figure 4.16. Requesting Hold from Idle Bus

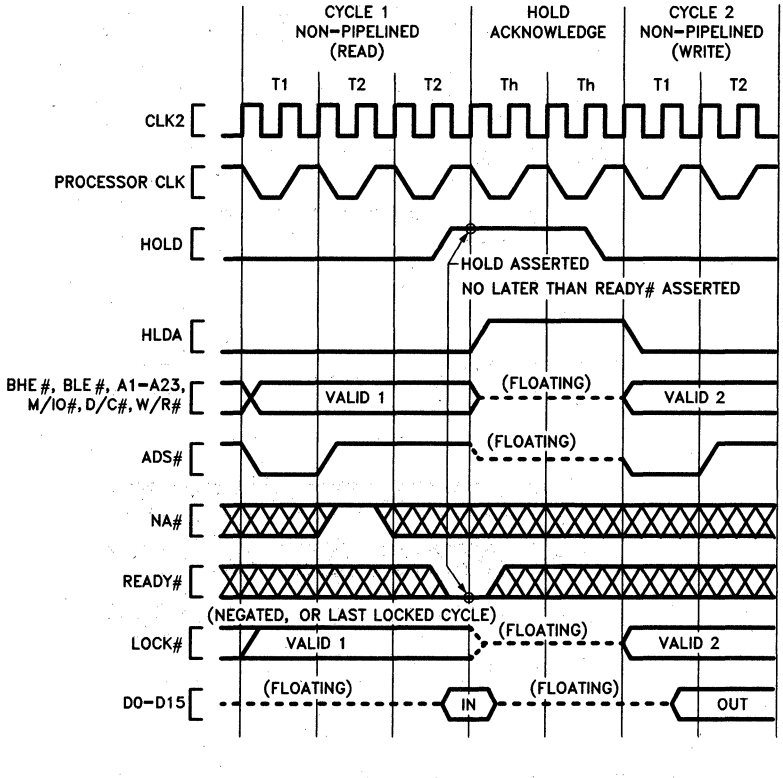
RESET DURING HOLD ACKNOWLEDGE

RESET being asserted takes priority over HOLD being asserted. If RESET is asserted while HOLD remains asserted, the 80376 drives its pins to defined states during reset, as in **Table 4.5, Pin State During Reset,** and performs internal reset activity as usual.

If HOLD remains asserted when RESET is inactive, the 80376 enters the hold acknowledge state before performing its first bus cycle, provided HOLD is still asserted when the 80376 processor would otherwise perform its first bus cycle. If HOLD remains asserted when RESET is inactive, the BUSY# input is still sampled as usual to determine whether a self test is being requested.

BUS ACTIVITY DURING AND FOLLOWING RESET

RESET is the highest priority input signal, capable of interrupting any processor activity when it is asserted. A bus cycle in progress can be aborted at any stage, or idle states or bus hold acknowledge states discontinued so that the reset state is established.

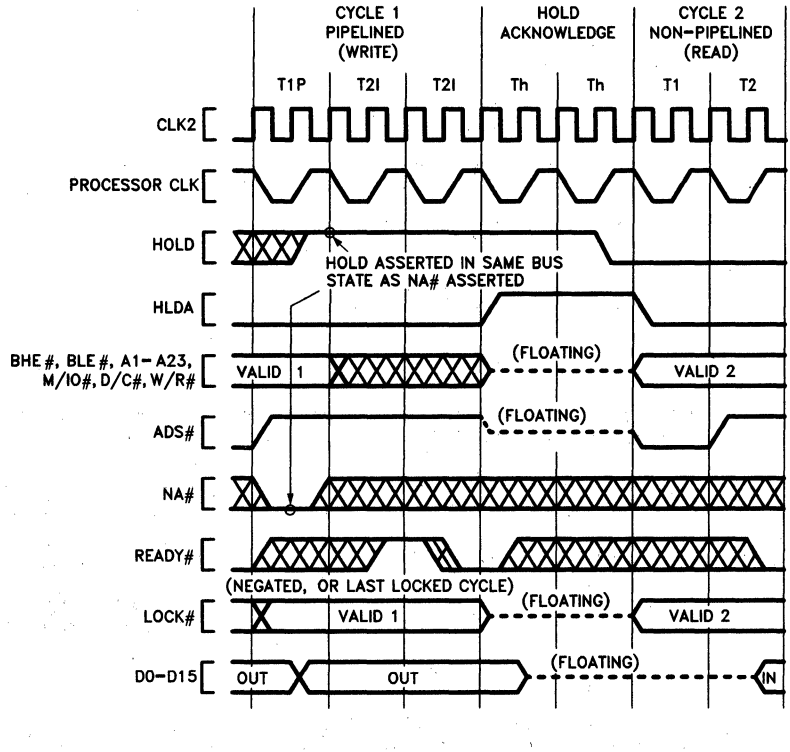


NOTE:
HOLD is a synchronous input and can be asserted at any CLK2 edge, provided setup and hold (t_{23} and t_{24}) requirements are met. This waveform is useful for determining Hold Acknowledge latency.

Figure 4.17. Requesting Hold from Active Bus (NA# Inactive)

RESET should remain asserted for at least 15 CLK2 periods to ensure it is recognized throughout the 80376, and at least 80 CLK2 periods if a 80376 self-test is going to be requested at the falling edge. RESET asserted pulses less than 15 CLK2 periods may not be recognized. RESET pulses less than 80 CLK2 periods followed by a self-test may cause the self-test to report a failure when no true failure exists.

Provided the RESET falling edge meets setup and hold times t_{25} and t_{26}, the internal processor clock phase is defined at that time as illustrated by Figure 4.19 and Figure 6.7.

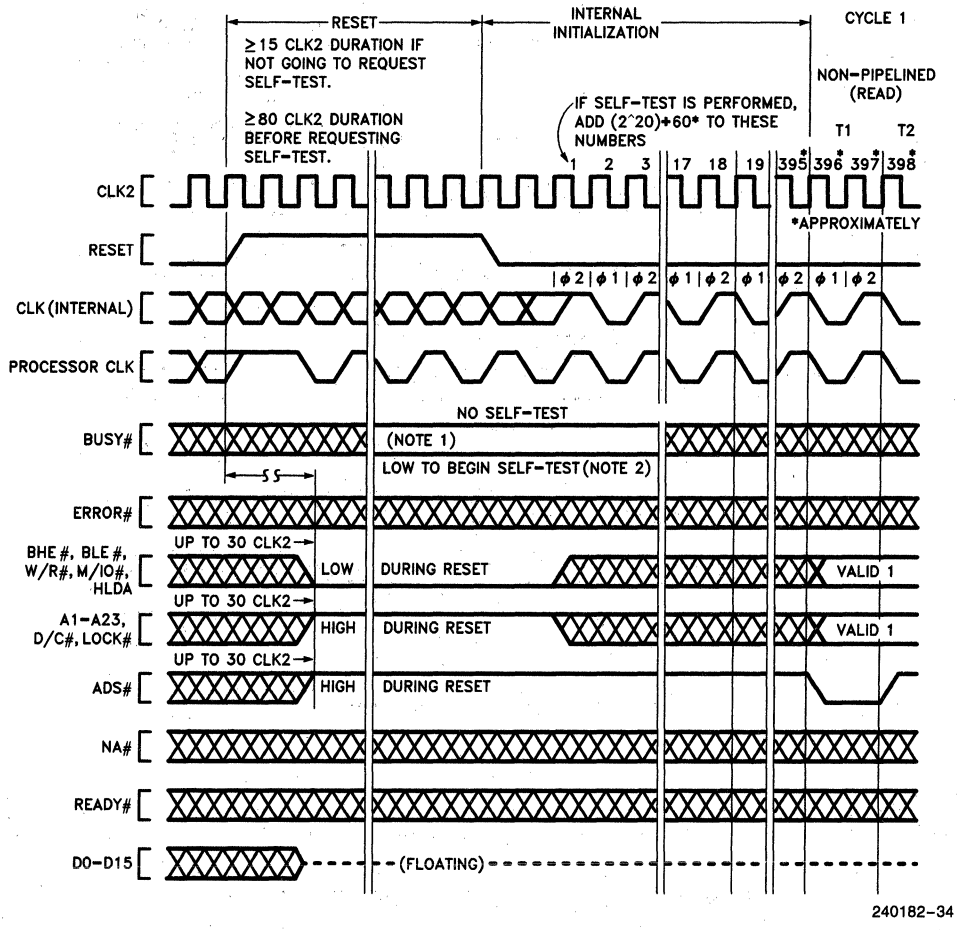


NOTE:
HOLD is a synchronous input and can be asserted at any CLK2 edge, provided setup and hold (t_{23} and t_{24}) requirements are met. This waveform is useful for determining Hold Acknowledge latency.

Figure 4.18. Requesting Hold from Idle Bus (NA# Active)

An 80376 self-test may be requested at the time RESET goes inactive by having the BUSY# input at a LOW level as shown in Figure 4.19. The self-test requires (2^{20} + approximately 60) CLK2 periods to complete. The self-test duration is not affected by the test results. Even if the self-test indicates a problem, the 80376 attempts to proceed with the reset sequence afterwards.

After the RESET falling edge (and after the self-test if it was requested) the 80376 performs an internal initialization sequence for approximately 350 to 450 CLK2 periods.



NOTES:
1. BUSY# should be held stable for 8 CLK2 periods before and after the CLK2 period in which RESET falling edge occurs.
2. If self-test is requested, the 80376 outputs remain in their reset state as shown here.

Figure 4.19. Bus Activity from Reset until First Code Fetch

4.5 Self-Test Signature

Upon completion of self-test (if self-test was requested by driving BUSY# LOW at the falling edge of RESET) the EAX register will contain a signature of 00000000H indicating the 80376 passed its self-test of microcode and major PLA contents with no problems detected. The passing signature in EAX, 00000000H, applies to all 80376 revision levels. Any non-zero signature indicates the 80376 unit is faulty.

4.6 Component and Revision Identifiers

To assist 80376 users, the 80376 after reset holds a component identifier and revision identifier in its DX register. The upper 8 bits of DX hold 33H as identification of the 80376 component. (The lower nibble, 03H, refers to the Intel386™ architecture. The upper nibble, 30H, refers to the third member of the Intel386 family). The lower 8 bits of DX hold an 8-bit unsigned binary number related to the

component revision level. The revision identifier will, in general, chronologically track those component steppings which are intended to have certain improvements or distinction from previous steppings. The 80376 revision identifier will track that of the 80386 where possible.

The revision identifier is intended to assist 80376 users to a practical extent. However, the revision identifier value is not guaranteed to change with every stepping revision, or to follow a completely uniform numerical sequence, depending on the type or intention of revision, or manufacturing materials required to be changed. Intel has sole discretion over these characteristics of the component.

Table 4.7. Component and Revision Identifier History

80376 Stepping Name	Revision Identifier
A0	05H

4.7 Coprocessor Interfacing

The 80376 provides an automatic interface for the Intel 80387SX numeric floating-point coprocessor. The 80387SX coprocessor uses an I/O mapped interface driven automatically by the 80376 and assisted by three dedicated signals: BUSY#, ERROR# and PEREQ.

As the 80376 begins supporting a coprocessor instruction, it tests the BUSY# and ERROR# signals to determine if the coprocessor can accept its next instruction. Thus, the BUSY# and ERROR# inputs eliminate the need for any "preamble" bus cycles for communication between processor and coprocessor. The 80387SX can be given its command opcode immediately. The dedicated signals provide instruction synchronization, and eliminate the need of using the 80376 WAIT opcode (9BH) for 80387SX instruction synchronization (the WAIT opcode was required when the 8086 or 8088 was used with the 8087 coprocessor).

Custom coprocessors can be included in 80376 based systems by memory-mapped or I/O-mapped interfaces. Such coprocessor interfaces allow a completely custom protocol, and are not limited to a set of coprocessor protocol "primitives". Instead, memory-mapped or I/O-mapped interfaces may use all applicable 80376 instructions for high-speed coprocessor communication. The BUSY# and ERROR# inputs of the 80376 may also be used for the custom coprocessor interface, if such hardware assist is desired. These signals can be tested by the 80376 WAIT opcode (9BH). The WAIT instruction will wait until the BUSY# input is inactive (interruptable by an NMI or enabled INTR input), but generates an exception 16 fault if the ERROR# pin is active when the BUSY# goes (or is) inactive. If the custom coprocessor interface is memory-mapped, protection of the addresses used for the interface can be provided with the segmentation mechanism of the 80376. If the custom interface is I/O-mapped, protection of the interface can be provided with the 80376 IOPL (I/O Privilege Level) mechanism.

The 80387SX numeric coprocessor interface is I/O mapped as shown in Table 4.8. Note that the 80387SX coprocessor interface addresses are beyond the 0H-0FFFFH range for programmed I/O. When the 80376 supports the 80387SX coprocessor, the 80376 automatically generates bus cycles to the coprocessor interface addresses.

Table 4.8 Numeric Coprocessor Port Addresses

Address in 80376 I/O Space	80387SX Coprocessor Register
8000F8H	Opcode Register
8000FCH	Operand Register
8000FEH	Operand Register

SOFTWARE TESTING FOR COPROCESSOR PRESENCE

When software is used to test coprocessor (80387SX) presence, it should use only the following coprocessor opcodes: FNINIT, FNSTCW and FNSTSW. To use other coprocessor opcodes when a coprocessor is known to be not present, first set EM = 1 in the 80376 CR0 register.

5.0 PACKAGE THERMAL SPECIFICATIONS

The Intel 80376 embedded processor is specified for operation when case temperature is within the range of 0°C–115°C for the ceramic 88-pin PGA package, and 0°C–110°C for the 100-pin plastic package. The case temperature may be measured in any environment, to determine whether the 80376 is within specified operating range. The case temperature should be measured at the center of the top surface.

The ambient temperature is guaranteed as long as T_C is not violated. The ambient temperature can be calculated from the θ_{jc} and θ_{ja} from the following equations:

$$T_J = T_c + P*\theta_{jc}$$

$$T_A = T_j - P*\theta_{ja}$$

$$T_C = T_a + P*[\theta_{ja} - \theta_{jc}]$$

Values for θ_{ja} and θ_{jc} are given in Table 5.1 for the 100-lead fine pitch. θ_{ja} is given at various airflows. Table 5.2 shows the maximum T_a allowable (without exceeding T_c) at various airflows. Note that T_a can be improved further by attaching "fins" or a "heat sink" to the package. P is calculated using the maximum ***hot*** I_{cc}.

Table 5.1. 80376 Package Thermal Characteristics Thermal Resistances (°C/Watt) θ_{jc} and θ_{ja}

Package	θ_{jc}	θ_{ja} Versus Airflow-ft/min (m/sec)					
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
100-Lead Fine Pitch	7	33	27	24	21	18	17
88-Pin PGA	2	25	20	17	14	12	11

Assuming I_{CC} hot of 360 mA, V_{CC} of 5.0V, and a T_{CASE} of 110°C for plastic and 115°C for the 88-Pin PGA Package:

Table 5.2. 80376 Maximum Allowable Ambient Temperature at Various Airflows

Package	θ_{jc}	T_A(°C) vs Airflow-ft/min (m/sec)					
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
100-Lead Fine Pitch	7	63	74	79	85	91	92
88-Pin PGA	2	74	83	88	93	97	99

6.0 ELECTRICAL SPECIFICATIONS

The following sections describe recommended electrical connections for the 80376, and its electrical specifications.

6.1 Power and Grounding

The 80376 is implemented in CHMOS III technology and has modest power requirements. However, its high clock frequency and 47 output buffers (address, data, control, and HLDA) can cause power surges as multiple output buffers drive new signal levels simultaneously. For clean on-chip power distribution at high frequency, 14 V_{CC} and 18 V_{SS} pins separately feed functional units of the 80376.

Power and ground connections must be made to all external V_{CC} and GND pins of the 80376. On the circuit board, all V_{CC} pins should be connected on a V_{CC} plane and all V_{SS} pins should be connected on a GND plane.

POWER DECOUPLING RECOMMENDATIONS

Liberal decoupling capacitors should be placed near the 80376. The 80376 driving its 24-bit address bus and 16-bit data bus at high frequencies can cause transient power surges, particularly when driving large capacitive loads. Low inductance capacitors and interconnects are recommended for best high frequency electrical performance. Inductance can be reduced by shortening circuit board traces between the 80376 and decoupling capacitors as much as possible.

RESISTOR RECOMMENDATIONS

The ERROR# and BUSY# inputs have internal pull-up resistors of approximately 20 KΩ and the PEREQ input has an internal pull-down resistor of approximately 20 KΩ built into the 80376 to keep these signals inactive when the 80387SX is not present in the system (or temporarily removed from its socket).

In typical designs, the external pull-up resistors shown in Table 6.1 are recommended. However, a particular design may have reason to adjust the resistor values recommended here, or alter the use of pull-up resistors in other ways.

Table 6.1. Recommended Resistor Pull-Ups to V_{CC}

Pin	Signal	Pull-Up Value	Purpose
16	ADS#	20 KΩ ± 10%	Lightly Pull ADS# Inactive during 80376 Hold Acknowledge States
26	LOCK#	20 KΩ ± 10%	Lightly Pull LOCK# Inactive during 80376 Hold Acknowledge States

OTHER CONNECTION RECOMMENDATIONS

For reliable operation, always connect unused inputs to an appropriate signal level. N/C pins should always remain **unconnected. Connection of N/C pins to V_{CC} or V_{SS} will result in incompatibility with future steppings of the 80376.**

Particularly when not using interrupts or bus hold (as when first prototyping), prevent any chance of spurious activity by connecting these associated inputs to GND:

—INTR
—NMI
—HOLD

If not using address pipelining connect the NA# pin to a pull-up resistor in the range of 20 KΩ to V_{CC}.

6.2 Absolute Maximum Ratings

Table 6.2. Maximum Ratings

Parameter	Maximum Rating
Storage Temperature	−65°C to +150°C
Case Temperature under Bias	−65°C to +120°C
Supply Voltage with Respect to V_{SS}	−0.5V to +6.5V
Voltage on Other Pins	−0.5V to (V_{CC} + 0.5)V

Table 6.2 gives a stress ratings only, and functional operation at the maximums is not guaranteed. Functional operating conditions are given in **Section 6.3, D.C. Specifications,** and **Section 6.4, A.C. Specifications.**

Extended exposure to the Maximum Ratings may affect device reliability. Furthermore, although the 80376 contains protective circuitry to resist damage from static electric discharge, always take precautions to avoid high static voltages or electric fields.

6.3 D.C. Specifications

ADVANCE INFORMATION SUBJECT TO CHANGE

Table 6.3: 80376 D.C. Characteristics

Functional Operating Range: V_{CC} = 5V ±10%; T_{CASE} = 0°C to 115°C 88-pin PGA, T_{CASE} = 0°C to 110°C 100-pin plastic

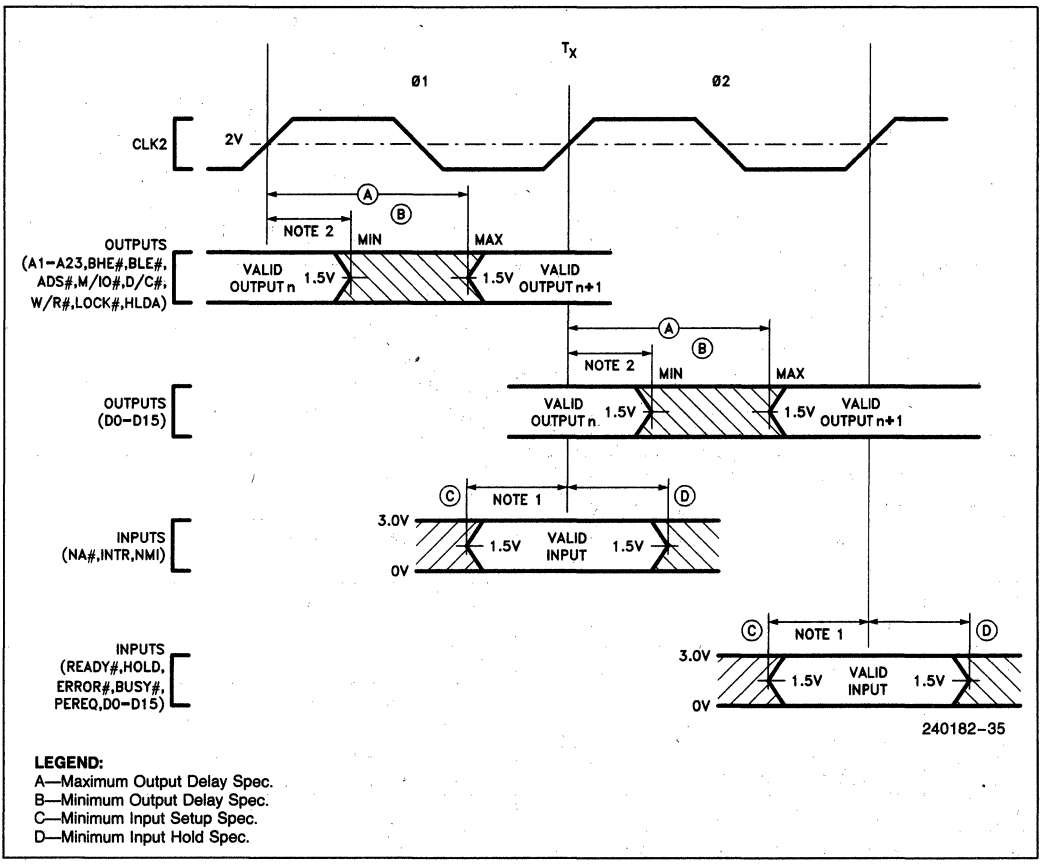
Symbol	Parameter	Min	Max	Unit
V_{IL}	Input LOW Voltage	−0.3	+0.8	V(1)
V_{IH}	Input HIGH Voltage	2.0	V_{CC} +0.3	V(1)
V_{ILC}	CLK2 Input LOW Voltage	−0.3	+0.8	V(1)
V_{IHC}	CLK2 Input HIGH Voltage	V_{CC} − 0.8	V_{CC} + 0.3	V(1)
V_{OL}	Output LOW Voltage			
I_{OL} = 4 mA:	A_{23}–A_1, D_{15}–D_0		0.45	V(1)
I_{OL} = 5 mA:	BHE#, BLE#, W/R#, D/C#, M/IO#, LOCK#, ADS#, HLDA		0.45	V(1)
V_{OH}	Output High Voltage			
I_{OH} = −1 mA:	A_{23}–A_1, D_{15}–D_0	2.4		V(1)
I_{OH} = −0.2 mA:	A_{23}–A_1, D_{15}–D_0	V_{CC} − 0.5		V(1)
I_{OH} = −0.9 mA:	BHE#, BLE#, W/R#, D/C#, M/IO#, LOCK#, ADS#, HLDA	2.4		V(1)
I_{OH} = −0.18 mA:	BHE#, BLE#, W/R#, D/C#, M/IO#, LOCK#, ADS#, HLDA	V_{CC} − 0.5		V(1)
I_{LI} *	Input Leakage Current (For All Pins except PEREQ, BUSY# and ERROR#)		±15	µA, 0V ≤ V_{IN} ≤ V_{CC}(1)
I_{IH}	Input Leakage Current (PEREQ Pin)		200	µA, V_{IH} = 2.4V(1, 2)
I_{IL}	Input Leakage Current (Busy# and ERROR# Pins)		−400	µA, V_{IL} = 0.45V(3)
I_{LO}	Output Leakage Current		±15	µA, 0.45V ≤ V_{OUT} ≤ V_{CC}(1)
I_{CC}	Supply Current at HOT		400 360	mA(4) mA(6)
C_{IN}	Input Capacitance		10	pF, F_C = 1 MHz(5)
C_{OUT}	Output or I/O Capacitance		12	pF, F_C = 1 MHz(5)
C_{CLK}	CLK2 Capacitance		20	pF, F_C = 1 MHz(5)

NOTES:
1. Tested at the minimum operating frequency of the part.
2. PEREQ input has an internal pull-down resistor.
3. BUSY# and ERROR# inputs each have an internal pull-up resistor.
4. I_{CC} max measurement at worse case frequency, V_{CC} and temperature (0°C).
5. Not 100% tested.
6. I_{CC} HOT max measurement at worse case frequency, V_{CC} and max temperature.

The A.C. specifications given in Table 6.4 consist of output delays, input setup requirements and input hold requirements. All A.C. specifications are relative to the CLK2 rising edge crossing the 2.0V level.

A.C. specification measurement is defined by Figure 6.1. Inputs must be driven to the voltage levels indicated by Figure 6.1 when A.C. specifications are measured. 80376 output delays are specified with minimum and maximum limits measured as shown. The minimum 80376 delay times are hold times provided to external circuitry. 80376 input setup and hold times are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct 80376 processor operation.

Outputs NA#, W/R#, D/C#, M/IO#, LOCK#, BHE#, BLE#, A_{23}–A_1 and HLDA only change at the beginning of phase one. D_{15}–D_0 (write cycles) only change at the beginning of phase two. The READY#, HOLD, BUSY#, ERROR#, PEREQ and D_{15}–D_0 (read cycles) inputs are sampled at the beginning of phase one. The NA#, INTR and NMI inputs are sampled at the beginning of phase two.



LEGEND:
A—Maximum Output Delay Spec.
B—Minimum Output Delay Spec.
C—Minimum Input Setup Spec.
D—Minimum Input Hold Spec.

Figure 6.1. Drive Levels and Measurement Points for A.C. Specifications

6.4 A.C. Specifications

ADVANCE INFORMATION SUBJECT TO CHANGE

Table 6.4. 80376 A.C. Characteristics at 16 MHz

Functional Operating Range: V_{CC} = 5V ±10%; T_{CASE} = 0°C to 115°C for 88-pin PGA, 0°C to 110°C for 100-pin plastic

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Operating Frequency	4	16	MHz		Half CLK2 Freq
t_1	CLK2 Period	31	125	ns	6.3	*
t_{2a}	CLK2 HIGH Time	9		ns	6.3	At 2(3)
t_{2b}	CLK2 HIGH Time	5		ns	6.3	At $(V_{CC}-0.8)$V(3)
t_{3a}	CLK2 LOW Time	9		ns	6.3	At 2V(3)
t_{3b}	CLK2 LOW Time	7		ns	6.3	At 0.8V(3)
t_4	CLK2 Fall Time		8	ns	6.3	$(V_{CC}-0.8)$V to 0.8V(3)
t_5	CLK2 Rise Time		8	ns	6.3	0.8V to $(V_{CC}-0.8)$(3)
t_6	$A_{23}-A_1$ Valid Delay	4	36	ns	6.5	C_L = 120 pF(4)
t_7	$A_{23}-A_1$ Float Delay	4	40	ns	6.6	(1)
t_8	BHE#, BLE#, LOCK# Valid Delay	4	36	ns	6.5	C_L = 75 pF(4)
t_9	BHE#, BLE#, LOCK# Float Delay	4	40	ns	6.6	(1)
t_{10}	W/R#, M/IO#, D/C#, ADS# Valid Delay	6	33	ns	6.5	C_L = 75 pF(4)
t_{11}	W/R#, M/IO#, D/C#, ADS# Float Delay	6	35	ns	6.6	(1)
t_{12}	$D_{15}-D_0$ Write Data Valid Delay	4	40	ns	6.5	C_L = 120 pF(4)
t_{13}	$D_{15}-D_0$ Write Data Float Delay	4	35	ns	6.6	(1)
t_{14}	HLDA Valid Delay	6	33	ns	6.6	C_L = 75 pF(4)
t_{15}	NA# Setup Time	5		ns	6.4	
t_{16}	NA# Hold Time	21		ns	6.6	
t_{19}	READY# Setup Time	19		ns	6.4	
t_{20}	READY# Hold Time	4		ns	6.4	
t_{21}	Setup Time $D_{15}-D_0$ Read Data	9		ns	6.4	
t_{22}	Hold Time $D_{15}-D_0$ Read Data	6		ns	6.4	
t_{23}	HOLD Setup Time	26		ns	6.4	
t_{24}	HOLD Hold Time	5		ns	6.4	
t_{25}	RESET Setup Time	13		ns	6.7	
t_{26}	RESET Hold Time	4		ns	6.7	

NOTE:
The 80376 does not have t_{17} or t_{18} timing specifications.

Table 6.4. 80376 A.C. Characteristics at 16 MHz

Functional Operating Range: V_{CC} = 5V ±10%; T_{CASE} = 0°C to 115°C for 80-pin PGA, 0°C to 110°C for 100-pin plastic (Continued)

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{27}	NMI, INTR Setup Time	16		ns	6.4	(2)
t_{28}	NMI, INTR Hold Time	16		ns	6.4	(2)
t_{29}	PEREQ, ERROR#, BUSY# Setup Time	16		ns	6.4	(2)
t_{30}	PEREQ, ERROR#, BUSY# Hold Time	5		ns	6.4	(2)

NOTES:
1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3. These are not tested. They are guaranteed by design characterization.
4. Tested with C_L set to 50 pF and derated to support the indicated distributed capacitive load. See Figure 6.8 for the capacitive derating curve.

A.C. TEST LOADS

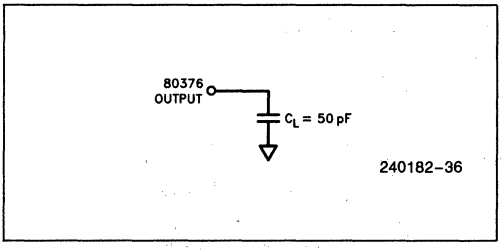


Figure 6.2. A.C. Test Loads

A.C. TIMING WAVEFORMS

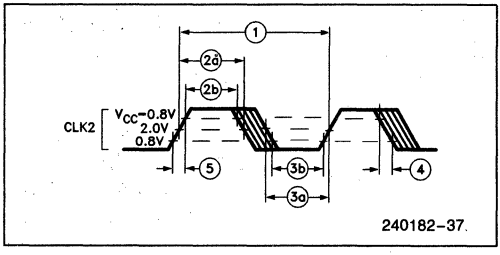


Figure 6.3. CLK2 Waveform

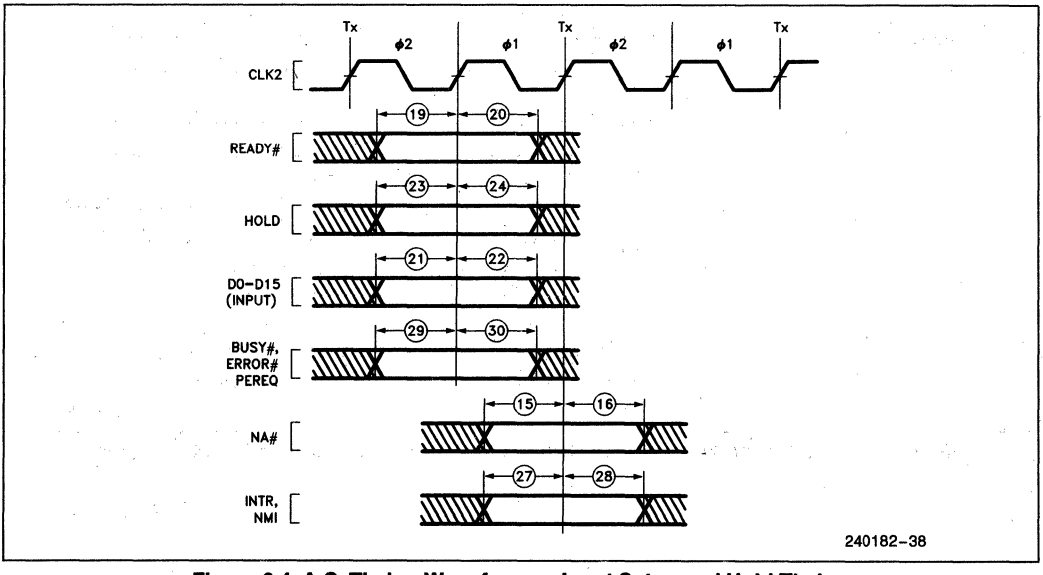


Figure 6.4. A.C. Timing Waveforms—Input Setup and Hold Timing

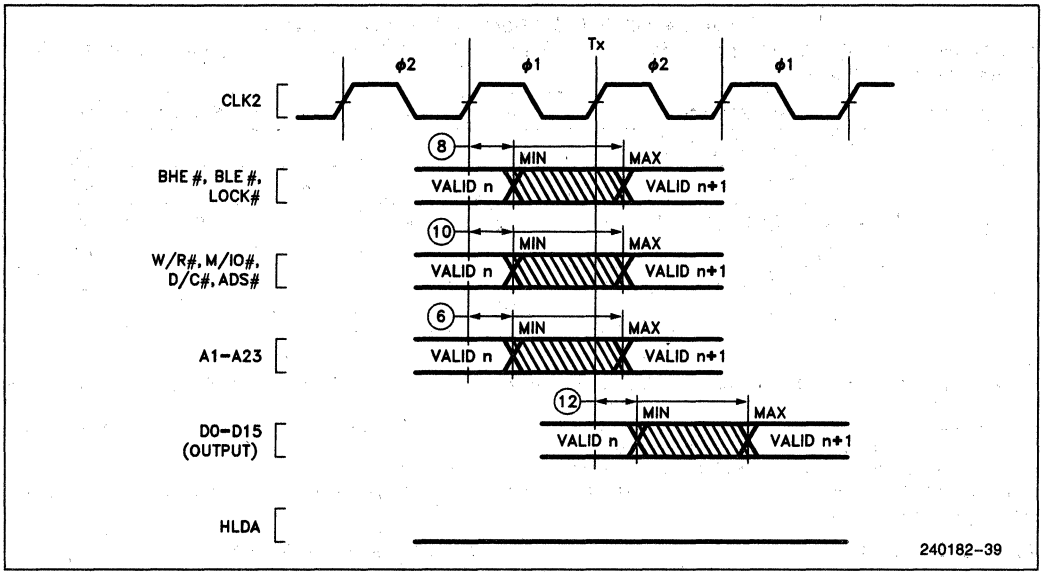


Figure 6.5. A.C. Timing Waveforms—Output Valid Delay Timing

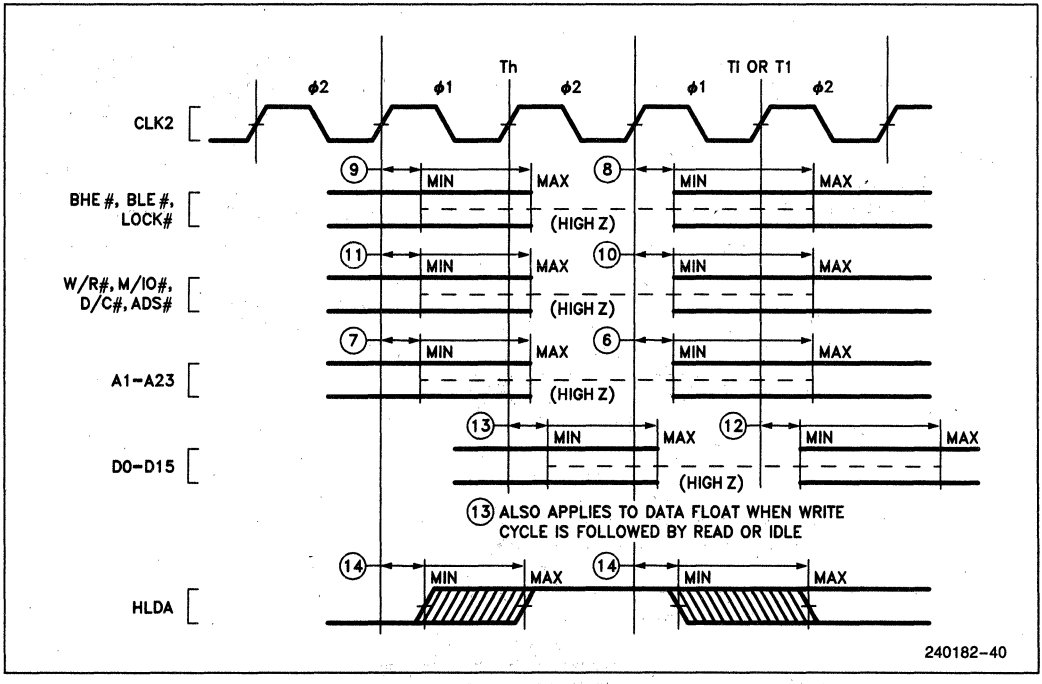
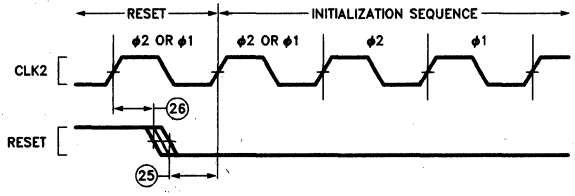
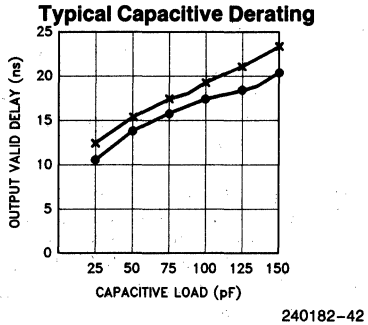


Figure 6.6. A.C. Timing Waveforms—Output Float Delay and HLDA Valid Delay Timing



The second internal processor phase following RESET high-to-low transition (provided t_{25} and t_{26} are met) is Φ2.

Figure 6.7. A.C. Timing Waveforms—RESET Setup and Hold Timing, and Internal Phase



•-Low to High
X-High to Low

Figure 6.8. Capacitive Derating Curve

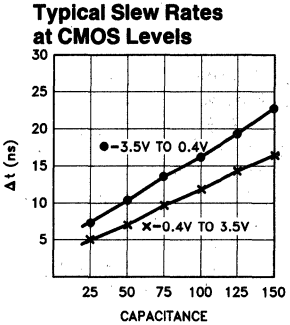
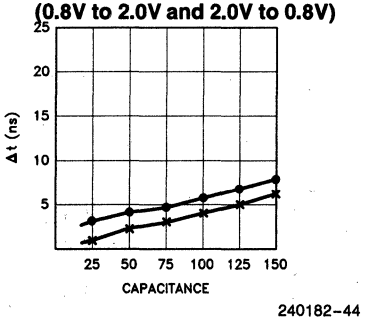


Figure 6.9. CMOS Level Slew Rates for Output Buffers



•-Low to High
X-High to Low

Figure 6.10. TTL Level Slew Rates for Output Buffers

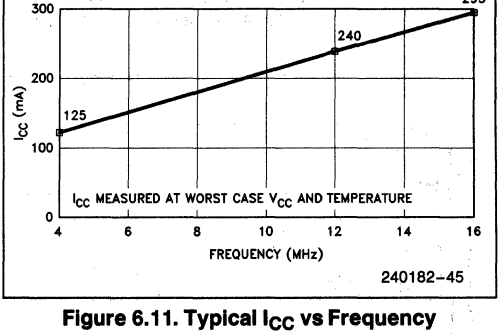


Figure 6.11. Typical I_{CC} vs Frequency

6.5 Designing for ICE™-376 Emulator (Advanced Data)

The 376 embedded processor in-circuit emulator product is the ICE-376 emulator. Use of the emulator requires the target system to provide a socket that is compatible with the ICE-376 emulator. The 80376 offers two different probes for emulating user systems: an 88-pin PGA probe and a 100-pin fine pitch flat-pack probe. The 100-pin fine pitch flat-pack probe requires a socket, called the 100-pin PQFP, which is available from 3-M text-tool (part number 2-0100-07243-000). The ICE-376 emulator probe attaches to the target system via an adapter which replaces the 80376 component in the target system. Because of the high operating frequency of 80376 systems and of the ICE-376 emulator, there is no buffering between the 80376 emulation processor in the ICE-376 emulator probe and the target system. A direct result of the non-buffered interconnect is that the ICE-376 emulator shares the address and data bus with the user's system, and the RESET signal is intercepted by the ICE emulator hardware. In order for the ICE-376 emulator to be functional in the user's system without the Optional Isolation Board (OIB) the designer must be aware of the following conditions:

1. The bus controller must only enable data transceivers onto the data bus during valid read cycles of the 80376, other local devices or other bus masters.
2. Before another bus master drives the local processor address bus, the other master must gain control of the address bus by asserting HOLD and receiving the HLDA response.
3. The emulation processor receives the RESET signal 2 or 4 CLK2 cycles later than an 80376 would, and responds to RESET later. Correct phase of the response is guaranteed.

In addition to the above considerations, the ICE-376 emulator processor module has several electrical and mechanical characteristics that should be taken into consideration when designing the 80376 system.

Capacitive Loading: ICE-376 adds up to 27 pF to each 80376 signal.

Drive Requirements: ICE-376 adds one FAST TTL load on the CLK2, control, address, and data lines. These loads are within the processor module and are driven by the 80376 emulation processor, which has standard drive and loading capability listed in Tables 6.3 and 6.4.

Power Requirements: For noise immunity and CMOS latch-up protection the ICE-376 emulator processor module is powered by the user system. The circuitry on the processor module draws up to 1.4A including the maximum 80376 I_{CC} from the user 80376 socket.

80376 Location and Orientation: The ICE-376 emulator processor module may require lateral clearance. Figure 6.12 shows the clearance requirements of the iMP adapter and Figure 6.13 shows the clearance requirements of the 88-pin PGA adapter. The

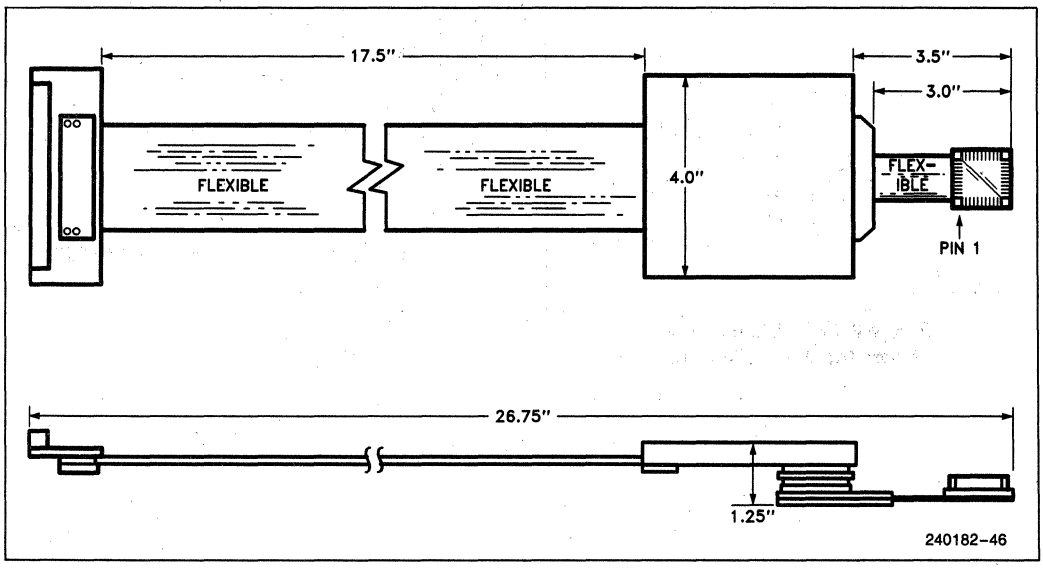


Figure 6.12. Preliminary ICE™-376 Emulator User Cable with PQFP Adapter

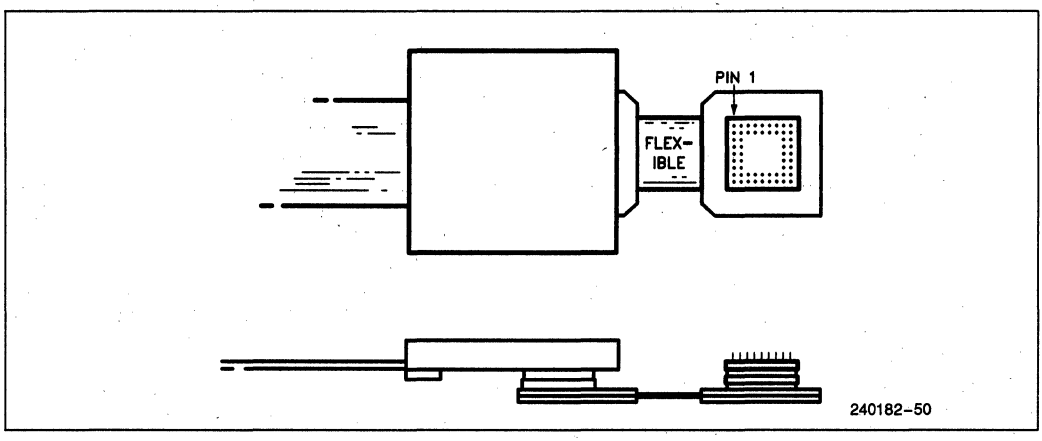


Figure 6.13. Preliminary ICE™-376 Emulator User Cable with 88-Pin PGA Adapter

optional isolation board (OIB), which provides extra electrical buffering and has the same lateral clearance requirements as Figures 6.12 and 6.13, adds an additional 0.5 inches to the vertical clearance requirement. This is illustrated in Figure 6.14.

Optional Isolation Board (OIB) and the CLK2 speed reduction: Due to the unbuffered probe design, the ICE-376 emulator is susceptible to errors on the user's bus. The OIB allows the ICE-376 emulator to function in user systems with faults (shorted signals, etc.). After electrical verification the OIB may be removed. When the OIB is installed, the user system must have a maximum CLK2 frequency of 20 MHz.

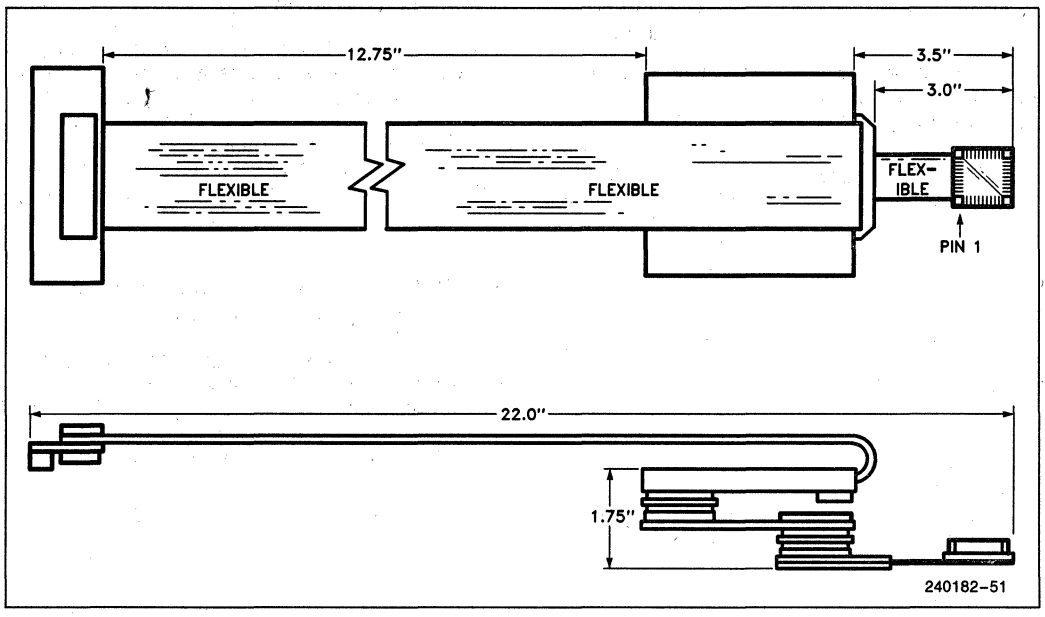


Figure 6.14. Preliminary ICE™-376 Emulator User Cable with OIB and PQFP Adapter

7.0 DIFFERENCES BETWEEN THE 80376 AND THE 80386

The following are the major differences between the 80376 and the 80386.

1. The 80376 generates byte selects on BHE# and BLE# (like the 8086 and 80286 microprocessors) to distinguish the upper and lower bytes on its 16-bit data bus. The 80386 uses four-byte selects, BE0#–BE3#, to distinguish between the different bytes on its 32-bit bus.
2. The 80376 has no bus sizing option. The 80386 can select between either a 32-bit bus or a 16-bit bus by use of the BS16# input. The 80376 has a 16-bit bus size.
3. The NA# pin operation in the 80376 is identical to that of the NA# pin on the 80386 with one exception: the NA# pin of the 80386 cannot be activated on 16-bit bus cycles (where BS16# is LOW in the 80386 case), whereas NA# can be activated on any 80376 bus cycle.
4. The contents of all 80376 registers at reset are identical to the contents of the 80386 registers at reset, except the DX register. The DX register contains a component-stepping identifier at reset, i.e.

   in 80386, after reset DH = 3 indicates 80386
   DL = revision number;
   in 80376, after reset DH = 33H indicates 80376
   DL = revision number.

5. The 80386 uses A_{31} and M/IO# as a select for numerics coprocessor. The 80376 uses the A_{23} and M/IO# to select its numerics coprocessor.
6. The 80386 prefetch unit fetches code in four-byte units. The 80376 prefetch unit reads two bytes as one unit (like the 80286 microprocessor). In BS16# mode, the 80386 takes two consecutive bus cycles to complete a prefetch request. If there is a data read or write request after the prefetch starts, the 80386 will fetch all four bytes before addressing the new request.
7. The 80376 has no paging mechanism.
8. The 80376 starts executing code in what corresponds to the 80386 protected mode. The 80386 starts execution in real mode, which is then used to enter protected mode.
9. The 80386 has a virtual-86 mode that allows the execution of a real mode 8086 program as a task in protected mode. The 80376 has no virtual-86 mode.
10. The 80386 maps a 48-bit logical address into a 32-bit physical address by segmentation and paging. The 80376 maps its 48-bit logical address into a 24-bit physical address by segmentation only.
11. The 80376 uses the 80387SX numerics coprocessor for floating point operations, while the 80386 uses the 80387 coprocessor.
12. The 80386 can execute from 16-bit code segments. The 80376 can **only** execute from 32-bit code Segments.

8.0 INSTRUCTION SET

This section describes the 376 embedded processor instruction set. Table 8.1 lists all instructions along with instruction encoding diagrams and clock counts. Further details of the instruction encoding are then provided in the following sections, which completely describe the encoding structure and the definition of all fields occurring within 80376 instructions.

8.1 80376 Instruction Encoding and Clock Count Summary

To calculate elapsed time for an instruction, multiply the instruction clock count, as listed in Table 8.1 below, by the processor clock period (e.g. 62.5 ns for an 80376 operating at 16 MHz). The actual clock count of an 80376 program will average 10% more

than the calculated clock count due to instruction sequences which execute faster than they can be fetched from memory.

Instruction Clock Count Assumptions:

1. The instruction has been prefetched, decoded, and is ready for execution.
2. Bus cycles do not require wait states.
3. There are no local bus HOLD requests delaying processor acess to the bus.
4. No exceptions are detected during instruction execution.
5. If an effective address is calculated, it does not use two general register components. One register, scaling and displacement can be used within the clock counts showns. However, if the effective address calculation uses two general register components, add 1 clock to the clock count shown.
6. Memory reference instruction accesses byte or aligned 16-bit operands.

Instruction Clock Count Notation

— If two clock counts are given, the smaller refers to a register operand and the larger refers to a memory operand.

—n = number of times repeated.

—m = number of components in the next instruction executed, where the entire displacement (if any) counts as one component, the entire immediate data (if any) counts as one component, and all other bytes of the instruction and prefix(es) each count as one component.

Misaligned or 32-Bit Operand Accesses:

— If instructions accesses a misaligned 16-bit operand or 32-bit operand on even address add:

2* clocks for read or write.

4** clocks for read and write.

— If instructions accesses a 32-bit operand on odd address add:

4* clocks for read or write.

8** clocks for read and write.

Wait States:

Wait states add 1 clock per wait state to instruction execution for each data access.

Table 8.1. 80376 Instruction Set Clock Count Summary

Instruction	Format	Clock Counts	Number of Data Cycles	Notes
GENERAL DATA TRANSFER				
MOV = Move:				
Register to Register/Memory	1000100w mod reg r/m	2/2*	0/1*	a
Register/Memory to Register	1000101w mod reg r/m	2/4*	0/1*	a
Immediate to Register/Memory	1100011w mod000 r/m immediate data	2/2*	0/1*	a
Immediate to Register (Short Form)	1011 w reg immediate data	2	2	
Memory to Accumulator (Short Form)	1010000w full displacement	4*	1*	a
Accumulator to Memory (Short Form)	1010001w full displacement	2*	1*	a
Register/Memory to Segment Register	10001110 mod sreg3 r/m	22/23	0/6*	a,b,c
Segment Register to Register/Memory	10001100 mod sreg3 r/m	2/2*	0/1*	a
MOVSX = Move with Sign Extension				
Register from Register/Memory	00001111 1011111w mod reg r/m	3/6*	0/1*	a
MOVZX = Move with Zero Extension				
Register from Register/Memory	00001111 1011011w mod reg r/m	3/6*	0/1*	a
PUSH = Push:				
Register/Memory	11111111 mod110 r/m	7/9*	2/4*	a
Register (Short Form)	01010 reg	4	2	a
Segment Register (ES, CS, SS or DS)	000 sreg2 110	4	2	a
Segment Register (FS or GS)	00001111 10 sreg3 000	4	2	a
Immediate	01101s0 immediate data	4	2	a
PUSHA = Push All	01100000	34	16	a
POP = Pop				
Register/Memory	10001111 mod000 r/m	7/9*	2/4*	a
Register (Short Form)	01011 reg	6	2	a
Segment Register (ES, SS or DS)	000 sreg 2 111	25	6	a, b, c
Segment Register (FS or GS)	00001111 10 sreg 3 001	25	6	a, b, c
POPA = Pop All	01100001	40	16	a
XCHG = Exchange				
Register/Memory with Register	1000011w mod reg r/m	3/5**	0/2**	a, m
Register with Accumulator (Short Form)	10010 reg	3	0	
IN = Input from:				
Fixed Port	1110010w port number	6*	1*	f,k
		26*	1*	f,l
Variable Port	1110110w	7*	1*	f,k
		27*	1*	f,l
OUT = Output to:				
Fixed Port	1110011w port number	4*	1*	f,k
		24*	1*	f,l
Variable Port	1110111w	5*	1*	f,k
		26*	1*	f,l
LEA = Load EA to Register	10001101 mod reg r/m	2		

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number of Data Cycles	Notes
SEGMENT CONTROL				
LDS = Load Pointer to DS	11000101 | mod reg r/m	26*	6*	a, b, c
LES = Load Pointer to ES	11000100 | mod reg r/m	26*	6*	a, b, c
LFS = Load Pointer to FS	00001111 | 10110100 | mod reg r/m	29*	6*	a, b, c
LGS = Load Pointer to GS	00001111 | 10110101 | mod reg r/m	29*	6*	a, b, c
LSS = Load Pointer to SS	00001111 | 10110010 | mod reg r/m	26*	6*	a, b, c
FLAG CONTROL				
CLC = Clear Carry Flag	11111000	2		
CLD = Clear Direction Flag	11111100	2		
CLI = Clear Interrupt Enable Flag	11111010	8		f
CLTS = Clear Task Switched Flag	00001111 | 00000110	5		e
CMC = Complement Carry Flag	11110101	2		
LAHF = Load AH into Flag	10011111	2		
POPF = Pop Flags	10011101	5		a, g
PUSHF = Push Flags	10011100	4		a
SAHF = Store AH into Flags	10011110	3		
STC = Set Carry Flag	11111001	2		
STD = Set Direction Flag	11111101	2		
STI = Set Interrupt Enable Flag	11111011	8		f
ARITHMETIC				
ADD = Add				
Register to Register	000000dw | mod reg r/m	2		
Register to Memory	0000000w | mod reg r/m	7**	2**	a
Memory to Register	0000001w | mod reg r/m	6*	1*	a
Immediate to Register/Memory	100000sw | mod000 r/m | immediate data	2/7**	0/2**	a
Immediate to Accumulator (Short Form)	0000010w | immediate data	2		
ADC = Add with Carry				
Register to Register	000100dw | mod reg r/m	2		
Register to Memory	0001000w | mod reg r/m	7**	2**	a
Memory to Register	0001001w | mod reg r/m	6*	1*	a
Immediate to Register/Memory	100000sw | mod010 r/m | immediate data	2/7**	0/2**	a
Immediate to Accumulator (Short Form)	0001010w | immediate data	2		
INC = Increment				
Register/Memory	1111111w | mod000 r/m	2/6**	0/2**	a
Register (Short Form)	01000 reg	2		
SUB = Subtract				
Register from Register	001010dw | mod reg r/m	2		

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number Of Data Cycles	Notes
ARITHMETIC (Continued)				
Register from Memory	0010100w mod reg r/m	7**	2**	a
Memory from Register	0010101w mod reg r/m	6*	1	a
Immediate from Register/Memory	100000sw mod 101 r/m immediate data	2/7**	0/1**	a
Immediate from Accumulator (Short Form)	0010110w immediate data			
SBB = Subtract with Borrow				
Register from Register	000110dw mod reg r/m	2		
Register from Memory	0001100w mod reg r/m	7*	2**	a
Memory from Register	0001101w mod reg r/m	6	1*	a
Immediate from Register/Memory	100000sw mod 011 r/m immediate data	2/7**	0/2**	a
Immediate from Accumulator (Short Form)	0001110w immediate data	2		
DEC = Decrement				
Register/Memory	1111111w reg 001 r/m	2/6**	0/2**	a
Register (Short Form)	01001 reg	2		
CMP = Compare				
Register with Register	0011100w mod reg r/m	2		
Memory with Register	0011100w mod reg r/m	5*	1*	a
Register with Memory	0011101w mod reg r/m	6**	2**	a
Immediate with Register/Memory	100000sw mod 111 r/m immediate data	2/5*	0/1*	a
Immediate with Accumulator (Short Form)	0011110w immediate data	2		
NEG = Change Sign	1111011w mod 011 r/m	2/6*	0/2*	a
AAA = ASCII Adjust for Add	00110111	4		
AAS = ASCII Adjust for Subtract	00111111	4		
DAA = Decimal Adjust for Add	00100111	4		
DAS = Decimal Adjust for Subtract	00101111	4		
MUL = Multiply (Unsigned)				
Accumulator with Register/Memory	1111011w mod 100 r/m			
Multiplier—Byte		12–17/15–20	0/1	a,n
—Word		12–25/15–28*	0/1*	a,n
—Doubleword		12–41/17–46*	0/2*	a,n
IMUL = Integer Multiply (Signed)				
Accumulator with Register/Memory	1111011w mod 101 r/m			
Multiplier—Byte		12–17/15–20	0/1	a,n
—Word		12–25/15–28*	0/1*	a,n
—Doubleword		12–41/17–46*	0/2*	a,n
Register with Register/Memory	00001111 10101111 mod reg r/m			
Multiplier—Byte		12–17/15–20	0/1	a,n
—Word		12–25/15–28*	0/1*	a,n
—Doubleword		12–41/17–46*	0/2*	a,n
Register/Memory with Immediate to Register	011010s1 mod reg r/m immediate data			
—Word		13–26/14–27*	0/1*	a,n
—Doubleword		13–42/16–45*	0/2*	a,n

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number Of Data Cycles	Notes
ARITHMETIC (Continued)				
DIV = Divide (Unsigned)				
Accumulator by Register/Memory	1111011w mod110 r/m			
Divisor—Byte		14/17	0/1	a, o
—Word		22/25*	0/1*	a, o
—Doubleword		38/43*	0/2*	a, o
IDIV = Integer Divide (Signed)				
Accumulator by Register/Memory	1111011w mod111 r/m			
Divisor—Byte		19/22	0/1	a, o
—Word		27/30*	0/1	a, o
—Doubleword		43/48*	0/2*	a, o
AAD = ASCII Adjust for Divide	11010101 00001010	19		
AAM = ASCII Adjust for Multiply	11010100 00001010	17		
CBW = Convert Byte to Word	10011000	3		
CWD = Convert Word to Double Word	10011001	2		
LOGIC				
Shift Rotate Instructions				
Not Through Carry (**ROL, ROR, SAL, SAR, SHL,** and **SHR**)				
Register/Memory by 1	1101000w mod TTT r/m	3/7**	0/2**	a
Register/Memory by CL	1101001w mod TTT r/m	3/7**	0/2**	a
Register/Memory by Immediate Count	1100000w mod TTT r/m immed 8-bit data	3/7**	0/2**	a
Through Carry (**RCL** and **RCR**)				
Register/Memory by 1	1101000w mod TTT r/m	9/10**	0/2**	a
Register/Memory by CL	1101001w mod TTT r/m	9/10**	10/2**	a
Register/Memory by Immediate Count	1100000w mod TTT r/m immed 8-bit data	9/10**	0/2**	a

TTT	Instruction
000	ROL
001	ROR
010	RCL
011	RCR
100	SHL/SAL
101	SHR
111	SAR

SHLD = Shift Left Double						
Register/Memory by Immediate	00001111	10100100	mod reg	r/m	immed 8-bit data	3/7**	0/2**	
Register/Memory by CL	00001111	10100101	mod reg	r/m		3/7**	0/2**	
SHRD = Shift Right Double								
Register/Memory by Immediate	00001111	10101100	mod reg	r/m	immed 8-bit data	3/7**	0/2**	
Register/Memory by CL	00001111	10101101	mod reg	r/m		3/7**	0/2**	
AND = And								
Register to Register	001000dw	mod reg	r/m			2		

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number of Data Cycles	Notes
LOGIC (Continued)				
Register to Memory	0010000w mod reg r/m	7*	2**	a
Memory to Register	0010001w mod reg r/m	6*	1*	a
Immediate to Register/Memory	1000000w mod 100 r/m immediate data	2/7**	0/2**	a
Immediate to Accumulator (Short Form)	0010010w immediate data	2		
TEST = And Function to Flags, No Result				
Register/Memory and Register	1000010w mod reg r/m	2/5*	0/1*	a
Immediate Data and Register/Memory	1111011w mod 000 r/m immediate data	2/5*	0/1*	a
Immediate Data and Accumulator (Short Form)	1010100w immediate data	2		
OR = Or				
Register to Register	000010dw mod reg r/m	2		
Register to Memory	0000100w mod reg r/m	7**	2**	a
Memory to Register	0000101w mod reg r/m	6*	1*	a
Immediate to Register/Memory	1000000w mod 001 r/m immediate data	2/7**	0/2**	a
Immediate to Accumulator (Short Form)	0000110w immediate data	2		
XOR = Exclusive Or				
Register to Register	001100dw mod reg r/m	2		
Register to Memory	0011000w mod reg r/m	7**	2**	a
Memory to Register	0011001w mod reg r/m	6*	1*	a
Immediate to Register/Memory	1000000w mod 110 r/m immediate data	2/7**	0/2**	a
Immediate to Accumulator (Short Form)	0011010w immediate data	2		
NOT = Invert Register/Memory	1111011w mod 010 r/m	2/6**	0/2**	a
STRING MANIPULATION				
CMPS = Compare Byte Word	1010011w	10*	2*	a
INS = Input Byte/Word from DX Port	0110110w	9** 29**	1** 1**	a,f,k a,f,l
LODS = Load Byte/Word to AL/AX/EAX	1010110w	5*	1*	a
MOVS = Move Byte Word	1010010w	7**	2**	a
OUTS = Output Byte/Word to DX Port	0110111w	8** 28**	1** 1**	a,f,k a,f,l
SCAS = Scan Byte Word	1010111w	7*	1*	a
STOS = Store Byte/Word from AL/AX/EX	1010101w	4*	1*	a
XLAT = Translate String	11010111	5*	1*	a
REPEATED STRING MANIPULATION Repeated by Count in CX or ECX				
REPE CMPS = Compare String (Find Non-Match)	11110011 1010011w	5 + 9n**	2n**	a

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number of Data Cycles	Notes
REPEATED STRING MANIPULATION (Continued)				
REPNE CMPS = Compare String				
(Find Match)	11110010 1010011w	5 + 9n**	2n**	a
REP INS = Input String	11110011 0110110w	7 + 6n*	1n*	a,f,k
		27 + 6n*	1n*	a,f,l
REP LODS = Load String	11110011 1010110w	5 + 6n*	1n*	a
REP MOVS = Move String	11110011 1010010w	7 + 4n**	2n**	a
REP OUTS = Output String	11110011 0110111w	6 + 5n*	1n*	a,f,k
		26 + 5n*	1n*	a,f,l
REPE SCAS = Scan String				
(Find Non-AL/AX/EAX)	11110011 1010111w	5 + 8n*	1n*	a
REPNE SCAS = Scan String				
(Find AL/AX/EAX)	11110010 1010111w	5 + 8n*	1n*	a
REP STOS = Store String	11110011 1010101w	5 + 5n*	1n*	a
BIT MANIPULATION				
BSF = Scan Bit Forward	00001111 10111100 mod reg r/m	10 + 3n**	2n**	a
BSR = Scan Bit Reverse	00001111 10111101 mod reg r/m	10 + 3n**	2n**	a
BT = Test Bit				
Register/Memory, Immediate	00001111 10111010 mod 100 r/m immed 8-bit data	3/6*	0/1*	a
Register/Memory, Register	00001111 10100011 mod reg r/m	3/12*	0/1*	a
BTC = Test Bit and Complement				
Register/Memory, Immediate	00001111 10111010 mod 111 r/m immed 8-bit data	6/8*	0/2*	a
Register/Memory, Register	00001111 10111011 mod reg r/m	6/13*	0/2*	a
BTR = Test Bit and Reset				
Register/Memory, Immediate	00001111 10111010 mod 110 r/m immed 8-bit data	6/8*	0/2*	a
Register/Memory, Register	00001111 10110011 mod reg r/m	6/13*	0/2*	a
BTS = Test Bit and Set				
Register/Memory, Immediate	00001111 10111010 mod 101 r/m immed 8-bit data	6/8*	0/2*	a
Register/Memory, Register	00001111 10101011 mod reg r/m	6/13*	0/2*	a
CONTROL TRANSFER				
CALL = Call				
Direct within Segment	11101000 full displacement	9 + m*	2	j
Register/Memory				
Indirect within Segment	11111111 mod 010 r/m	9 + m/12 + m	2/3	a, j
Direct Intersegment	10011010 unsigned full offset, selector	42 + m	9	c, d, j

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number of Data Cycles	Notes
CONTROL TRANSFER (Continued) (Direct Intersegment)				
Via Call Gate to Same Privilege Level		64 + m	13	a,c,d,j
Via Call Gate to Different Privilege Level, (No Parameters)		86 + m	13	a,c,d,j
Via Call Gate to Different Privilege Level, (x Parameters)		106 + 8x + m	13 + 4x	a,c,d,j
From 386 Task to 386 TSS		392	124	a,c,d,j
Indirect Intersegment	11111111 | mod 011 r/m	46 + m	10	a,c,d,j
Via Call Gate to Same Privilege Level		68 + m	14	a,c,d,j
Via Call Gate to Different Privilege Level, (No Parameters)		102 + m	14	a,c,d,j
Via Call Gate to Different Privilege Level, (x Parameters)		110 + 8x + m	14 + 4x	a,c,d,j
From 386 Task to 386 TSS		399	130	a,c,d,j
JMP = Unconditional Jump				
Short	11101011 | 8-bit displacement	7 + m		j
Direct within Segment	11101001 | full displacement	7 + m		j
Register/Memory Indirect within Segment	11111111 | mod 100 r/m	9 + m/14 + m	2/4	a,j
Direct Intersegment	11101010 | unsigned full offset, selector	37 + m	5	c,d,j
Via Call Gate to Same Privilege Level		53 + m	9	a,c,d,j
From 386 Task to 386 TSS		395	124	a,c,d,j
Indirect Intersegment	11111111 | mod 101 r/m	37 + m	9	a,c,d,j
Via Call Gate to Same Privilege Level		59 + m	13	a,c,d,j
From 386 Task to 386 TSS		401	124	a,c,d,j

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format			Clock Counts	Number of Data Cycles	Notes
CONTROL TRANSFER (Continued)						
RET = Return from CALL:						
Within Segment	11000011			12 + m	2	a,j,p
Within Segment Adding Immediate to SP	11000010	16-bit displ		12 + m	2	a,j,p
Intersegment	11001011			[illegible]	4	a,c,d,j,p
Intersegment Adding Immediate to SP	11001010	16-bit displ		[illegible] + m	4	a,c,d,j,p
to Different Privilege Level						
Intersegment				[illegible]	4	c,d,j,p
Intersegment Adding Immediate to SP				[illegible]	4	c,d,j,p
CONDITIONAL JUMPS						
NOTE: Times Are Jump "Taken or Not Taken"						
JO = Jump on Overflow						
8-Bit Displacement	01110000	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10000000	full displacement	7 + m or 3		j
JNO = Jump on Not Overflow						
8-Bit Displacement	01110001	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10000001	full displacement	7 + m or 3		j
JB/JNAE = Jump on Below/Not Above or Equal						
8-Bit Displacement	01110010	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10000010	full displacement	7 + m or 3		j
JNB/JAE = Jump on Not Below/Above or Equal						
8-Bit Displacement	01110011	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10000011	full displacement	7 + m or 3		j
JE/JZ = Jump on Equal/Zero						
8-Bit Displacement	01110100	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10000100	full displacement	7 + m or 3		j
JNE/JNZ = Jump on Not Equal/Not Zero						
8-Bit Displacement	01110101	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10000101	full displacement	7 + m or 3		j
JBE/JNA = Jump on Below or Equal/Not Above						
8-Bit Displacement	01110110	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10000110	full displacement	7 + m or 3		j
JNBE/JA = Jump on Not Below or Equal/Above						
8-Bit Displacement	01110111	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10000111	full displacement	7 + m or 3		j
JS = Jump on Sign						
8-Bit Displacement	01111000	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10001000	full displacement	7 + m or 3		j

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format			Clock Counts	Number of Data Cycles	Notes
CONDITIONAL JUMPS (Continued)						
JNS = Jump on Not Sign						
8-Bit Displacement	01111001	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10001001	full displacement	7 + m or 3		j
JP/JPE = Jump on Parity/Parity Even						
8-Bit Displacement	01111010	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10001010	full displacement	7 + m or 3		j
JNP/JPO = Jump on Not Parity/Parity Odd						
8-Bit Displacement	01111011	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10001011	full displacement	7 + m or 3		j
JL/JNGE = Jump on Less/Not Greater or Equal						
8-Bit Displacement	01111100	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10001100	full displacement	7 + m or 3		j
JNL/JGE = Jump on Not Less/Greater or Equal						
8-Bit Displacement	01111101	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10001101	full displacement	7 + m or 3		j
JLE/JNG = Jump on Less or Equal/Not Greater						
8-Bit Displacement	01111110	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10001110	full displacement	7 + m or 3		j
JNLE/JG = Jump on Not Less or Equal/Greater						
8-Bit Displacement	01111111	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10001111	full displacement	7 + m or 3		j
JCXZ = Jump on CX Zero	11100011	8-bit displ		9 + m or 5		j
JECXZ = Jump on ECX Zero	11100011	8-bit displ		9 + m or 5		j
(Address Size Prefix Differentiates JCXZ from JECXZ)						
LOOP = Loop CX Times	11100010	8-bit displ		11 + m		j
LOOPZ/LOOPE = Loop with Zero/Equal	11100001	8-bit displ		11 + m		j
LOOPNZ/LOOPNE = Loop While Not Zero	11100000	8-bit displ		11 + m		j
CONDITIONAL BYTE SET						
NOTE: Times Are Register/Memory						
SETO = Set Byte on Overflow						
To Register/Memory	00001111	10010000	mod 000 r/m	4/5*	0/1*	a
SETNO = Set Byte on Not Overflow						
To Register/Memory	00001111	10010001	mod 000 r/m	4/5*	0/1*	a
SETB/SETNAE = Set Byte on Below/Not Above or Equal						
To Register/Memory	00001111	10010010	mod 000 r/m	4/5*	0/1*	a

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format			Clock Counts	Number of Data Cycles	Notes
CONDITIONAL BYTE SET (Continued)						
SETNB = Set Byte on Not Below/Above or Equal						
To Register/Memory	00001111	10010011	mod000 r/m	4/5*	0/1*	a
SETE/SETZ = Set Byte on Equal/Zero						
To Register/Memory	00001111	10010100	mod000 r/m	4/5*	0/1*	a
SETNE/SETNZ = Set Byte on Not Equal/Not Zero						
To Register/Memory	00001111	10010101	mod000 r/m	4/5*	0/1*	a
SETBE/SETNA = Set Byte on Below or Equal/Not Above						
To Register/Memory	00001111	10010110	mod000 r/m	4/5*	0/1*	a
SETNBE/SETA = Set Byte on Not Below or Equal/Above						
To Register/Memory	00001111	10010111	mod000 r/m	4/5*	0/1*	a
SETS = Set Byte on Sign						
To Register/Memory	00001111	10011000	mod000 r/m	4/5*	0/1*	a
SETNS = Set Byte on Not Sign						
To Register/Memory	00001111	10011001	mod000 r/m	4/5*	0/1*	a
SETP/SETPE = Set Byte on Parity/Parity Even						
To Register/Memory	00001111	10011010	mod000 r/m	4/5*	0/1*	a
SETNP/SETPO = Set Byte on Not Parity/Parity Odd						
To Register/Memory	00001111	10011011	mod000 r/m	4/5*	0/1*	a
SETL/SETNGE = Set Byte on Less/Not Greater or Equal						
To Register/Memory	00001111	10011100	mod000 r/m	4/5*	0/1*	a
SETNL/SETGE = Set Byte on Not Less/Greater or Equal						
To Register/Memory	00001111	10011101	mod000 r/m	4/5*	0/1*	a
SETLE/SETNG = Set Byte on Less or Equal/Not Greater						
To Register/Memory	00001111	10011110	mod000 r/m	4/5*	0/1*	a
SETNLE/SETG = Set Byte on Not Less or Equal/Greater						
To Register/Memory	00001111	10011111	mod000 r/m	4/5*	0/1*	a
ENTER = Enter Procedure	11001000	16-bit displacement, 8-bit level				
L = 0				10		a
L = 1				14	1	a
L > 1				17 + 8(n − 1)	4(n − 1)	a
LEAVE = Leave Procedure	11001001			6		a

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts*	Number of Data Cycles	Notes
INTERRUPT INSTRUCTIONS				
INT = Interrupt:				
Type Specified	11001101 | type			
Via Interrupt or Trap Gate to Same Privilege Level		71	14	c,d,j,p
Via Interrupt or Trap Gate to Different Privilege Level		111	14	c,d,j,p
From 386 Task to 386 TSS via Task Gate		467	140	c,d,j,p
Type 3	11001100			
Via Interrupt or Trap Gate to Same Privilege Level		71	14	c,d,j,p
Via Interrupt or Trap Gate to Different Privilege Level		111	14	c,d,j,p
From 386 Task to 386 TSS via Task Gate		308	138	c,d,j,p
INTO = Interrupt 4 if Overflow Flag Set	11001110			
If OF = 1:		3		
If OF = 0				
Via Interrupt or Trap Gate to Same Privilege Level		71	14	c,d,j,p
Via Interrupt or Trap Gate to Different Privilege Level		111	14	c,d,j,p
From 386 Task to 386 TSS via Task Gate		413	138	c,d,j,p

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number Of Data Cycles	Notes
INTERRUPT INSTRUCTIONS (Continued)				
Bound = Out of Range Interrupt 5 If Detect Value	01100010 | mod reg r/m			
If in Range		13	0	a,c,d,j,o,p
If Out of Range:				
Via Interrupt or Trap Gate to Same Privilege Level		71	14	c,d,j,p
Via Interrupt or Trap Gate to Different Privilege Level		77	14	c,d,j,p
From 386 Task to 386 TSS via Task Gate		398	138	c,d,j,p
INTERRUPT RETURN				
IRET = Interrupt Return	11001111			
To the Same Privilege Level (within Task)		42	5	a,c,d,j,p
To Different Privilege Level (within Task)		86	5	a,c,d,j,p
From 386 Task to 386 TSS		328	138	c,d,j,p
PROCESSOR CONTROL				
HLT = HALT	11110100	5		b
MOV = Move to and from Control/Debug/Test Registers				
CR0 from register	00001111 | 00100010 | 11 eee reg	10		b
Register from CR0	00001111 | 00100000 | 11 eee reg	6		b
DR0–3 from Register	00001111 | 00100011 | 11 eee reg	22		b
DR6–7 from Register	00001111 | 00100011 | 11 eee reg	16		b
Register from DR6–7	00001111 | 00100001 | 11 eee reg	14		b
Register from DR0–3	00001111 | 00100001 | 11 eee reg	22		b
NOP = No Operation	10010000	3		
WAIT = Wait until BUSY# Pin is Negated	10011011	6		

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number of Data Cycles	Notes
PROCESSOR EXTENSION INSTRUCTIONS				
Processor Extension Escape	11011TTT | mod LLL r/m TTT and LLL bits are opcode information for coprocessor.	See 80387SX Data Sheet		a
PREFIX BYTES				
Address Size Prefix	01100111	0		f
LOCK = Bus Lock Prefix	11110000	0		f
Operand Size Prefix	01100110	0		f
Segment Override Prefix				
CS:	00101110	0		f
DS:	00111110	0		f
ES:	00100110	0		f
FS:	01100100	0		f
GS:	01100101	0		f
SS:	00110110	0		f
PROTECTION CONTROL				
ARPL = Adjust Requested Privilege Level				
From Register/Memory	01100011 | mod reg r/m	20/21**	2**	a
LAR = Load Access Rights				
From Register/Memory	00001111 | 00000010 | mod reg r/m	17/18*	1*	a,c,i,p
LGDT = Load Global Descriptor				
Table Register	00001111 | 00000001 | mod 010 r/m	13**	3*	a,e
LIDT = Load Interrupt Descriptor				
Table Register	00001111 | 00000001 | mod 011 r/m	13**	3*	a,e
LLDT = Load Local Descriptor				
Table Register to Register/Memory	00001111 | 00000000 | mod 010 r/m	24/28*	5*	a,c,e,p
LMSW = Load Machine Status Word				
From Register/Memory	00001111 | 00000001 | mod 110 r/m	10/13*	1*	a,e
LSL = Load Segment Limit				
From Register/Memory	00001111 | 00000011 | mod reg r/m			
Byte-Granular Limit		24/27*	2*	a,c,i,p
Page-Granular Limit		29/32*	2*	a,c,i,p
LTR = Load Task Register				
From Register/Memory	00001111 | 00000000 | mod 001 r/m	27/31*	4*	a,c,e,p
SGDT = Store Global Descriptor				
Table Register	00001111 | 00000001 | mod 000 r/m	11*	3*	a
SIDT = Store Interrupt Descriptor				
Table Register	00001111 | 00000001 | mod 001 r/m	11*	3*	a
SLDT = Store Local Descriptor Table Register				
To Register/Memory	00001111 | 00000000 | mod 000 r/m	2/2*	4*	a

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number of Data Cycles	Notes
PROTECTION CONTROL (Continued)				
SMSW = Store Machine Status Word	00001111 | 00000001 | mod 1 0 0 r/m	2/2*	1*	a, c
STR = Store Task Register				
To Register/Memory	00001111 | 00000000 | mod 0 0 1 r/m	2/2*	1*	a
VERR = Verify Read Access				
Register/Memory	00001111 | 00000000 | mod 1 0 0 r/m	10/11**	2**	a,c,i,p
VERW = Verify Write Access	00001111 | 00000000 | mod 1 0 1 r/m	15/16**	2**	a,c,i,p

NOTES:

a. Exception 13 fault (general violation) will occur if the memory operand in CS, DS, ES, FS or GS cannot be used due to either a segment limit violation or access rights violation. If a stack limit is violated, and exception 12 (stack segment limit violation or not present) occurs.

b. For segment load operations, the CPL, RPL and DPL must agree with the privilege rules to avoid an exception 13 fault (general protection violation). The segments's descriptor must indicate "present" or exception 11 (CS, DS, ES, FS, GS not present). If the SS register is loaded and a stack segment not present is detected, an exception 12 (stack segment limit violation or not present occurs).

c. All segment descriptor accesses in the GDT or LDT made by this instruction will automatically assert LOCK# to maintain descriptor integrity in multiprocessor systems.

d. JMP, CALL, INT, RET and IRET instructions referring to another code segment will cause an exception 13 (general protection violation) if an applicable privilege rule is volated.

e. An exception 13 fault occurs if CPL is greater than 0.

f. An exception 13 fault occurs if CPL is greater than IOPL.

g. The IF bit of the flag register is not updated if CPL is greater than IOPL. The IOPL field of the flag register is updated only if CPL = 0.

h. Any violation of privelege rules as applied to the selector operand does not cause a protection exception; rather, the zero flag is cleared.

i. If the coprocessor's memory operand violates a segment limit or segment access rights, an exception 13 fault (general protection exception) will occur before the ESC instruction is executed. An exception 12 fault (stack segment limit violation or no present) will occur if the stack limit is violated by the operand's starting address.

j. The destination of a JMP, CALL, INT, RET or IRET must be in the defined limit of a code segment or an exception 13 fault (general protection violation) will occur.

k. If CPL $\leq$ IOPL

l. If CPL > IOPL

m. LOCK# is automatically asserted, regardless of the presence or absence of the LOCK# prefix.

n. The 80376 uses an early-out multiply algorithm. The actual number of clocks depends on the position of the most significant bit in the operand (multiplier). Clock counts given are minimum to maximum. To calculate actual clocks use the following formula:

Actual Clock = if m < > 0 then max ($[\log_2 |m|]$, 3) + 9 clocks:
if m = 0 then 12 clocks (where m is the multiplier)

o. An exception may occur, depending on the value of the operand.

p. LOCK# is asserted during descriptor table accesses.

8.2 INSTRUCTION ENCODING

Overview

All instruction encodings are subsets of the general instruction format shown in Figure 8.1. Instructions consist of one or two primary opcode bytes, possibly an address specifier consisting of the "mod r/m" byte and "scaled index" byte, a displacement if required, and an immediate data field if required.

Within the primary opcode or opcodes, smaller encoding fields may be defined. These fields vary according to the class of operation. The fields define such information as direction of the operation, size of the displacements, register encoding, or sign extension.

Almost all instructions referring to an operand in memory have an addressing mode byte following the primary opcode byte(s). This byte, the mod r/m byte, specifies the address mode to be used. Certain encodings of the mod r/m byte indicate a second addressing byte, the scale-index-base byte, follows the mod r/m byte to fully specify the addressing mode.

Addressing modes can include a displacement immediately following the mod r/m byte, or scaled index byte. If a displacement is present, the possible sizes are 8, 16 or 32 bits.

If the instruction specifies an immediate operand, the immediate operand follows any displacement bytes. The immediate operand, if specified, is always the last field of the instruction.

Figure 8.1 illustrates several of the fields that can appear in an instruction, such as the mod field and the r/m field, but the Figure does not show all fields. Several smaller fields also appear in certain instructions, sometimes within the opcode bytes themselves. Table 8.2 is a complete list of all fields appearing in the 80376 instruction set. Further ahead, following Table 8.2, are detailed tables for each field.

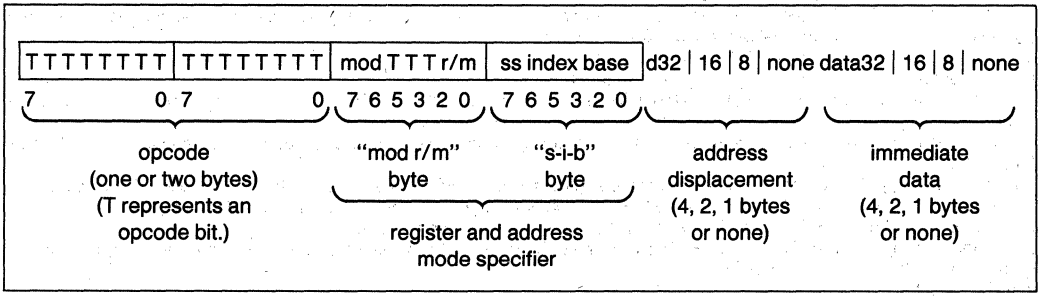


Figure 8.1. General Instruction Format

Table 8.2. Fields within 80376 Instructions

Field Name	Description	Number of Bits
w	Specifies if Data is Byte or Full Size (Full Size is either 16 or 32 Bits	1
d	Specifies Direction of Data Operation	1
s	Specifies if an Immediate Data Field Must be Sign-Extended	1
reg	General Register Specifier	3
mod r/m	Address Mode Specifier (Effective Address can be a General Register)	2 for mod; 3 for r/m
ss	Scale Factor for Scaled Index Address Mode	2
index	General Register to be used as Index Register	3
base	General Register to be used as Base Register	3
sreg2	Segment Register Specifier for CS, SS, DS, ES	2
sreg3	Segment Register Specifier for CS, SS, DS, ES, FS, GS	3
tttn	For Conditional Instructions, Specifies a Condition Asserted or a Condition Negated	4

Note: Table 8.1 shows encoding of individual instructions.

16-Bit Extensions of the Instruction Set

Two prefixes, the Operand Size Prefix (66H) and the Effective Address Size Prefix (67H), allow overriding individually the Default selection of operand size and effective address size. These prefixes may precede any opcode bytes and affect only the instruction they precede. If necessary, one or both of the prefixes may be placed before the opcode bytes. The presence of the Operand Size Prefix and the Effective Address Prefix will allow 16-bit data operation and 16-bit effective address calculations.

For instructions with more than one prefix, the order of prefixes is unimportant.

Unless specified otherwise, instructions with 8-bit and 16-bit operands do not affect the contents of the high-order bits of the extended registers.

Encoding of Instruction Fields

Within the instruction are several fields indicating register selection, addressing mode and so on.

ENCODING OF OPERAND LENGTH (w) FIELD

For any given instruction performing a data operation, the instruction will execute as a 32-bit operation. Within the constraints of the operation size, the w field encodes the operand size as either one byte or the full operation size, as shown in the table below.

w Field	Operand Size During 16-Bit Data Operations	Operand Size During 32-Bit Data Operations
0	8 Bits	8 Bits
1	16 Bits	32 Bits

ENCODING OF THE GENERAL REGISTER (reg) FIELD

The general register is specified by the reg field, which may appear in the primary opcode bytes, or as the reg field of the "mod r/m" byte, or as the r/m field of the "mod r/m" byte.

Encoding of reg Field When w Field is not Present in Instruction

reg Field	Register Selected During 16-Bit Data Operations	Register Selected During 32-Bit Data Operations
000	AX	EAX
001	CX	ECX
010	DX	EDX
011	BX	EBX
100	SP	ESP
101	BP	EBP
101	SI	ESI
101	DI	EDI

Encoding of reg Field When w Field is Present in Instruction

Register Specified by reg Field During 16-Bit Data Operations:		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Register Specified by reg Field During 32-Bit Data Operations		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	EAX
001	CL	ECX
010	DL	EDX
011	BL	EBX
100	AH	ESP
101	CH	EBP
110	DH	ESI
111	BH	EDI

ENCODING OF THE SEGMENT REGISTER (sreg) FIELD

The sreg field in certain instructions is a 2-bit field allowing one of the CS, DS, ES or SS segment registers to be specified. The sreg field in other instructions is a 3-bit field, allowing the FS and GS segment registers to be specified also.

2-Bit sreg2 Field

2-Bit sreg2 Field	Segment Register Selected
00	ES
01	CS
10	SS
11	DS

3-Bit sreg3 Field

3-Bit sreg3 Field	Segment Register Selected
000	ES
001	CS
010	SS
011	DS
100	FS
101	GS
110	do not use
111	do not use

ENCODING OF ADDRESS MODE

Except for special instructions, such as PUSH or POP, where the addressing mode is pre-determined, the addressing mode for the current instruction is specified by addressing bytes following the primary opcode. The primary addressing byte is the "mod r/m" byte, and a second byte of addressing information, the "s-i-b" (scale-index-base) byte, can be specified.

The s-i-b byte (scale-index-base byte) is specified when using 32-bit addressing mode and the "mod r/m" byte has r/m = 100 and mod = 00, 01 or 10. When the sib byte is present, the 32-bit addressing mode is a function of the mod, ss, index, and base fields.

The primary addressing byte, the "mod r/m" byte, also contains three bits (shown as TTT in Figure 8.1) sometimes used as an extension of the primary opcode. The three bits, however, may also be used as a register field (reg).

When calculating an effective address, either 16-bit addressing or 32-bit addressing is used. 16-bit addressing uses 16-bit address components to calculate the effective address while 32-bit addressing uses 32-bit address components to calculate the effective address. When 16-bit addressing is used, the "mod r/m" byte is interpreted as a 16-bit addressing mode specifier. When 32-bit addressing is used, the "mod r/m" byte is interpreted as a 32-bit addressing mode specifier.

Tables on the following three pages define all encodings of all 16-bit addressing modes and 32-bit addressing modes.

Encoding of Normal Address Mode with "mod r/m" byte (no "s-i-b" byte present):

mod r/m	Effective Address
00 000	DS:[EAX]
00 001	DS:[ECX]
00 010	DS:[EDX]
00 011	DS:[EBX]
00 100	s-i-b is present
00 101	DS:d32
00 110	DS:[ESI]
00 111	DS:[EDI]
01 000	DS:[EAX + d8]
01 001	DS:[ECX + d8]
01 010	DS:[EDX + d8]
01 011	DS:[EBX + d8]
01 100	s-i-b is present
01 101	SS:[EBP + d8]
01 110	DS:[ESI + d8]
01 111	DS:[EDI + d8]

mod r/m	Effective Address
10 000	DS:[EAX + d32]
10 001	DS:[ECX + d32]
10 010	DS:[EDX + d32]
10 011	DS:[EBX + d32]
10 100	s-i-b is present
10 101	SS:[EBP + d32]
10 110	DS:[ESI + d32]
10 111	DS:[EDI + d32]
11 000	register—see below
11 001	register—see below
11 010	register—see below
11 011	register—see below
11 100	register—see below
11 101	register—see below
11 110	register—see below
11 111	register—see below

Register Specified by reg or r/m during Normal Data Operations:		
mod r/m	function of w field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Register Specified by reg or r/m during 16-Bit Data Operations: (66H Prefix)		
mod r/m	function of w field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Encoding of 16-bit Address Mode with "mod r/m" Byte Using 67H Prefix

mod r/m	Effective Address
00 000	DS:[BX + SI]
00 001	DS:[BX + DI]
00 010	SS:[BP + SI]
00 011	SS:[BP + DI]
00 100	DS:[SI]
00 101	DS:[DI]
00 110	DS:d16
00 111	DS:[BX]
01 000	DS:[BX + SI + d8]
01 001	DS:[BX + DI + d8]
01 010	SS:[BP + SI + d8]
01 011	SS:[BP + DI + d8]
01 100	DS:[SI + d8]
01 101	DS:[DI + d8]
01 110	SS:[BP + d8]
01 111	DS:[BX + d8]

mod r/m	Effective Address
10 000	DS:[BX + SI + d16]
10 001	DS:[BX + DI + d16]
10 010	SS:[BP + SI + d16]
10 011	SS:[BP + DI + d16]
10 100	DS:[SI + d16]
10 101	DS:[DI + d16]
10 110	SS:[BP + d16]
10 111	DS:[BX + d16]
11 000	register—see below
11 001	register—see below
11 010	register—see below
11 011	register—see below
11 100	register—see below
11 101	register—see below
11 110	register—see below
11 111	register—see below

Encoding of 32-bit Address Mode ("mod r/m" byte and "s-i-b" byte present):

mod base	Effective Address
00 000	DS:[EAX + (scaled index)]
00 001	DS:[ECX + (scaled index)]
00 010	DS:[EDX + (scaled index)]
00 011	DS:[EBX + (scaled index)]
00 100	SS:[ESP + (scaled index)]
00 101	DS:[d32 + (scaled index)]
00 110	DS:[ESI + (scaled index)]
00 111	DS:[EDI + (scaled index)]
01 000	DS:[EAX + (scaled index) + d8]
01 001	DS:[ECX + (scaled index) + d8]
01 010	DS:[EDX + (scaled index) + d8]
01 011	DS:[EBX + (scaled index) + d8]
01 100	SS:[ESP + (scaled index) + d8]
01 101	SS:[EBP + (scaled index) + d8]
01 110	DS:[ESI + (scaled index) + d8]
01 111	DS:[EDI + (scaled index) + d8]
10 000	DS:[EAX + (scaled index) + d32]
10 001	DS:[ECX + (scaled index) + d32]
10 010	DS:[EDX + (scaled index) + d32]
10 011	DS:[EBX + (scaled index) + d32]
10 100	SS:[ESP + (scaled index) + d32]
10 101	SS:[EBP + (scaled index) + d32]
10 110	DS:[ESI + (scaled index) + d32]
10 111	DS:[EDI + (scaled index) + d32]

NOTE:
Mod field in "mod r/m" byte; ss, index, base fields in "s-i-b" byte.

ss	Scale Factor
00	x1
01	x2
10	x4
11	x8

index	Index Register
000	EAX
001	ECX
010	EDX
011	EBX
100	no index reg**
101	EBP
110	ESI
111	EDI

****IMPORTANT NOTE:**
When index field is 100, indicating "no index register," then ss field MUST equal 00. If index is 100 and ss does not equal 00, the effective address is undefined.

ENCODING OF OPERATION DIRECTION (d) FIELD

In many two-operand instructions the d field is present to indicate which operand is considered the source and which is the destination.

d	Direction of Operation
0	Register/Memory <- - Register "reg" Field Indicates Source Operand; "mod r/m" or "mod ss index base" Indicates Destination Operand
1	Register <- - Register/Memory "reg" Field Indicates Destination Operand; "mod r/m" or "mod ss index base" Indicates Source Operand

ENCODING OF SIGN-EXTEND (s) FIELD

The s field occurs primarily to instructions with immediate data fields. The s field has an effect only if the size of the immediate data is 8 bits and is being placed in a 16-bit or 32-bit destination.

s	Effect on Immediate Data8	Effect on Immediate Data 16|32
0	None	None
1	Sign-Extend Data8 to Fill 16-Bit or 32-Bit Destination	None

ENCODING OF CONDITIONAL TEST (tttn) FIELD

For the conditional instructions (conditional jumps and set on condition), tttn is encoded with n indicating to use the condition (n = 0) or its negation (n = 1), and tt giving the condition to test.

Mnemonic	Condition	tttn
O	Overflow	0000
NO	No Overflow	0001
B/NAE	Below/Not Above or Equal	0010
NB/AE	Not Below/Above or Equal	0011
E/Z	Equal/Zero	0100
NE/NZ	Not Equal/Not Zero	0101
BE/NA	Below or Equal/Not Above	0110
NBE/A	Not Below or Equal/Above	0111
S	Sign	1000
NS	Not Sign	1001
P/PE	Parity/Parity Even	1010
NP/PO	Not Parity/Parity Odd	1011
L/NGE	Less Than/Not Greater or Equal	1100
NL/GE	Not Less Than/Greater or Equal	1101
LE/NG	Less Than or Equal/Greater Than	1110
NLE/G	Not Less or Equal/Greater Than	1111

ENCODING OF CONTROL OR DEBUG REGISTER (eee) FIELD

For the loading and storing of the Control and Debug registers.

When Interpreted as Control Register Field

eee Code	Reg Name
000	CR0
010	Reserved
011	Reserved
Do not use any other encoding	

When Interpreted as Debug Register Field

eee Code	Reg Name
000	DR0
001	DR1
010	DR2
011	DR3
110	DR6
111	DR7
Do not use any other encoding	

9.0 REVISION HISTORY

This 80376 data sheet, version -002, contains updates and improvements to previous versions. A revision summary is listed here for your convenience.

The sections significantly revised since version -001 are:

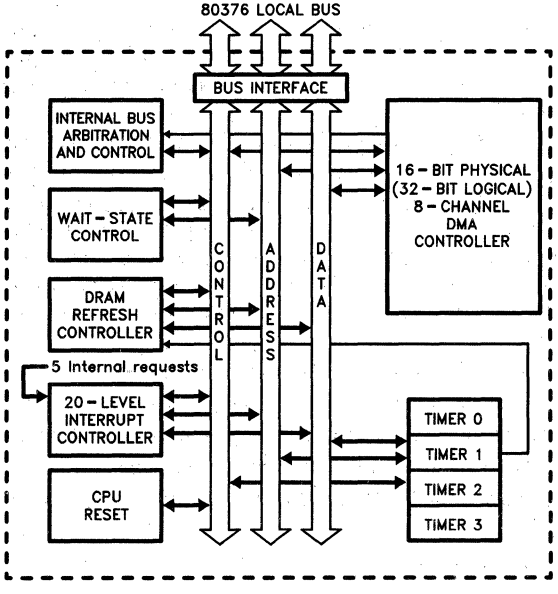
Front Page	The 80376 Microarchitecture diagram was added.
Section 1.0	Figure 1.2 was updated to show both top and bottom views of the 88-pin PGA package.
Section 2.0	Figure 2.0 was updated to show the 16-bit registers SI, DI, BP and SP.
Section 2.1	Figure 2.2 was updated to show the correct bit polarity for bit 4 in the CR0 register.
Section 2.1	Tables 2.1 and 2.2 were updated to include additional information on the EFLAGs and CR0 registers.
Section 2.3	Figure 2.3 was updated to more accurately reflect the addressing mechanism of the 80376.
Section 2.6	In the subsection **Maskable Interrupt** a paragraph was added to describe the effect of interrupt gates on the IF EFLAGs bit.
Section 2.8	Table 2.7 was updated to reflect the correct power up condition of the CR0 register.
Section 2.10	Figure 2.6 was updated to show the correct bit positions of the BT, BS and BD bits in the DR6 register.
Section 3.0	Figure 3.1 was updated to clearly show the address calculation process.
Section 3.2	The subsection **DESCRIPTORS** was elaborated upon to clearly define the relationship between the linear address space and physical address space of the 80376.
Section 3.2	Figures 3.3 and 3.4 were updated to show the AVL bit field.
Section 3.3	The last sentence in the first paragraph of subsection **PROTECTION AND I/O PERMISSION BIT MAP** was deleted. This was an incorrect statement.
Section 4.1	In the Subsection **ADDRESS BUS (BHE#, BLE#, A_{23}–A_1** last sentence in the first paragraph was updated to reflect the numerics operand addresses as 8000FCH and 8000FEH. Because the 80376 sometimes does a double word I/O access a second access to 8000FEH can be seen.
Section 4.1	The Subsection **Hold Lantencies** was updated to describe how 32-bit and unaligned accesses are internally locked but do not assert the LOCK# signal.
Section 4.2	Table 4.6 was updated to show the correct active data bits during a BLE# assertion.
Section 4.4	This section was updated to correctly reflect the pipelining of the address and status of the 80376 as opposed to "Address Pipelining" which occurs on processors such as the 80286.
Section 4.6	Table 4.7 was updated to show the correct Revision number, 05H.
Section 4.7	Table 4.8 was updated to show the numerics operand register 8000FEH. This address is seen when the 80376 does a DWORD operation to the port address 8000FCH.
Section 5.0	In the first paragraph the case temperatures were updated to correctly reflect the 0°C–115°C for the ceramic package and 0°C–110°C for the plastic package.
Section 6.2	Table 6.2 was updated to correctly reflect the Case Temperature under Bias specification of −65°C–120°C.
Section 6.4	Figure 6.8 vertical axis was updated to reflect "Output Valid Delay (ns)".
Section 6.4	Figure 6.11 was updated to show typical I_{CC} vs Frequency for the 80376.
Section 6.5	This entire section was updated to reflect the new ICE-376 emulator.
Section 8.1	The clock counts and opcodes for various instructions were updated to their correct value.
Section 8.2	The section **INSTRUCTION ENCODING** was appended to the data sheet.

82370
INTEGRATED SYSTEM PERIPHERAL

- **High Performance 32-Bit DMA Controller for 16-Bit Bus**
  - **— 16 MBytes/Sec Maximum Data Transfer Rate at 16 MHz**
  - **— 8 Independently Programmable Channels**
- **20-Source Interrupt Controller**
  - **— Individually Programmable Interrupt Vectors**
  - **— 15 External, 5 Internal Interrupts**
  - **— 82C59A Superset**
- **Four 16-Bit Programmable Interval Timers**
  - **— 82C54 Compatible**
- **Software Compatible to 82380**
- **Programmable Wait State Generator**
  - **— 0 to 15 Wait States Pipelined**
  - **— 0 to 16 Wait States Non-Pipelined**
- **DRAM Refresh Controller**
- **80376 Shutdown Detect and Reset Control**
  - **— Software/Hardware Reset**
- **High Speed CHMOS III Technology**
- **100-Pin Plastic Quad Flat-Pack Package and 132-Pin Pin Grid Array Package**
  (See Packaging Handbook Order #231369)
- **Optimized for Use with the 80376 Microprocessor**
  - **— Resides on Local Bus for Maximum Bus Bandwidth**

The 82370 is a multi-function support peripheral that integrates system functions necessary in an 80376 environment. It has eight channels of high performance 32-bit DMA (32-bit internal, 16-bit external) with the most efficient transfer rates possible on the 80376 bus. System support peripherals integrated into the 82370 provide Interrupt Control, Timers, Wait State generation, DRAM Refresh Control, and System Reset logic.

The 82370's DMA Controller can transfer data between devices of different data path widths using a single channel. Each DMA channel operates independently in any of several modes. Each channel has a temporary data storage register for handling non-aligned data without the need for external alignment logic.



290164-1

Internal Block Diagram

October 1988
Order Number: 290164-002

Pin Descriptions

The 82370 provides all of the signals necessary to interface an 80376 host processor. It has a separate 24-bit address and 16-bit data bus. It also has a set of control signals to support operation as a bus master or a bus slave. Several special function signals exist on the 82370 for interfacing the system support peripherals to their respective system counterparts. Following are the definitions of the individual pins of the 82370. These brief descriptions are provided as a reference. Each signal is further defined within the sections which describe the associated 82370 function.

Symbol	Type	Name and Function
A_1–A_{23}	I/O	**ADDRESS BUS:** Outputs physical memory or port I/O addresses. See **Address Bus** (2.2.3) for additional information.
BHE# BLE#	I/O	**BYTE ENABLES:** Indicate which data bytes of the data bus take part in a bus cycle. See **Byte Enable** (2.2.4) for additional information.
D_0–D_{15}	I/O	**DATA BUS:** This is the 16-bit data bus. These pins are active outputs during interrupt acknowledges, during Slave accesses, and when the 82370 is in the Master Mode.
CLK2	I	**PROCESSOR CLOCK:** This pin must be connected to the processor's clock, CLK2. The 82370 monitors the phase of this clock in order to remain synchronized with the CPU. This clock drives all of the internal synchronous circuitry.
D/C#	I/O	**DATA/CONTROL:** D/C# is used to distinguish between CPU control cycles and DMA or CPU data access cycles. It is active as an output only in the Master Mode.
W/R#	I/O	**WRITE/READ:** W/R# is used to distinguish between write and read cycles. It is active as an output only in the Master Mode.
M/IO#	I/O	**MEMORY/IO:** M/IO# is used to distinguish between memory and IO accesses. It is active as an output only in the Master Mode.
ADS#	I/O	**ADDRESS STATUS:** This signal indicates presence of a valid address on the address bus. It is active as output only in the Master Mode. ADS# is active during the first T-state where addresses and control signals are valid.
NA#	I	**NEXT ADDRESS:** Asserted by a peripheral or memory to begin a pipelined address cycle. This pin is monitored only while the 82370 is in the Master Mode. In the Slave Mode, pipelining is determined by the current and past status of the ADS# and READY# signals.
HOLD	O	**HOLD REQUEST:** This is an active-high signal to the Bus Master to request control of the system bus. When control is granted, the Bus Master activates the hold acknowledge signal (HLDA).
HLDA	I	**HOLD ACKNOWLEDGE:** This input signal tells the DMA controller that the Bus Master has relinquished control of the system bus to the DMA controller.

Pin Descriptions (Continued)

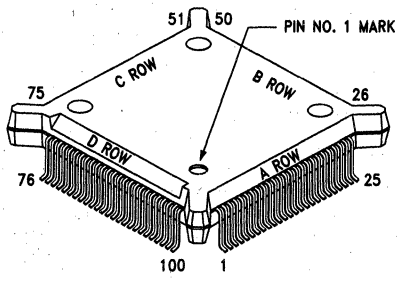
Symbol	Type	Name and Function
DREQ (0–3, 5–7)	I	**DMA REQUEST:** The DMA Request inputs monitor requests from peripherals requiring DMA service. Each of the eight DMA channels has one DREQ input. These active-high inputs are internally synchronized and prioritized. Upon request, channel 0 has the highest priority and channel 7 the lowest.
DREQ4/IRQ9#	I	**DMA/INTERRUPT REQUEST:** This is the DMA request input for channel 4. It is also connected to the interrupt controller via interrupt request 9. This internal connection is available for DMA channel 4 only. The interrupt input is active low and can be programmed as either edge or level triggered. Either function can be masked by the appropriate mask register. Priorities of the DMA channel and the interrupt request are not related but follow the rules of the individual controllers. Note that this pin has a weak internal pull-up. This causes the interrupt request to be inactive, but the DMA request will be active if there is no external connection made. Most applications will require that either one or the other of these functions be used, but not both. For this reason, it is advised that DMA channel 4 be used for transfers where a software request is more appropriate (such as memory-to-memory transfers). In such an application, DREQ4 can be masked by software, freeing IRQ9# for other purposes.
EOP#	I/O	**END OF PROCESS:** As an output, this signal indicates that the current Requester access is the last access of the currently operating DMA channel. It is activated when Terminal Count is reached. As an input, it signals the DMA channel to terminate the current buffer and proceed to the next buffer, if one is available. This signal may be programmed as an asynchronous or synchronous input. EOP# must be connected to a pull-up resistor. This will prevent erroneous external requests for termination of a DMA process.
EDACK (0–2)	O	**ENCODED DMA ACKNOWLEDGE:** These signals contain the encoded acknowledgment of a request for DMA service by a peripheral. The binary code formed by the three signals indicates which channel is active. Channel 4 does not have a DMA acknowledge. The inactive state is indicated by the code 100. During a Requester access, EDACK presents the code for the active DMA channel. During a Target access, EDACK presents the inactive code 100.
IRQ (11–23)#	I	**INTERRUPT REQUEST:** These are active low interrupt request inputs. The inputs can be programmed to be edge or level sensitive. Interrupt priorities are programmable as either fixed or rotating. These inputs have weak internal pull-up resistors. Unused interrupt request inputs should be tied inactive externally.
INT	O	**INTERRUPT OUT:** INT signals that an interrupt request is pending.
CLKIN	I	**TIMER CLOCK INPUT:** This is the clock input signal to all of the 82370's programmable timers. It is independent of the system clock input (CLK2).
TOUT1/REF#	O	**TIMER 1 OUTPUT/REFRESH:** This pin is software programmable as either the direct output of Timer 1, or as the indicator of a refresh cycle in progress. As REF#, this signal is active during the memory read cycle which occurs during refresh.

Pin Descriptions (Continued)

Symbol	Type	Name and Function
TOUT2#/IRQ3#	I/O	**TIMER 2 OUTPUT/INTERRUPT REQUEST:** This is the inverted output of Timer 2. It is also connected directly to interrupt request 3. External hardware can use IRQ3# if Timer 2 is programmed as OUT = 0 (TOUT2# = 1).
TOUT3#	O	**TIMER 3 OUTPUT:** This is the inverted output of Timer 3.
READY#	I	**READY INPUT:** This active-low input indicates to the 82370 that the current bus cycle is complete. READY is sampled by the 82370 both while it is in the Master Mode, and while it is in the Slave Mode.
WSC (0-1)	I	**WAIT STATE CONTROL:** WSC0 and WSC1 are inputs used by the Wait-State Generator to determine the number of wait states required by the currently accessed memory or I/O. The binary code on these pins, combined with the M/IO# signal, selects an internal register in which a wait-state count is stored. The combination WSC = 11 disables the wait-state generator.
READYO#	O	**READY OUTPUT:** This is the synchronized output of the wait-state generator. It is also valid during CPU accesses to the 82370 in the Slave Mode when the 82370 requires wait states. READYO# should feed directly the processor's READY# input.
RESET	I	**RESET:** This synchronous input serves to initialize the state of the 82370 and provides basis for the CPURST output. RESET must be held active for at least 15 CLK2 cycles in order to guarantee the state of the 82370. After Reset, the 82370 is in the Slave Mode with all outputs except timers and interrupts in their inactive states. The state of the timers and interrupt controller must be initialized through software. This input must be active for the entire time required by the host processor to guarantee proper reset.
CHPSEL#	O	**CHIP SELECT:** This pin is driven active whenever the 82370 is addressed in a slave bus read or write cycle. It is also active during interrupt acknowledge cycles when the 82370 is driving the Data Bus. It can be used to control the local bus transceivers to prevent contention with the system bus.
CPURST	O	**CPU RESET:** CPURST provides a synchronized reset signal for the CPU. It is activated in the event of a software reset command, a processor shut-down detect, or a hardware reset via the RESET pin. The 82370 holds CPURST active for 62 clocks in response to either a software reset command or a shut-down detection. Otherwise CPURST reflects the RESET input.
V_{CC}		**POWER:** +5V input power.
V_{SS}		Ground Reference.

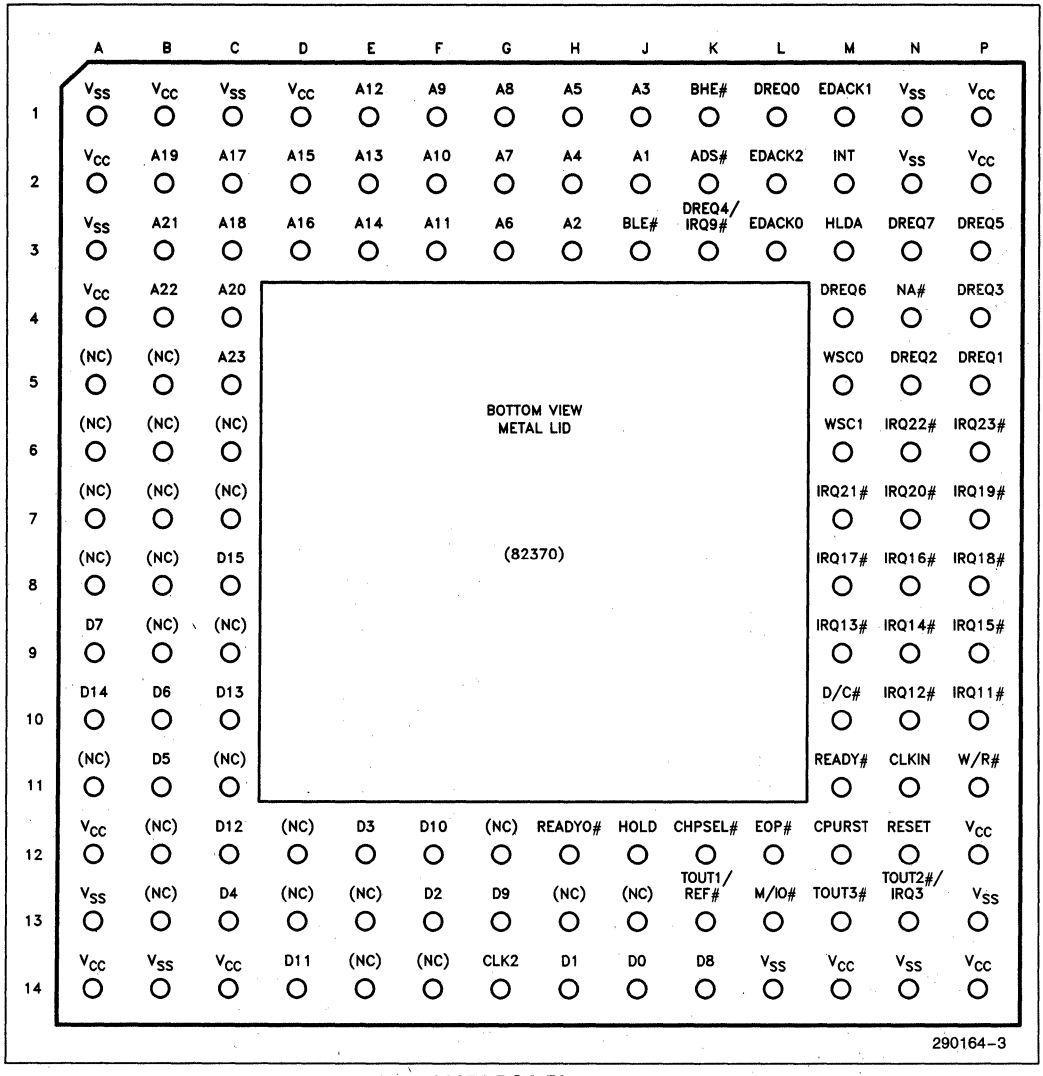
Table 1. Wait-State Select Inputs

Port Address	Wait-State Registers D7 D4	Wait-State Registers D3 D0	Select Inputs WSC1	Select Inputs WSC0
72H	MEMORY 0	I/O 0	0	0
73H	MEMORY 1	I/O 1	0	1
74H	MEMORY 2	I/O 2	1	0
	DISABLED		1	1
M/IO#	1	0		



100 Pin Quad Flat-Pack Pin Out (Top View)

A Row		B Row		C Row		D Row	
Pin	**Label**	**Pin**	**Label**	**Pin**	**Label**	**Pin**	**Label**
1	CPURST	26	V_{CC}	51	A_{11}	76	DREQ5
2	INT	27	D_{11}	52	A_{10}	77	DREQ4/IRQ9#
3	V_{CC}	28	D_4	53	A_9	78	DREQ3
4	V_{SS}	29	D_{12}	54	A_8	79	DREQ2
5	TOUT2#/IRQ3#	30	D_5	55	A_7	80	DREQ1
6	TOUT3#	31	D_{13}	56	A_6	81	DREQ0
7	D/C#	32	D_6	57	A_5	82	IRQ23#
8	V_{CC}	33	V_{SS}	58	V_{CC}	83	IRQ22#
9	W/R#	34	D_{14}	59	A_4	84	IRQ21#
10	M/IO#	35	D_7	60	A_3	85	IRQ20#
11	HOLD	36	D_{15}	61	A_2	86	IRQ19#
12	TOUT1/REF#	37	A_{23}	62	A_1	87	IRQ18#
13	CLK2	38	A_{22}	63	V_{SS}	88	IRQ17#
14	V_{SS}	39	A_{21}	64	BLE#	89	IRQ16#
15	READYO#	40	A_{20}	65	BHE#	90	IRQ15#
16	EOP#	41	A_{19}	66	V_{SS}	91	IRQ14#
17	CHPSEL#	42	A_{18}	67	ADS#	92	IRQ13#
18	V_{CC}	43	V_{CC}	68	V_{CC}	93	IRQ12#
19	D_0	44	A_{17}	69	EDACK2	94	IRQ11#
20	D_8	45	A_{16}	70	EDACK1	95	CLKIN
21	D_1	46	A_{15}	71	EDACK0	96	WSC0
22	D_9	47	A_{14}	72	HLDA	97	WSC1
23	D_2	48	V_{SS}	73	DREQ7	98	RESET
24	D_{10}	49	A_{13}	74	DREQ6	99	READY#
25	D_3	50	A_{12}	75	NA#	100	V_{SS}



82370 PGA Pinout

Pin	Label	Pin	Label	Pin	Label	Pin	Label
G14	CLK2	D14	D_{11}	L1	DREQ0	A2	V_{CC}
N12	RESET	F12	D_{10}	P6	IRQ23#	P2	V_{CC}
M12	CPURST	G13	D_9	N6	IRQ22#	A4	V_{CC}
C5	A_{23}	K14	D_8	M7	IRQ21#	A12	V_{CC}
B4	A_{22}	A9	D_7	N7	IRQ20#	P12	V_{CC}
B3	A_{21}	B10	D_6	P7	IRQ19#	A14	V_{CC}
C4	A_{20}	B11	D_5	P8	IRQ18#	C14	V_{CC}
B2	A_{19}	C13	D_4	M8	IRQ17#	M14	V_{CC}
C3	A_{18}	E12	D_3	N8	IRQ16#	P14	V_{CC}
C2	A_{17}	F13	D_2	P9	IRQ15#	A5	NC
D3	A_{16}	H14	D_1	N9	IRQ14#	B5	NC
D2	A_{15}	J14	D_0	M9	IRQ13#	A6	NC
E3	A_{14}	P11	W/R#	N10	IRQ12#	B6	NC
E2	A_{13}	L13	M/IO#	P10	IRQ11#	C6	NC
E1	A_{12}	K2	ADS#	M5	WSC0	A7	NC
F3	A_{11}	M10	D/C#	M6	WSC1	B7	NC
F2	A_{10}	N4	NA#	M13	TOUT3#	C7	NC
F1	A_9	M11	READY#	N13	TOUT2#/IRQ3#	A8	NC
G1	A_8	H12	READYO#	K13	TOUT1/REF#	B8	NC
G2	A_7	J12	HOLD	N11	CLKIN	B9	NC
G3	A_6	M3	HLDA	A1	V_{SS}	C9	NC
H1	A_5	M2	INT	C1	V_{SS}	A11	NC
H2	A_4	L12	EOP#	N1	V_{SS}	B11	NC
J1	A_3	L2	EDACK2	N2	V_{SS}	C11	NC
H3	A_2	M1	EDACK1	A3	V_{SS}	D12	NC
J2	A_1	L3	EDACK0	A13	V_{SS}	G12	NC
J3	BLE#	N3	DREQ7	P13	V_{SS}	B13	NC
K1	BHE#	M4	DREQ6	B14	V_{SS}	D13	NC
K12	CHPSEL#	P3	DREQ5	L14	V_{SS}	E13	NC
C8	D_{15}	K3	DREQ4/IRQ9#	N14	V_{SS}	H13	NC
A10	D_{14}	P4	DREQ3	B1	V_{CC}	J13	NC
C10	D_{13}	N5	DREQ2	D1	V_{CC}	E14	NC
C12	D_{12}	P5	DREQ1	P1	V_{CC}	F14	NC

1.0 FUNCTIONAL OVERVIEW

The 82370 contains several independent functional modules. The following is a brief discussion of the components and features of the 82370. Each module has a corresponding detailed section later in this data sheet. Those sections should be referred to for design and programming information.

1.1 82370 Architecture

The 82370 is comprised of several computer system functions that are normally found in separate LSI and VLSI components. These include: a high-performance, eight-channel, 32-bit Direct Memory Access Controller; a 20-level Programmable Interrupt Controller which is a superset of the 82C59A; four 16-bit Programmable Interval Timers which are functionally equivalent to the 82C54 timers; a DRAM Refresh Controller; a Programmable Wait State Generator; and system reset logic. The interface to the 82370 is optimized for high-performance operation with the 80376 microprocessor.

The 82370 operates directly on the 80376 bus. In the Slave Mode, it monitors the state of the processor at all times and acts or idles according to the commands of the host. It monitors the address pipeline status and generates the programmed number of wait states for the device being accessed. The 82370 also has logic to the reset of the 80376 via hardware or software reset requests and processor shutdown status.

After a system reset, the 82370 is in the Slave Mode. It appears to the system as an I/O device. It becomes a bus master when it is performing DMA transfers.

To maintain compatibility with existing software, the registers within the 82370 are accessed as bytes. If the internal logic of the 82370 requires a delay before another access by the processor, wait states are automatically inserted into the access cycle. This allows the programmer to write initialization routines, etc. without regard to hardware recovery times.

Figure 1-1 shows the basic architectural components of the 82370. The following sections briefly discuss the architecture and function of each of the distinct sections of the 82370.

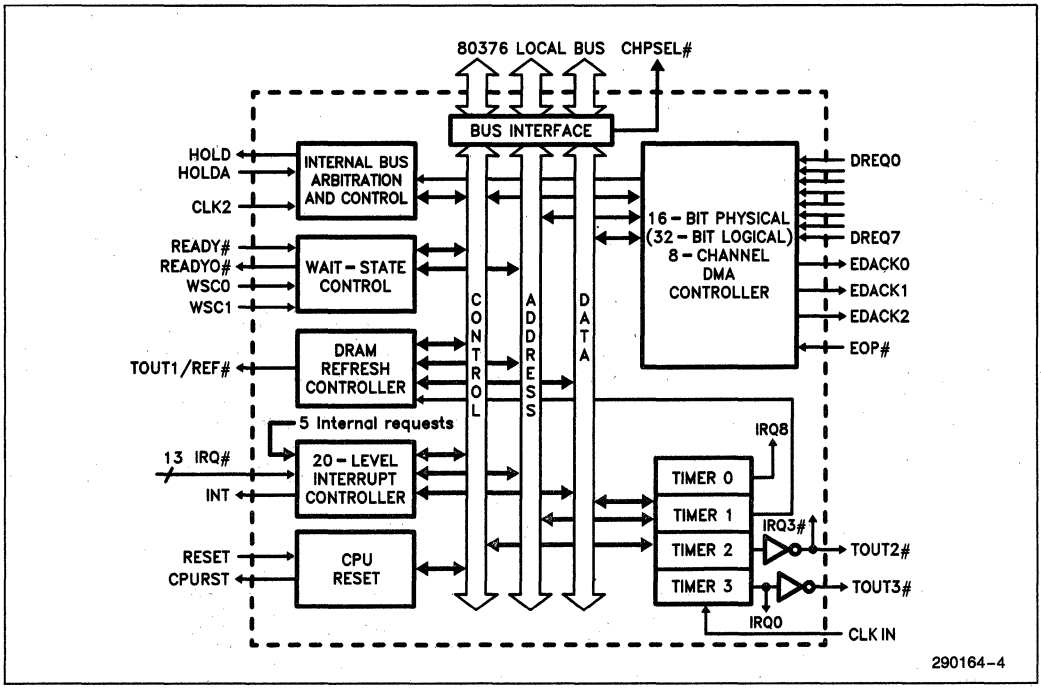


Figure 1-1. Architecture of the 82370

1.1.1 DMA CONTROLLER

The 82370 contains a high-performance, 8-channel DMA Controller. It provides a 32-bit internal data path. Through its 16-bit external physical data bus, it is capable of transferring data in any combination of bytes, words and double-words. The addresses of both source and destination can be independently incremented, decremented or held constant, and cover the entire 16-bit physical address space of the 80376. It can disassemble and assemble non-aligned data via a 32-bit internal temporary data storage register. Data transferred between devices of different data path widths can also be assembled and disassembled using the internal temporary data storage register. The DMA Controller can also transfer aligned data between I/O and memory on the fly, allowing data transfer rates up to 16 megabytes per second for an 82370 operating at 16 MHz. Figure 1-2 illustrates the functional components of the DMA Controller.

There are twenty-four general status and command registers in the 82370 DMA Controller. Through these registers any of the channels may be programmed into any of the possible modes. The operating modes of any one channel are independent of the operation of the other channels.

Each channel has three programmable registers which determine the location and amount of data to be transferred:

Byte Count Register— Number of bytes to transfer. (24-bits)

Requester Register — Byte Address of memory or peripheral which is requesting DMA service. (24-bits)

Target Register — Byte Address of peripheral or memory which will be accessed. (24-bits)

There are also port addresses which, when accessed, cause the 82370 to perform specific functions. The actual data written doesn't matter, the act of writing to the specific address causes the command to be executed. The commands which operate in this mode are: Master Clear, Clear Terminal Count Interrupt Request, Clear Mask Register, and Clear Byte Pointer Flip-Flop.

DMA transfers can be done between all combinations of memory and I/O; memory-to-memory, memory-to-I/O, I/O-to-memory, and I/O-to-I/O. DMA service can be requested through software and/or hardware. Hardware DMA acknowledge signals are available for all channels (except channel 4) through an encoded 3-bit DMA acknowledge bus (EDACK0–2).

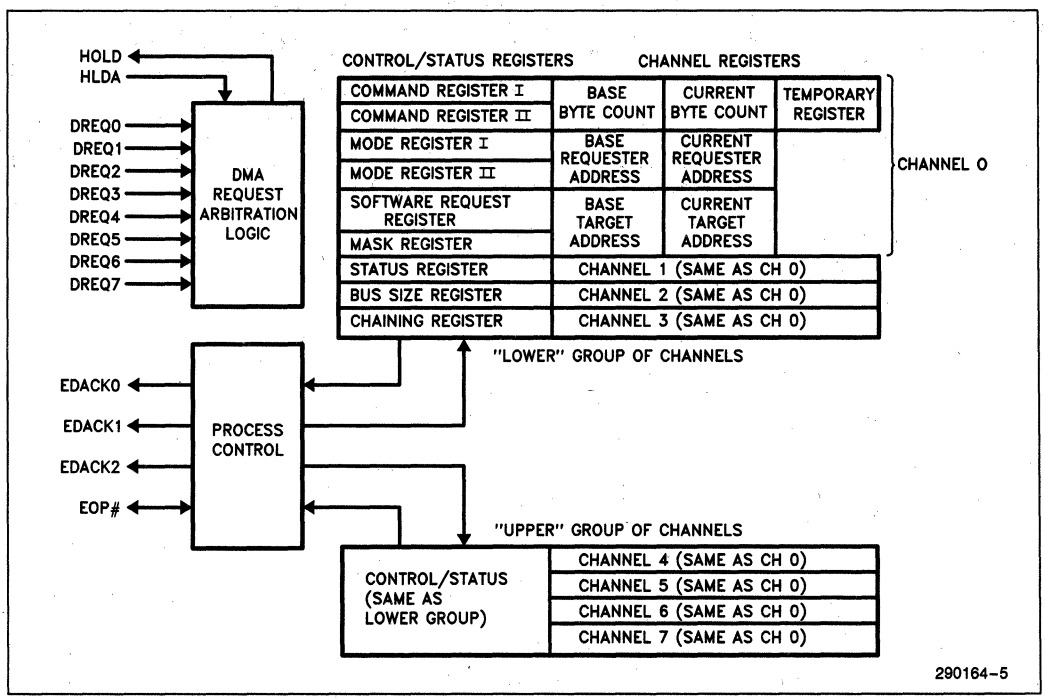


Figure 1-2. 82370 DMA Controller

The 82370 DMA Controller transfers blocks of data (buffers) in three modes: Single Buffer, Buffer Auto-Initialize, and Buffer Chaining. In the Single Buffer Process, the 82370 DMA Controller is programmed to transfer one particular block of data. Successive transfers then require reprogramming of the DMA channel. Single Buffer transfers are useful in systems where it is known at the time the transfer begins what quantity of data is to be transferred, and there is a contiguous block of data area available.

The Buffer Auto-Initialize Process allows the same data area to be used for successive DMA transfers without having to reprogram the channel.

The Buffer Chaining Process allows a program to specify a list of buffer transfers to be executed. The 82370 DMA Controller, through interrupt routines, is reprogrammed from the list. The channel is reprogrammed for a new buffer before the current buffer transfer is complete. This pipelining of the channel programming process allows the system to allocate non-contiguous blocks of data storage space, and transfer all of the data with one DMA process. The buffers that make up the chain do not have to be in contiguous locations.

Channel priority can be fixed or rotating. Fixed priority allows the programmer to define the priority of DMA channels based on hardware or other fixed parameters. Rotating priority is used to provide peripherals access to the bus on a shared basis.

With fixed priority, the programmer can set any channel to have the current lowest priority. This allows the user to reset or manually rotate the priority schedule without reprogramming the command registers.

1.1.2 PROGRAMMABLE INTERVAL TIMERS

Four 16-bit programmable interval timers reside within the 82370. These timers are identical in function to the timers in the 82C54 Programmable Interval Timer. All four of the timers share a common clock input which can be independent of the system clock. The timers are capable of operating in six different modes. In all of the modes, the current count can be latched and read by the 80376 at any time, making these very versatile event timers. Figure 1-3 shows the functional components of the Programmable Interval Timers.

The outputs of the timers are directed to key system functions, making system design simpler. Timer 0 is routed directly to an interrupt input and is not available externally. This timer would typically be used to generate time-keeping interrupts.

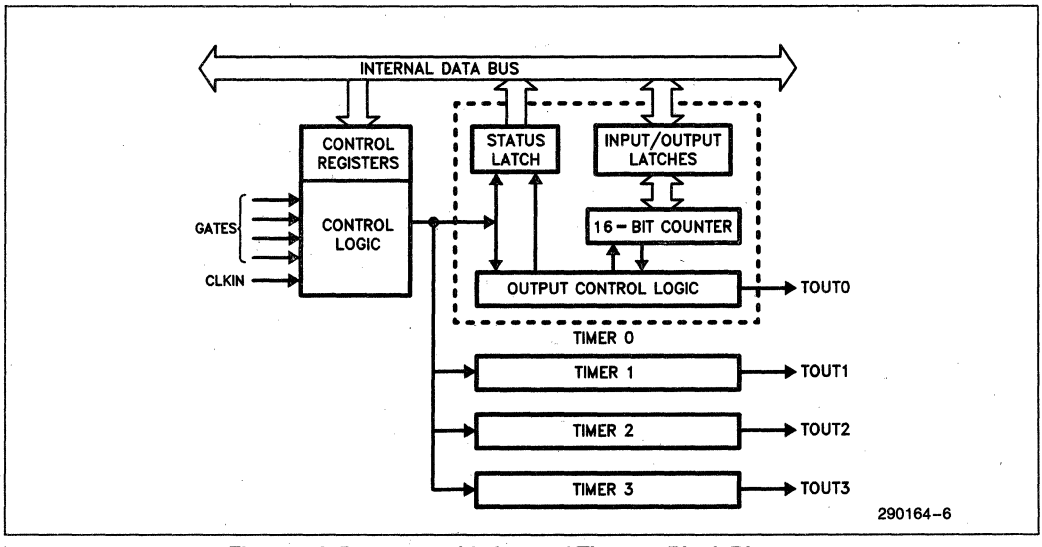


Figure 1-3. Programmable Interval Timers—Block Diagram

Timers 1 and 2 have outputs which are available for general timer/counter purposes as well as special functions. Timer 1 is routed to the refresh control logic to provide refresh timing. Timer 2 is connected to an interrupt request input to provide other timer functions. Timer 3 is a general purpose timer/counter whose output is available to external hardware. It is also connected internally to the interrupt request which defaults to the highest priority (IRQ0).

1.1.3 INTERRUPT CONTROLLER

The 82370 has the equivalent of three enhanced 82C59A Programmable Interrupt Controllers. These controllers can all be operated in the Master Mode, but the priority is always as if they were cascaded. There are 15 interrupt request inputs provided for the user, all of which can be inputs from external slave interrupt controllers. Cascading 82C59As to these request inputs allows a possible total of 120 external interrupt requests. Figure 1-4 is a block diagram of the 82370 Interrupt Controller.

Each of the interrupt request inputs can be individually programmed with its own interrupt vector, allowing more flexibility in interrupt vector mapping than was available with the 82C59A. An interrupt is provided to alert the system that an attempt is being made to program the vectors in the method of the 82C59A. This provides compatibility of existing software that used the 82C59A or 8259A with new designs using the 82370.

In the event of an unrequested or otherwise erroneous interrupt acknowledge cycle, the 82370 Interrupt Controller issues a default vector. This vector, programmed by the system software, will alert the system of unsolicited interrupts of the 80376.

The functions of the 82370 Interrupt Controller are identical to the 82C59A, except in regards to programming the interrupt vectors as mentioned above. Interrupt request inputs are programmable as either edge or level triggered and are software maskable. Priority can be either fixed or rotating and interrupt requests can be nested.

Enhancements are added to the 82370 for cascading external interrupt controllers. Master to Slave handshaking takes place on the data bus, instead of dedicated cascade lines.

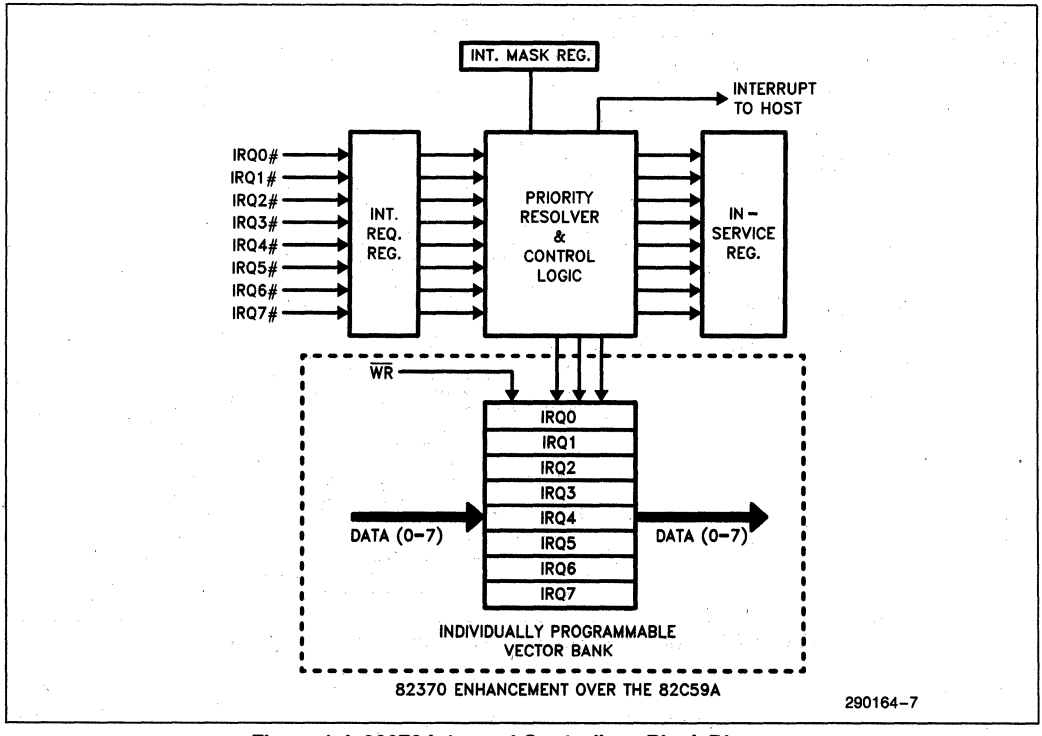


Figure 1-4. 82370 Interrupt Controller—Block Diagram

1.1.4 WAIT STATE GENERATOR

The Wait State Generator is a programmable READY generation circuit for the 80376 bus. A peripheral requiring wait states can request the Wait State Generator to hold the processor's READY input inactive for a predetermined number of bus states. Six different wait state counts can be programmed into the Wait State Generator by software; three for memory accesses and three for I/O accesses. A block diagram of the 82370 Wait State Generator is shown in Figure 1-5.

The peripheral being accessed selects the required wait state count by placing a code on a 2-bit wait state select bus. This code along with the M/IO# signal from the bus master is used to select one of six internal 4-bit wait state registers which has been programmed with the desired number of wait states. From zero to fifteen wait states can be programmed into the wait state registers. The Wait State generator tracks the state of the processor or current bus master at all times, regardless of which device is the current bus master and regardless of whether or not the wait state generator is currently active.

The 82370 Wait State Generator is disabled by making the select inputs both high. This allows hardware which is intelligent enough to generate its own ready signal to be accessed without penalty. As previously mentioned, deselecting the Wait State Generator does not disable its ability to determine the proper number of wait states due to pipeline status in subsequent bus cycles.

The number of wait states inserted into a pipelined bus cycle is the value in the selected wait state register. If the bus master is operating in the non-pipelined mode, the Wait State Generator will increase the number of wait states inserted into the bus cycle by one.

On reset, the Wait State Generator's registers are loaded with the value FFH, giving the maximum number of wait states for any access in which the wait state select inputs are active.

1.1.5 DRAM REFRESH CONTROLLER

The 82370 DRAM Refresh Controller consists of a 24-bit refresh address counter and bus arbitration logic. The output of Timer 1 is used to periodically request a refresh cycle. When the controller receives the request, it requests access to the system bus through the HOLD signal. When bus control is acknowledged by the processor or current bus master, the refresh controller executes a memory read operation at the address currently in the Refresh Address Register. At the same time, it activates a refresh signal (REF#) that the memory uses to force a refresh instead of a normal read. Control of the bus is transferred to the processor at the completion of this cycle. Typically a refresh cycle will take six clock cycles to execute on an 80376 bus.

The 82370 DRAM Refresh Controller has the highest priority when requesting bus access and will interrupt any active DMA process. This allows large blocks of data to be moved by the DMA controller without affecting the refresh function. Also the DMA controller is not required to completely relinquish the bus, the refresh controller simply steals a bus cycle between DMA accesses.

The amount by which the refresh address is incremented is programmable to allow for different bus widths and memory bank arrangements.

1.1.6 CPU RESET FUNCTION

The 82370 contains a special reset function which can respond to hardware reset signals as well as a

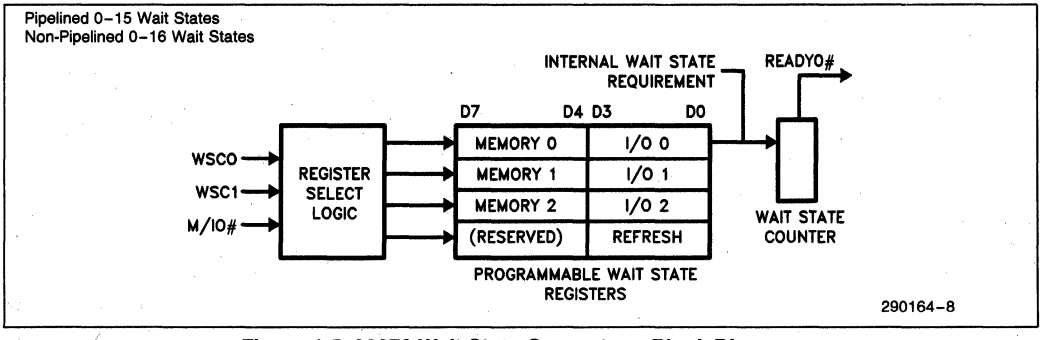


Figure 1-5. 82370 Wait State Generator—Block Diagram

software reset command. The circuit will hold the 80376's RESET line active while an external hardware reset signal is present at its RESET input. It can also reset the 80376 processor as the result of a software command. The software reset command causes the 82370 to hold the processor's RESET line active for a minimum of 62 clock cycles. The 80376 requires that its RESET line be held active for a minimum of 80 clock cycles to re-initialize. For a more detailed explanation and solution, see Appendix D (System Notes).

The 82370 can be programmed to sense the shutdown detect code on the status lines from the 80376. If the Shutdown Detect function is enabled, the 82370 will automatically reset the processor. A diagnostic register is available which can be used to determine the cause of reset.

1.1.7 REGISTER MAP RELOCATION

After a hardware reset, the internal registers of the 82370 are located in I/O space beginning at port address 0000H. The map of the 82370's registers is relocatable via a software command. The default mapping places the 82370 between I/O addresses 0000H and 00DBH. The relocation register allows this map to be moved to any even 256-byte boundary in the processor's 16-bit I/O address space or any even 64 kbyte boundary in the 24-bit memory address space.

1.2 Host Interface

The 82370 is designed to operate efficiently on the local bus of an 80376 microprocessor. The control signals of the 82370 are identical in function to those of the 80376. As a slave, the 82370 operates with all of the features available on the 80376 bus. When the 82370 is in the Master Mode, it looks identical to an 80376 to the connected devices.

The 82370 monitors the bus at all times, and determines whether the current bus cycle is a pipelined or non-pipelined access. All of the status signals of the processor are monitored.

The control, status, and data registers within the 82370 are located at fixed addresses relative to each other, but the group can be relocated to either memory or I/O space and to different locations within those spaces.

As a Slave device, the 82370 monitors the control/status lines of the CPU. The 82370 will generate all of the wait states it needs whenever it is accessed. This allows the programmer the freedom of accessing 82370 registers without having to insert NOPs in the program to wait for slower 82370 internal registers.

The 82370 can determine if a current bus cycle is a pipelined or a non-pipelined cycle. It does this by monitoring the ADS#, NA# and READY# signals and thereby keeping track of the current state of the 80376.

As a bus master, the 82370 looks like an 80376 to the rest of the system. This enables the designer greater flexibility in systems which include the 82370. The designer does not have to alter the interfaces of any peripherals designed to operate with the 80376 to accommodate the 82370. The 82370 will access any peripherals on the bus in the same manner as the 80376, including recognizing pipelined bus cycles.

The 82370 is accessed as an 8-bit peripheral. The 80376 places the data of all 8-bit accesses either on D(0–7) or D(8–15). The 82370 will only accept data on these lines when in the Slave Mode. When in the Master Mode, the 82370 is a full 16-bit machine, sending and receiving data in the same manner as the 80376.

2.0 80376 HOST INTERFACE

The 82370 contains a set of interface signals to operate efficiently with the 80376 host processor. These signals were designed so that minimal hardware is needed to connect the 82370 to the 80376. Figure 2-1 depicts a typical system configuration with the 80376 processor. As shown in the diagram, the 82370 is designed to interface directly with the 80376 bus.

Since the 82370 resides on the opposite side of the data bus transceivers with respect to the rest of the system peripherals, it is important to note that the transceivers should be controlled so that contention between the data bus transceivers and the 82370 will not occur. In order to ease the implementation of this, the 82370 activates the CHPSEL# signal which indicates that the 82370 has been addressed and may output data. This signal should be included in the direction and enable control logic of the transceiver. When any of the 82370 internal registers are read, the data bus transceivers should be disabled so that only the 82370 will drive the local bus.

This section describes the basic bus functions of the 82370 to show how this device interacts with the 80376 processor. Other signals which are not directly related to the host interface will be discussed in their associated functional block description.

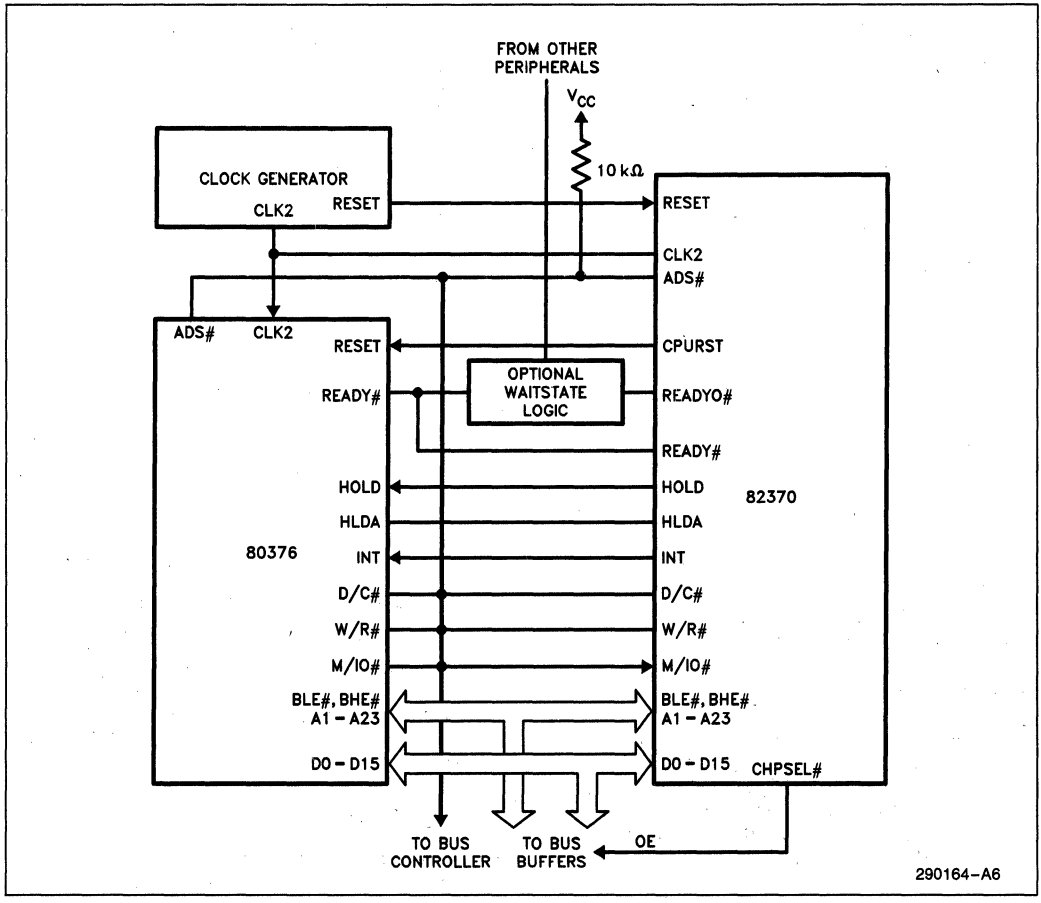


Figure 2-1. 80376/82370 System Configuration

2.1 Master and Slave Modes

At any time, the 82370 acts as either a Slave device or a Master device in the system. Upon reset, the 82370 will be in the Slave Mode. In this mode, the 80376 processor can read/write into the 82370 internal registers. Initialization information may be programmed into the 82370 during Slave Mode.

When DMA service (including DRAM Refresh Cycles generated by the 82370) is requested, the 82370 will request and subsequently get control of the 80376 local bus. This is done through the HOLD and HLDA (Hold Acknowledge) signals. When the 80376 processor responds by asserting the HLDA signal, the 82370 will switch into Master Mode and perform DMA transfers. In this mode, the 82370 is the bus master of the system. It can read/write data from/to memory and peripheral devices. The 82370 will return to the Slave Mode upon completion of DMA transfers, or when HLDA is negated.

2.2 80376 Interface Signals

As mentioned in the Architecture section, the Bus Interface module of the 82370 (see Figure 1-1) contains signals that are directly connected to the 80376 host processor. This module has separate

16-bit Data and 24-bit Address busses. Also, it has additional control signals to support different bus operations on the system. By residing on the 80376 local bus, the 82370 shares the same address, data and control lines with the processor. The following subsections discuss the signals which interface to the 80376 host processor.

2.2.1 CLOCK (CLK2)

The CLK2 input provides fundamental timing for the 82370. It is divided by two internally to generate the 82370 internal clock. Therefore, CLK2 should be driven with twice the 80376's frequency. In order to maintain synchronization with the 80376 host processor, the 82370 and the 80376 should share a common clock source.

The internal clock consists of two phases: PHI1 and PHI2. Each CLK2 period is a phase of the internal clock. PHI2 is usually used to sample input and set up internal signals and PHI1 is for latching internal data. Figure 2-2 illustrates the relationship of CLK2 and the 82370 internal clock signals. The CPURST signal generated by the 82370 guarantees that the 80376 will wake up in phase with PHI1.

2.2.2 DATA BUS (D_0-D_{15})

This 16-bit three-state bidirectional bus provides a general purpose data path between the 82370 and the system. These pins are tied directly to the corresponding Data Bus pins of the 80376 local bus. The Data Bus is also used for interrupt vectors generated by the 82370 in the Interrupt Acknowledge cycle.

During Slave I/O operations, the 82370 expects a single byte to be written or read. When the 80376 host processor writes into the 82370, either D_0-D_7 or D_8-D_{15} will be latched into the 82370, depending upon whether Byte Enable bit BLE# is 0 or 1 (see Table 2-1). When the 80376 host processor reads from the 82370, the single byte data will be duplicated twice on the Data Bus; i.e. on D_0-D_7 and D_8-D_{15}.

During Master Mode, the 82370 can transfer 16-, and 8-bit data between memory (or I/O devices) and I/O devices (or memory) via the Data Bus.

2.2.3 ADDRESS BUS (A_{23}-A_1)

These three-state bidirectional signals are connected directly to the 80376 Address Bus. In the Slave Mode, they are used as input signals so that the processor can address the 82370 internal ports/registers. In the Master Mode, they are used as output signals by the 82370 to address memory and peripheral devices. The Address Bus is capable of addressing 16 Mbytes of physical memory space (000000H to FFFFFFH), and 64 Kbytes of I/O addresses.

2.2.4 BYTE ENABLE (BHE#, BLE#)

The Byte Enable pins BHE# and BLE# select the specific byte(s) in the word addressed by A_1-A_{23}. During Master Mode operation, it is used as an output by the 82370 to address memory and I/O locations. The definition of BHE# and BLE# is further illustrated in Table 2-1.

NOTE:
The 82370 will activate BHE# when output in Master Mode. For a more detailed explanation and its solutions, see Appendix D (System Notes).

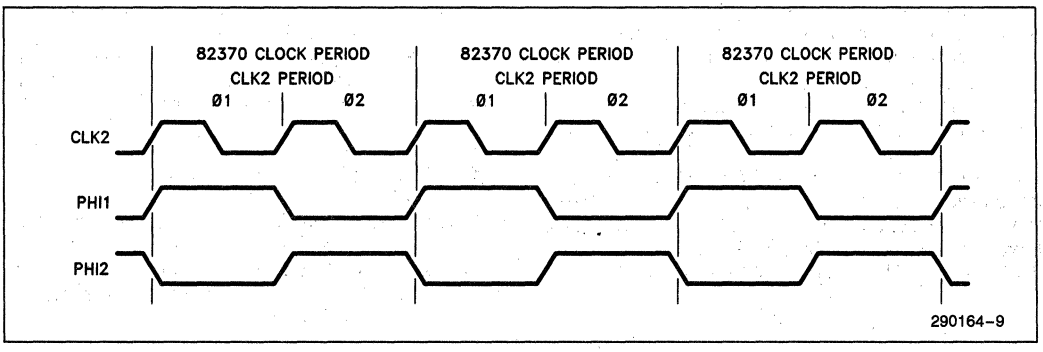


Figure 2-2. CLK2 and 82370 Internal Clock

As an output (Master Mode):

Table 2-1. Byte Enable Signals

BHE#	BLE#	Byte to be Accessed Relative to A_{23}–A_1	Logical Byte Presented on Data Bus During WRITE Only* D_{15}–D_8	D_7–D_0
0	0	0, 1	B	A
0	1	1	A	A
1	0	0	U	A
1	1	(Not Used)		

U = Undefined
A = Logical D_0–D_7
B = Logical D_8–D_{15}

***NOTE:**
Actual number of bytes accessed depends upon the programmed data path width.

Table 2-2. Bus Cycle Definition

M/IO#	D/C#	W/R#	As INPUTS	As OUTPUTS
0	0	0	Interrupt Acknowledge	NOT GENERATED
0	0	1	UNDEFINED	NOT GENERATED
0	1	0	I/O Read	I/O Read
0	1	1	I/O Write	I/O Write
1	0	0	UNDEFINED	NOT GENERATED
1	0	1	HALT if A_1 = 1 SHUTDOWN if A_1 = 0	NOT GENERATED
1	1	0	Memory Read	Memory Read
1	1	1	Memory Write	Memory Write

2.2.5 BUS CYCLE DEFINITION SIGNALS (D/C#, W/R#, M/IO#)

These three-state bidirectional signals define the type of bus cycle being performed. W/R# distinguishes between write and read cycles. D/C# distinguishes between processor data and control cycles. M/IO# distinguishes between memory and I/O cycles.

During Slave Mode, these signals are driven by the 80376 host processor; during Master Mode, they are driven by the 82370. In either mode, these signals will be valid when the Address Status (ADS#) is driven LOW. Exact bus cycle definitions are given in Table 2-2. Note that some combinations are recognized as inputs, but not generated as outputs. In the Master Mode, D/C# is always HIGH.

2.2.6 ADDRESS STATUS (ADS#)

This signal indicates that a valid address (A_1–A_{23}, BHE#, BLE#) and bus cycle definition (W/R#, D/C#, M/IO#) is being driven on the bus. In the Master Mode, it is driven by the 82370 as an output. In the Slave Mode, this signal is monitored as an input by the 82370. By the current and past status of ADS# and the READY# input, the 82370 is able to determine, during Slave Mode, if the next bus cycle is a pipelined address cycle. ADS# is asserted during T1 and T2P bus states (see Bus State Definition).

NOTE:
ADS# must be qualified with the rising edge of CLK2.

2.2.7 TRANSFER ACKNOWLEDGE (READY#)

This input indicates that the current bus cycle is complete. In the Master Mode, assertion of this signal indicates the end of a DMA bus cycle. In the Slave Mode, the 82370 monitors this input and ADS# to detect a pipelined address cycle. This signal should be tied directly to the READY# input of the 80376 host processor.

2.2.8 NEXT ADDRESS REQUEST (NA#)

This input is used to indicate to the 82370 in the Master Mode that the system is requesting address

pipelining. When driven LOW by either memory or peripheral devices during Master Mode, it indicates that the system is prepared to accept a new address and bus cycle definition signals from the 82370 before the end of the current bus cycle. If this input is active when sampled by the 82370, the next address is driven onto the bus, provided a bus request is already pending internally.

This input pin is monitored only in the Master Mode. In the Slave Mode, the 82370 uses the ADS# and READY# signals to determine address pipelining cycles, and NA# will be ignored.

2.2.9 RESET (RESET, CPURST)

RESET

This synchronous input suspends any operation in progress and places the 82370 in a known initial state. Upon reset, the 82370 will be in the Slave Mode waiting to be initialized by the 80376 host processor. The 82370 is reset by asserting RESET for 15 or more CLK2 periods. When RESET is asserted, all other input pins are ignored, and all other bus pins are driven to an idle bus state as shown in Table 2-3. The 82370 will determine the phase of its internal clock following RESET going inactive.

RESET is level-sensitive and must be synchronous to the CLK2 signal. The RESET setup and hold time requirements are shown in Figure 2-3.

Table 2-3. Output Signals Following RESET

Signal	Level
A_1–A_{23}, D_0–D_{15}, BHE#, BLE#	Float
D/C#, W/R#, M/IO#, ADS#	Float
READYO#	'1'
EOP#	'1' (Weak Pull-UP)
EDACK2–EDACK0	'100'
HOLD	'0'
INT	UNDEFINED*
TOUT1/REF#, TOUT2#/IRQ3#, TOUT3#	UNDEFINED*
CPURST	'0'
CHPSEL#	'1'

***NOTE:**
The Interrupt Controller and Programmable Interval Timer are initialized by software commands.

CPURST

This output signal is used to reset the 80376 host processor. It will go active (HIGH) whenever one of the following events occurs: a) 82370's RESET input is active; b) a software RESET command is issued to the 82370; or c) when the 82370 detects a processor Shutdown cycle and when this detection feature is enabled (see CPU Reset and Shutdown Detect). When activated, CPURST will be held active for 62 clocks. The timing of CPURST is such that the 80376 processor will be in synchronization with the 82370. This timing is shown in Figure 2-4.

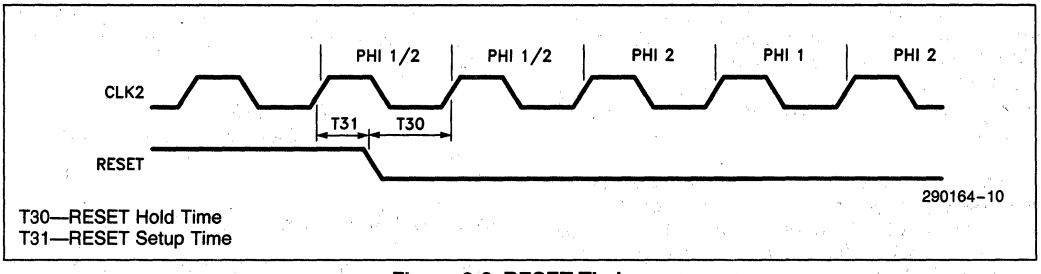


Figure 2-3. RESET Timing

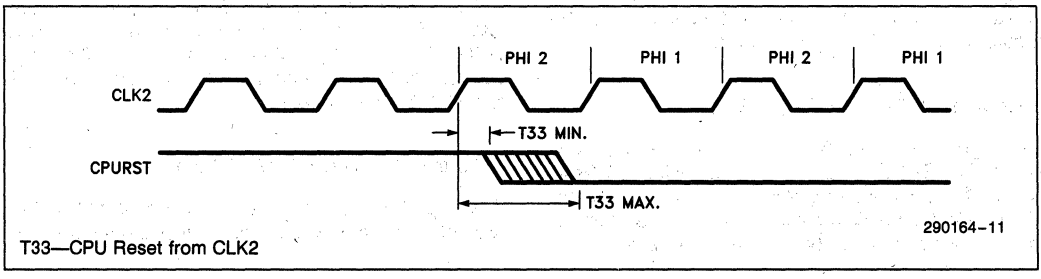


Figure 2-4. CPURST Timing

2.2.10 INTERRUPT OUT (INT)

This output pin is used to signal the 80376 host processor that one or more interrupt requests (either internal or external) are pending. The processor is expected to respond with an Interrupt Acknowledge cycle. This signal should be connected directly to the Maskable Interrupt Request (INTR) input of the 80376 host processor.

2.3 82370 Bus Timing

The 82370 internally divides the CLK2 signal by two to generate its internal clock. Figure 2-2 showed the relationship of CLK2 and the internal clock which consists of two phases: PHI1 and PHI2. Each CLK2 period is a phase of the internal clock.

In the 82370, whether it is in the Master or Slave Mode, the shortest time unit of bus activity is a bus state. A bus state, which is also referred as a 'T-state', is defined as one 82370 PHI2 clock period (i.e. two CLK2 periods). Recall in Table 2-2 various types of bus cycles in the 82370 are defined by the M/IO#, D/C# and W/R# signals. Each of these bus cycles is composed of two or more bus states. The length of a bus cycle depends on when the READY# input is asserted (i.e. driven LOW).

2.3.1 ADDRESS PIPELINING

The 82370 supports Address Pipelining as an option in both the Master and Slave Mode. This feature typically allows a memory or peripheral device to operate with one less wait state than would otherwise be required. This is possible because during a pipelined cycle, the address and bus cycle definition of the next cycle will be generated by the bus master while waiting for the end of the current cycle to be acknowledged. The pipelined bus is especially well suited for an interleaved memory environment. For 16 MHz interleaved memory designs with 100 ns access time DRAMs, zero wait state memory accesses can be achieved when pipelined addressing is selected.

In the Master Mode, the 82370 is capable of initiating, on a cycle-by-cycle basis, either a pipelined or non-pipelined access depending upon the state of the NA# input. If a pipelined cycle is requested (indicated by NA# being driven LOW), the 82370 will drive the address and bus cycle definition of the next cycle as soon as there is an internal bus request pending.

In the Slave Mode, the 82370 is constantly monitoring the ADS# and READY# signals on the processor local bus to determine if the current bus cycle is a pipelined cycle. If a pipelined cycle is detected, the 82370 will request one less wait state from the processor if the Wait State Generator feature is selected. On the other hand, during an 82370 internal register access in a pipelined cycle, it will make use of the advance address and bus cycle information. In all cases, Address Pipelining will result in a savings of one wait state.

2.3.2 MASTER MODE BUS TIMING

When the 82370 is in the Master Mode, it will be in one of six bus states. Figure 2-5 shows the complete bus state diagram of the Master Mode, including pipelined address states. As seen in the figure, the 82370 state diagram is very similar to that of the 80376. The major difference is that in the 82370, there is no Hold state. Also, in the 82370, the conditions for some state transitions depend upon whether it is the end of a DMA process.

NOTE:
The term 'end of a DMA process' is loosely defined here. It depends on the DMA modes of operation as well as the state of the EOP# and DREQ inputs. This is expained in detail in section 3—DMA Controller.

The 82370 will enter the idle state, Ti, upon RESET and whenever the internal address is not available at the end of a DMA cycle or at the end of a DMA process. When address pipelining is not used (NA# is not asserted), a new bus cycle always begins with state T1. During T1, address and bus cycle definition signals will be driven on the bus. T1 is always followed by T2.

If a bus cycle is not acknowledged (with READY#) during T2 and NA# is negated, T2 will be repeated. When the end of the bus cycle is acknowledged during T2, the following state will be T1 of the next bus cycle (if the internal address latch is loaded and if this is not the end of the DMA process). Otherwise, the Ti state will be entered. Therefore, if the memory or peripheral accessed is fast enough to respond within the first T2, the fastest non-pipelined cycle will take one T1 and one T2 state.

Use of the address pipelining feature allows the 82370 to enter three additional bus states: T1P, T2P and T2i. T1P is the first bus state of a pipelined bus cycle. T2P follows T1P (or T2) if NA# is asserted when sampled. The 82370 will drive the bus with the address and bus cycle definition signals of the next cycle during T2P. From the state diagram, it can be seen that after an idle state Ti, the first bus cycle must begin with T1, and is therefore a non-pipelined bus cycle. The next bus cycle can be pipelined if

NA# is asserted and the previous bus cycle ended in a T2P state. Once the 82370 is in a pipelined cycle and provided that NA# is asserted in subsequent cycles, the 82370 will be switching between T1P and T2P states. If the end of the current bus cycle is not acknowledged by the READY# input, the 82370 will extend the cycle by adding T2P states. The fastest pipelined cycle will consist of one T1P and one T2P state.

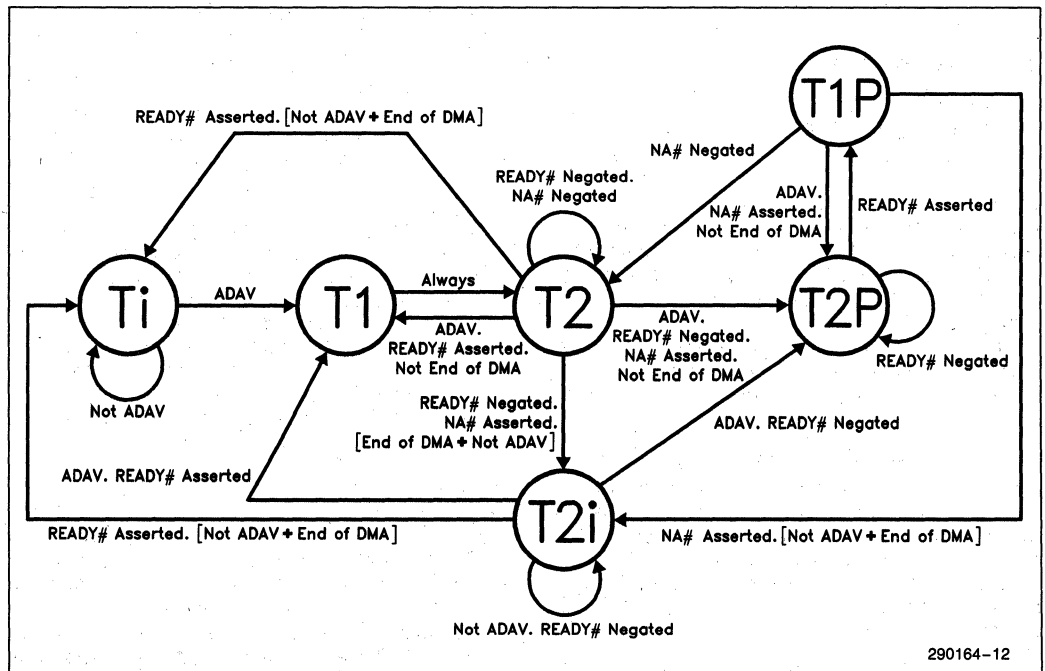
The 82370 will enter state T2i when NA# is asserted and when one of the following two conditions occurs. The first condition is when the 82370 is in state T2. T2i will be entered if READY# is not asserted and there is no next address available. This situation is similar to a wait state. The 82370 will stay in T2i for as long as this condition exists. The second condition which will cause the 82370 to enter T2i is when the 82370 is in state T1P. Before going to state T2P, the 82370 needs to wait in state T2i until the next address is available. Also, in both cases, if the DMA process is complete, the 82370 will enter the T2i state in order to finish the current DMA cycle.

Figure 2-6 is a timing diagram showing non-pipelined bus accesses in the Master Mode. Figure 2-7 shows the timing of pipelined accesses in the Master Mode.

2.3.3 SLAVE MODE BUS TIMING

Figure 2-8 shows the Slave Mode bus timing in both pipelined and non-pipelined cycles when the 82370 is being accessed. Recall that during Slave Mode, the 82370 will constantly monitor the ADS# and READY# signals to determine if the next cycle is pipelined. In Figure 2-8, the first cycle is non-pipelined and the second cycle is pipelined. In the pipelined cycle, the 82370 will start decoding the address and bus cycle signals one bus state earlier than in a non-pipelined cycle.

The READY# input signal is sampled by the 80376 host processor to determine the completion of a bus cycle. This occurs during the end of every T2, T2i and T2P state. Normally, the output of the 82370 Wait State Generator, READYO#, is directly connected to the READY# input of the 80376 host processor and the 82370. In such case, READYO# and READY# will be identical (see Wait State Generator).



NOTE:
ADAV—Internal Address Available

Figure 2-5. Master Mode State Diagram

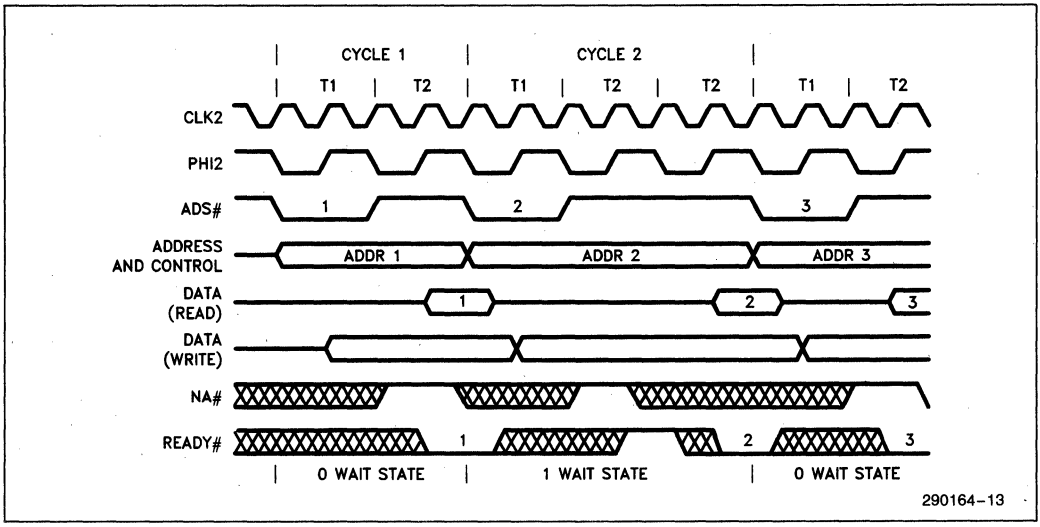


Figure 2-6. Non-Pipelined Bus Cycles

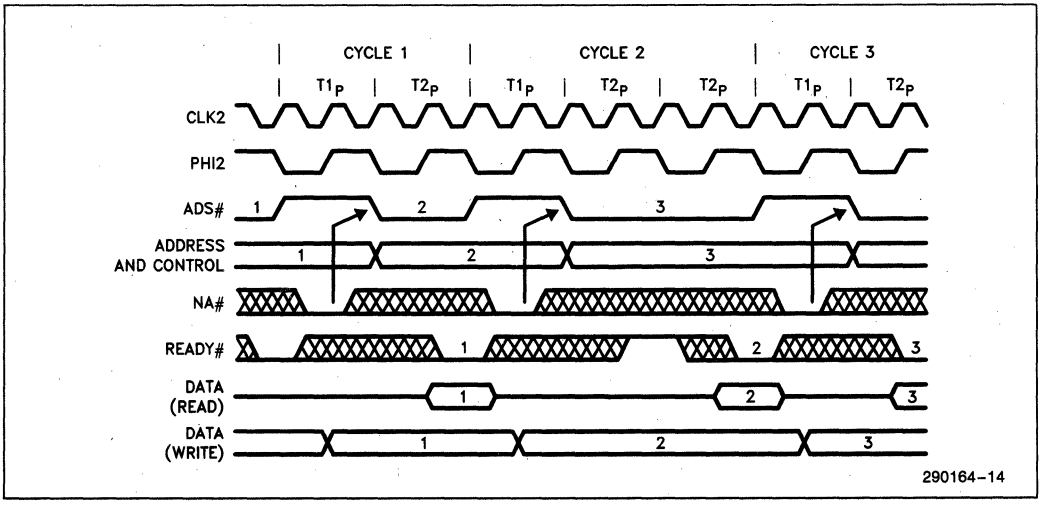
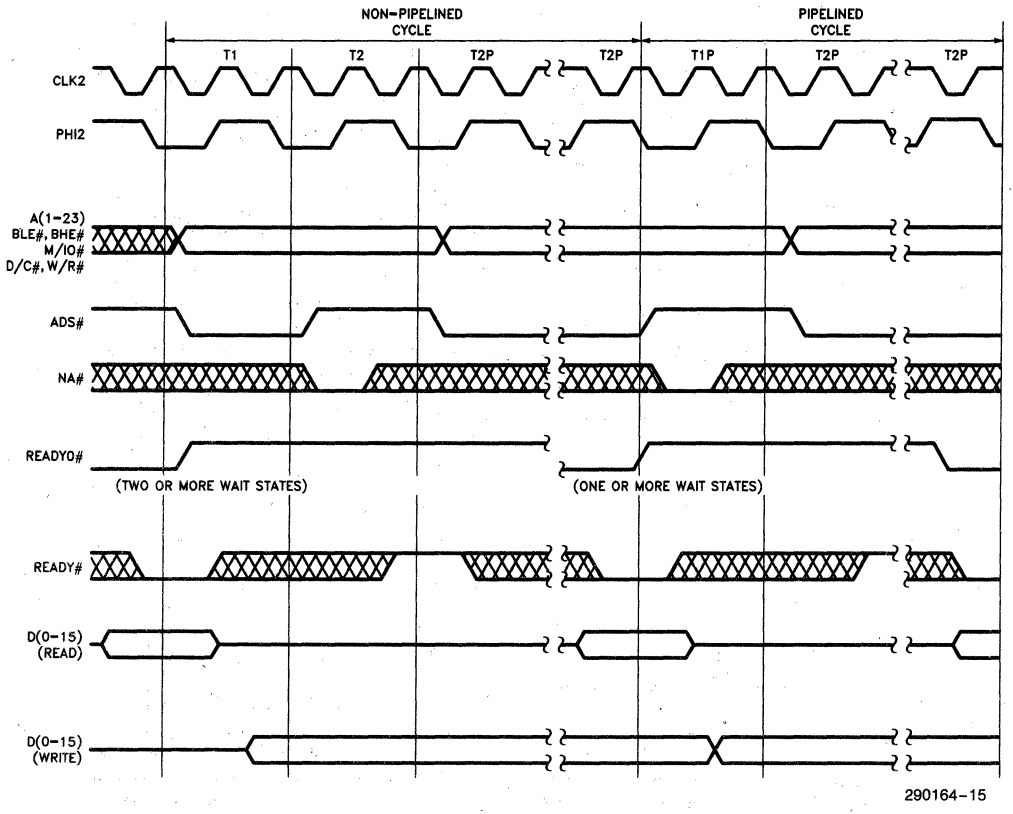


Figure 2-7. Pipelined Bus Cycles



NOTE:
NA# is shown here only for timing reference. It is not sampled by the 82370 during Slave Mode.
When the 82370 registers are accessed, it will take one or more wait states in pipelined and two or more wait states in non-pipelined cycle to complete the internal access.

Figure 2-8. Slave Read/Write Timing

3.0 DMA CONTROLLER

The 82370 DMA Controller is capable of transferring data between any combination of memory and/or I/O, with any combination of data path widths. The 82370 DMA Controller can be programmed to accommodate 8- or 16-bit devices. With its 16-bit external data path, it can transfer data in units of byte or a word. Bus bandwidth is optimized through the use of an internal temporary register which can disassemble or assemble data to or from either an aligned or non-aligned destination or source. Figure 3-1 is a block diagram of the 82370 DMA Controller.

The 82370 has eight channels of DMA. Each channel operates independently of the others. Within the operation of the individual channels, there are many different modes of data transfer available. Many of the operating modes can be intermixed to provide a very versatile DMA controller.

3.1 Functional Description

In describing the operation of the 82370's DMA Controller, close attention to terminology is required. Be-

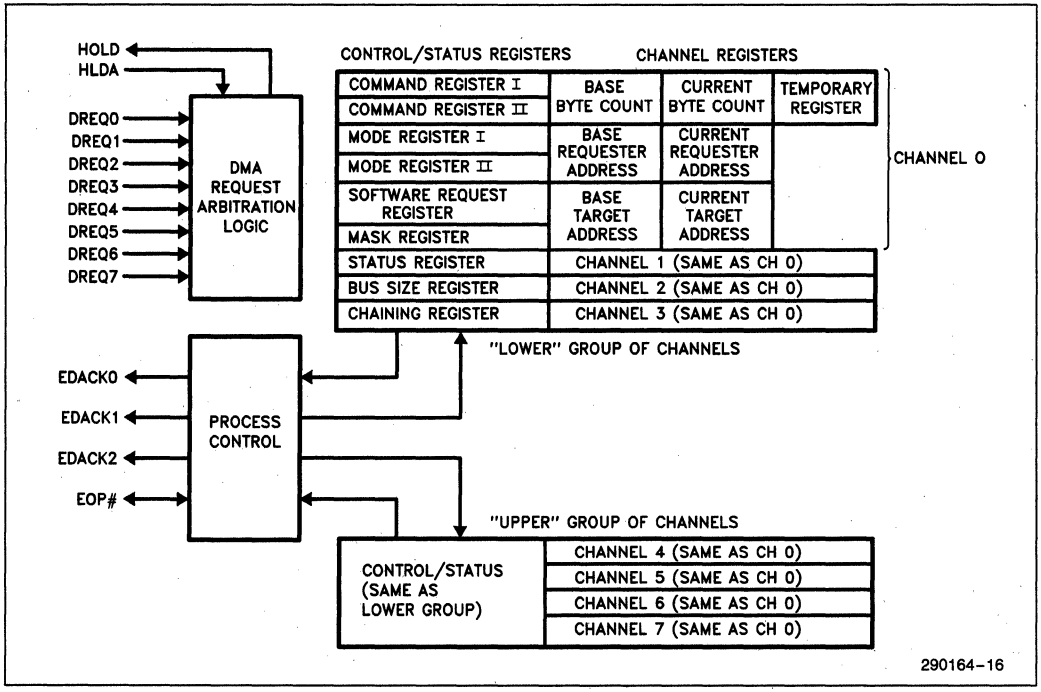


Figure 3-1. 82370 DMA Controller Block Diagram

fore entering the discussion of the function of the 82370 DMA Controller, the following explanations of some of the terminology used herein may be of benefit. First, a few terms for clarification:

DMA PROCESS—A DMA process is the execution of a programmed DMA task from beginning to end. Each DMA process requires intitial programming by the host 80376 microprocessor.

BUFFER—A contiguous block of data.

BUFFER TRANSFER—The action required by the DMA to transfer an entire buffer.

DATA TRANSFER—The DMA action in which a group of bytes or words are moved between devices by the DMA Controller. A data transfer operation may involve movement of one or many bytes.

BUS CYCLE—Access by the DMA to a single byte or word.

Each DMA channel consists of three major components. These components are identified by the contents of programmable registers which define the memory or I/O devices being serviced by the DMA. They are the Target, the Requester, and the Byte Count. They will be defined generically here and in greater detail in the DMA register definition section.

The Requester is the device which requires service by the 82370 DMA Controller, and makes the request for service. All of the control signals which the DMA monitors or generates for specific channels are logically related to the Requester. Only the Requester is considered capable of initiating or terminating a DMA process.

The Target is the device with which the Requester wishes to communicate. As far as the DMA process is concerned, the Target is a slave which is incapable of control over the process.

The direction of data transfer can be either from Requester to Target or from Target to Requester; i.e. each can be either a source or a destination.

The Requester and Target may each be either I/O or memory. Each has an address associated with it that can be incremented, decremented, or held constant. The addresses are stored in the Requester

Address Registers and Target Address Registers, respectively. These registers have two parts: one which contains the current address being used in the DMA process (Current Address Register), and one which holds the programmed base address (Base Address Register). The contents of the Base Registers are never changed by the 82370 DMA Controller. The Current Registers are incremented or decremented according to the progress of the DMA process.

The Byte Count is the component of the DMA process which dictates the amount of data which must be transferred. Current and Base Byte Count Registers are provided. The Current Byte Count Register is decremented once for each byte transferred by the DMA process. When the register is decremented past zero, the Byte Count is considered 'expired' and the process is terminated or restarted, depending on the mode of operation of the channel. The point at which the Byte Count expires is called 'Terminal Count' and several status signals are dependent on this event.

Each channel of the 82370 DMA Controller also contains a 32-bit Temporary Register for use in assembling and disassembling non-aligned data. The operation of this register is transparent to the user, although the contents of it may affect the timing of some DMA handshake sequences. Since there is data storage available for each channel, the DMA Controller can be interrupted without loss of data.

To avoid unexpected results, care should be taken in programming the byte count correctly when assembing and disassembling non-aligned data. For example:

Words to Bytes:
Transferring two words to bytes, but setting the byte count to three, will result in three bytes transferred and the final byte flushed.

Bytes to Words:
Transferring six bytes to three words, but setting the byte count to five, will result in the sixth byte transferred being undefined.

The 82370 DMA Controller is a slave on the bus until a request for DMA service is received via either a software request command or a hardware request signal. The host processor may access any of the control/status or channel registers at any time the 82370 is a bus slave. Figure 3-2 shows the flow of operations that the DMA Controller performs.

At the time a DMA service request is received, the DMA Controller issues a bus hold request to the host processor. The 82370 becomes the bus master when the host relinquishes the bus by asserting a hold acknowledge signal. The channel to be serviced will be the one with the highest priority at the time the DMA Controller becomes the bus master. The DMA Controller will remain in control of the bus until the hold acknowledge signal is removed, or until the current DMA transfer is complete.

While the 82370 DMA Controller has control of the bus, it will perform the required data transfer(s). The type of transfer, source and destination addresses, and amount of data to transfer are programmed in the control registers of the DMA channel which received the request for service.

At completion of the DMA process, the 82370 will remove the bus hold request. At this time the 82370 becomes a slave again, and the host returns to being a master. If there are other DMA channels with requests pending, the controller will again assert the hold request signal and restart the bus arbitration and switching process.

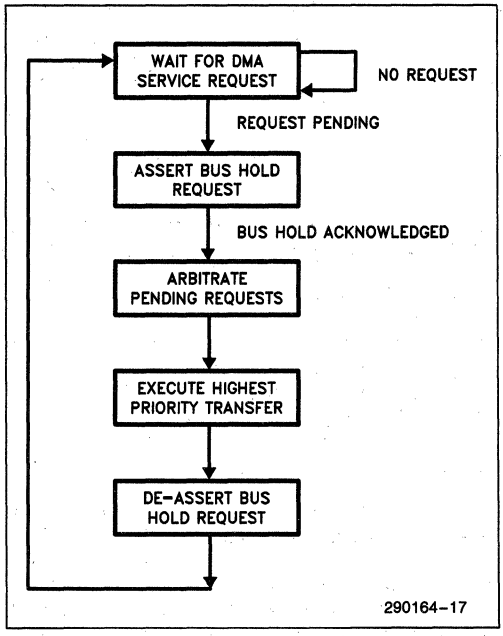


Figure 3-2. Flow of DMA Controller Operation

3.2 Interface Signals

There are fourteen control signals dedicated to the DMA process. They include eight DMA Channel Requests (DREQn), three Encoded DMA Acknowledge signals (EDACKn), Processor Hold and Hold Ac-

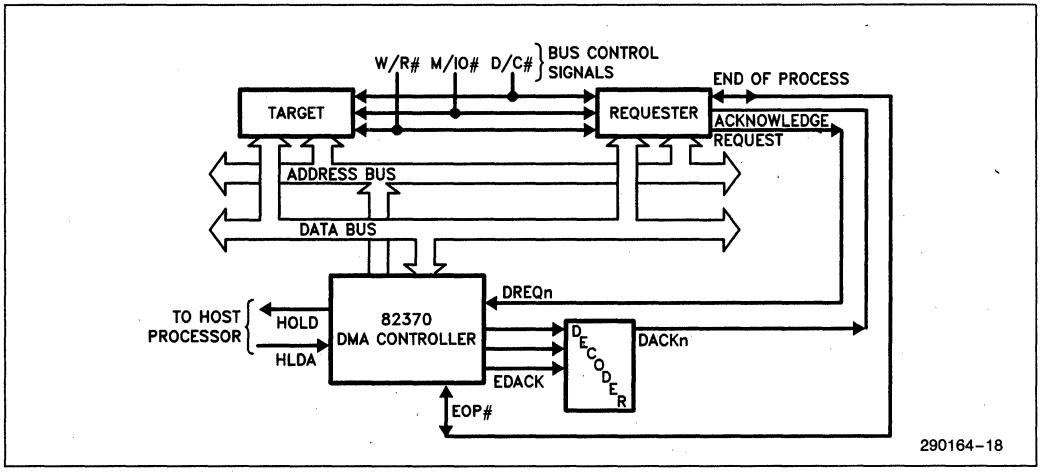


Figure 3-3. Requester, Target and DMA Controller Interconnection

knowledge (HOLD, HLDA), and End-of-Process (EOP#). The DREQn inputs and EDACK (0–2) outputs are handshake signals to the devices requiring DMA service. The HOLD output and HLDA input are handshake signals to the host processor. Figure 3-3 shows these signals and how they interconnect between the 82370 DMA Controller, and the Requester and Target devices.

3.2.1 DREQn and EDACK (0–2)

These signals are the handshake signals between the peripheral and the 82370. When the peripheral requires DMA service, it asserts the DREQn signal of the channel which is programmed to perform the service. The 82370 arbitrates the DREQn against other pending requests and begins the DMA process after finishing other higher priority processes.

When the DMA service for the requested channel is in progress, the EDACK (0–2) signals represent the DMA channel which is accessing the Requester. The 3-bit code on the EDACK (0–2) lines indicates the number of the channel presently being serviced. Table 3-2 shows the encoding of these signals. Note that Channel 4 does not have a corresponding hardware acknowledge.

The DMA acknowledge (EDACK) signals indicate the active channel only during DMA accesses to the Requester. During accesses to the Target, EDACK (0–2) has the idle code (100). EDACK (0–2) can thus be used to select a Requester device during a transfer.

DREQn can be programmed as either an Asynchronous or Synchronous input. See section 3.4.1 for details on synchronous versus asynchronous operation of these pins.

Table 3-2. EDACK Encoding During a DMA Transfer

EDACK2	EDACK1	EDACK0	Active Channel
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	Target Access
1	0	1	5
1	1	0	6
1	1	1	7

The EDACKn signals are always active. They either indicate 'no acknowledge' or they indicate a bus access to the requester. The acknowledge code is either 100, for an idle DMA or during a DMA access to the Target, or 'n' during a Requester access, where n is the binary value representing the channel. A simple 3-line to 8-line decoder can be used to provide discrete acknowledge signals for the peripherals.

3.2.2 HOLD AND HLDA

The Hold Request (HOLD) and Hold Acknowledge (HLDA) signals are the handshake signals between the DMA Controller and the host processor. HOLD is an output from the 82370 and HLDA is an input. HOLD is asserted by the DMA Controller when there is a pending DMA request, thus requesting the processor to give up control of the bus so the DMA process can take place. The 80376 responds by asserting HLDA when it is ready to relinquish control of the bus.

The 82370 will begin operations on the bus one clock cycle after the HLDA signal goes active. For this reason, other devices on the bus should be in the slave mode when HLDA is active.

HOLD and HLDA should not be used to gate or select peripherals requesting DMA service. This is because of the use of DMA-like operations by the DRAM Refresh Controller. The Refresh Controller is arbitrated with the DMA Controller for control of the bus, and refresh cycles have the highest priority. A refresh cycle will take place between DMA cycles without relinquishing bus control. See section 3.4.3 for a more detailed discussion of the interaction between the DMA Controller and the DRAM Refresh Controller.

3.2.3 EOP#

EOP# is a bi-directional signal used to indicate the end of a DMA process. The 82370 activates this as an output during the T2 states of the last Requester bus cycle for which a channel is programmed to execute. The Requester should respond by either withdrawing its DMA request, or interrupting the host processor to indicate that the channel needs to be programmed with a new buffer. As an input, this signal is used to tell the DMA Controller that the peripheral being serviced does not require any more data to be transferred. This indicates that the current buffer is to be terminated.

EOP# can be programmed as either an Asynchronous or a Synchronous input. See section 3.4.1 for details on synchronous versus asynchronous operation of this pin.

3.3 Modes of Operation

The 82370 DMA Controller has many independent operating functions. When designing peripheral interfaces for the 82370 DMA Controller, all of the functions or modes must be considered. All of the channels are independent of each other (except in priority of operation) and can operate in any of the modes. Many of the operating modes, though independently programmable, affect the operation of other modes. Because of the large number of combinations possible, each programmable mode is discussed here with its affects on the operation of other modes. The entire list of possible combinations will not be presented.

Table 3-1 shows the categories of DMA features available in the 82370. Each of the five major categories is independent of the others. The sub-categories are the available modes within the major function or mode category. The following sections explain each mode or function and its relation to other features.

Table 3-1. DMA Operating Modes

I. TARGET/REQUESTER DEFINITION
a. Data Transfer Direction
b. Device Type
II. BUFFER PROCESSES
a. Single Buffer Process
b. Buffer Auto-Initialize Process
c. Buffer Chaining Process
III. DATA TRANSFER/HANDSHAKE MODES
a. Single Transfer Mode
b. Demand Transfer Mode
c. Block Transfer Mode
d. Cascade Mode
IV. PRIORITY ARBITRATION
a. Fixed
b. Rotating
c. Programmable Fixed
V. BUS OPERATION
a. Fly-By (Single-Cycle)/Two-Cycle
b. Data Path Width
c. Read, Write, or Verify Cycles

3.3.1 TARGET/REQUESTER DEFINITION

All DMA transfers involve three devices: the DMA Controller, the Requester, and the Target. Since the devices to be accessed by the DMA Controller vary widely, the operating characteristics of the DMA Controller must be tailored to the Requester and Target devices.

The Requester can be defined as either the source or the destination of the data to be transferred. This is done by specifying a Write or a Read transfer, respectively. In a Read transfer, the Target is the data source and the Requester is the destination for the data. In a Write transfer, the Requester is the source and the Target is the destination.

The Requester and Target addresses can each be independently programmed to be incremented, decremented, or held constant. As an example, the 82370 is capable of reversing a string of data by having the Requester address increment and the Target address decrement in a memory-to-memory transfer.

3.3.2 BUFFER TRANSFER PROCESSES

The 82370 DMA Controller allows three programmable Buffer Transfer Processes. These processes define the logical way in which a buffer of data is accessed by the DMA.

The three Buffer Transfer Processes include the Single Buffer Process, the Buffer Auto-Initialize Process, and the Buffer Chaining Process. These processes require special programming considerations. See the DMA Programming section for more details on setting up the Buffer Transfer Processes.

Single Buffer Process

The Single Buffer Process allows the DMA channel to transfer only one buffer of data. When the buffer has been completely transferred (Current Byte Count decremented past zero or EOP# input active), the DMA process ends and the channel becomes idle. In order for that channel to be used again, it must be reprogrammed.

The Single Buffer Process is usually used when the amount of data to be transferred is known exactly, and it is also known that there is not likely to be any data to follow before the operating system can reprogram the channel.

Buffer Auto-Initialize Process

The Buffer Auto-Initialize Process allows multiple groups of data to be transferred to or from a single buffer. This process does not require reprogramming. The Current Registers are automatically reprogrammed from the Base Registers when the current process is terminated, either by an expired Byte Count or by an external EOP# signal. The data transferred will always be between the same Target and Requester.

The auto-initialization/process-execution cycle is repeated until the channel is either disabled or re-programmed.

Buffer Chaining Process

The Buffer Chaining Process is useful for transferring large quantities of data into non-contiguous buffer areas. In this process, a single channel is used to process data from several buffers, while having to program the channel only once. Each new buffer is programmed in a pipelined operation that provides the new buffer information while the old buffer is being processed. The chain is created by loading new buffer information while the 82370 DMA Controller is processing the Current Buffer. When the Current Buffer expires, the 82370 DMA Controller automatically restarts the channel using the new buffer information.

Loading the new buffer information is done by an interrupt routine which is requested by the 82370. Interrupt Request 1 (IRQ1) is tied internally to the 82370 DMA Controller for this purpose. IRQ1 is generated by the 82370 when the new buffer information is loaded into the channel's Current Registers, leaving the Base Registers 'empty'. The interrupt service routine loads new buffer information into the Base Registers. The host processor is required to load the information for another buffer before the current Byte Count expires. The process repeats until the host programs the channel back to single buffer operation, or until the channel runs out of buffers.

The channel runs out of buffers when the Current Buffer expires and the Base Registers have not yet been loaded with new buffer information. When this occurs, the channel must be reprogrammed.

If an external EOP# is encountered while executing a Buffer Chaining Process, the current buffer is considered expired and the new buffer information is loaded into the Current Registers. If the Base Registers are 'empty', the chain is terminated.

The channel uses the Base Target Address Register as an indicator of whether or not the Base Registers are full. When the most significant byte of the Base Target Register is loaded, the channel considers all of the Base Registers loaded, and removes the interrupt request. This requires that the other Base Registers (Base Requester Address, Base Byte Count) must be loaded before the Base Target Address Register. The reason for implementing the reloading process this way is that, for most applications, the Byte Count and the Requester will not change from one buffer to the next, and therefore do not need to be reprogrammed. The details of programming the channel for the Buffer Chaining Process can be found in the section on DMA programming.

3.3.3 DATA TRANSFER MODES

Three Data Transfer modes are available in the 82370 DMA Controller. They are the Single Transfer, Block Transfer, and Demand Transfer Modes. These transfer modes can be used in conjunction with any one of three Buffer Transfer modes: Single Buffer, Auto-Initialized Buffer and Buffer Chaining. Any Data Transfer Mode can be used under any of the Buffer Transfer Modes. These modes are independently available for all DMA channels.

Different devices being serviced by the DMA Controller require different handshaking sequences for data transfers to take place. Three handshaking modes are available on the 82370, giving the designer the opportunity to use the DMA Controller as efficiently as possible. The speed at which data can

be presented or read by a device can affect the way a DMA Controller uses the host's bus, thereby affecting not only data throughput during the DMA process, but also affecting the host's performance by limiting its access to the bus.

Single Transfer Mode

In the Single Transfer Mode, one data transfer to or from the Requester is performed by the DMA Controller at a time. The DREQn input is arbitrated and the HOLD/HLDA sequence is executed for each transfer. Transfers continue in this manner until the Byte Count expires, or until EOP# is sampled active. If the DREQn input is held active continuously, the entire DREQ-HOLD-HLDA-DACK sequence is repeated over and over until the programmed number of bytes has been transferred. Bus control is released to the host between each transfer. Figure 3-4 shows the logical flow of events which make up a buffer transfer using the Single Transfer Mode. Refer to section 3.4 for an explanation of the bus control arbitration procedure.

The Single Transfer Mode is used for devices which require complete handshake cycles with each data access. Data is transferred to or from the Requester only when the Requester is ready to perform the transfer. Each transfer requires the entire DREQ-HOLD-HLDA-DACK handshake cycle. Figure 3-5 shows the timing of the Single Transfer Mode cycle.

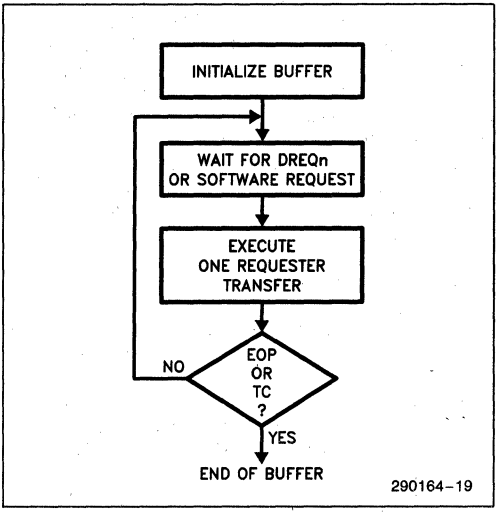
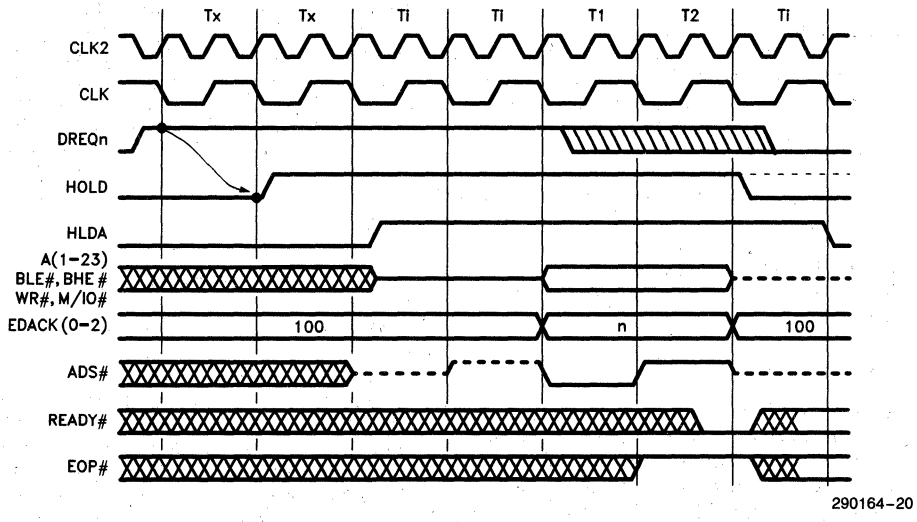


Figure 3-4. Buffer Transfer in Single Transfer Mode



NOTE:
The Single Transfer Mode is more efficient (15%–20%) in the case where the source is the Target. Because of the internal pipeline of the 82370 DMA Controller, two idle states are added at the end of a transfer in the case where the source is the Requester.

Figure 3-5. DMA Single Transfer Mode

Block Transfer Mode

In the Block Transfer Mode, the DMA process is initiated by a DMA request and continues unti the Byte Count expires, or until EOP# is activated by the Requester. The DREQn signal need only be held active until the first Requester access. Only a refresh cycle will interrupt the block transfer process.

Figure 3-6 illustrates the operation of the DMA during the Block Transfer Mode. Figure 3-7 shows the timing of the handshake signals during Block Mode Transfers.

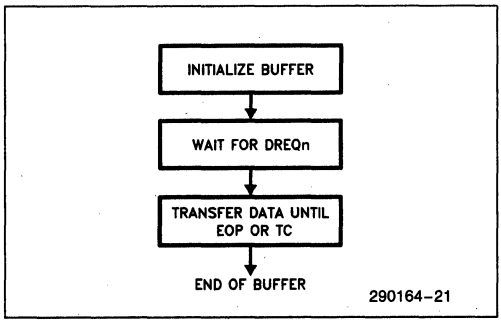


Figure 3-6. Buffer Transfer in Block Transfer Mode

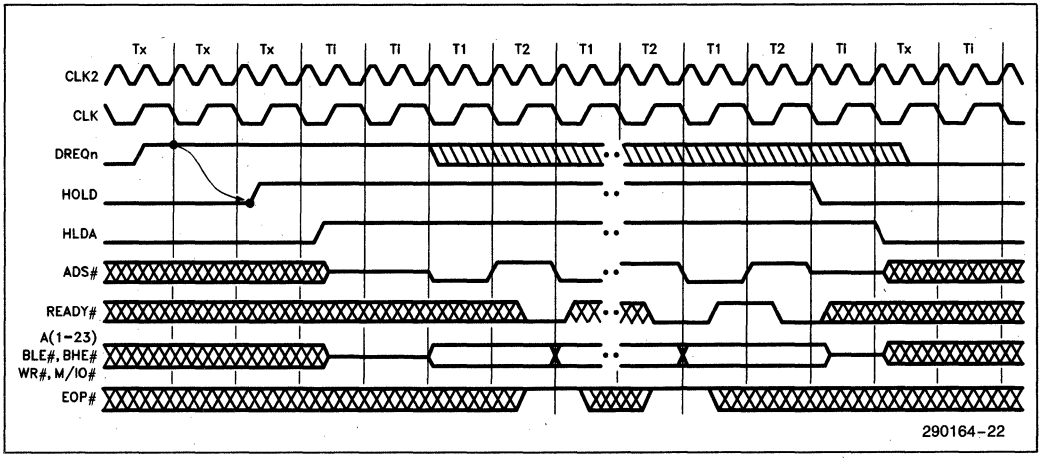


Figure 3-7. Block Mode Transfers

Demand Transfer Mode

The Demand Transfer Mode provides the most flexible handshaking procedures during the DMA process. A Demand Transfer is initiated by a DMA request. The process continues until the Byte Count expires, or an external EOP# is encountered. If the device being serviced (Requester) desires, it can interrupt the DMA process by de-activating the DREQn line. Action is taken on the condition of DREQn during Requester accesses only. The access during which DREQn is sampled inactive is the last Requester access which will be performed during the current transfer. Figure 3-8 shows the flow of events during the transfer of a buffer in the Demand Mode.

When the DREQn line goes inactive, the DMA Controller will complete the current transfer, including any necessary accesses to the Target, and relinquish control of the bus to the host. The current process information is saved (byte count, Requester and Target addresses, and Temporary Register).

The Requester can restart the transfer process by reasserting DREQn. The 82370 will arbitrate the request with other pending requests and begin the process where it left off. Figure 3-9 shows the timing of handshake signals during Demand Transfer Mode operation.

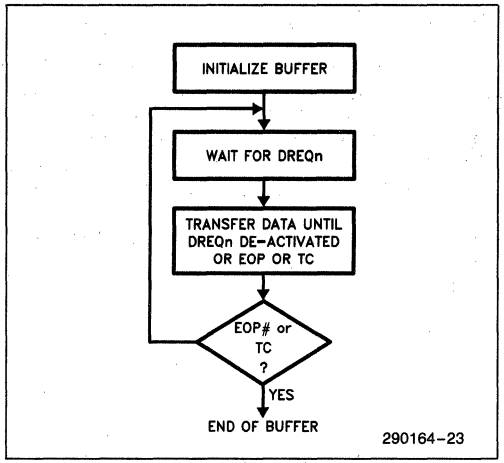


Figure 3-8. Buffer Transfer in Demand Transfer Mode

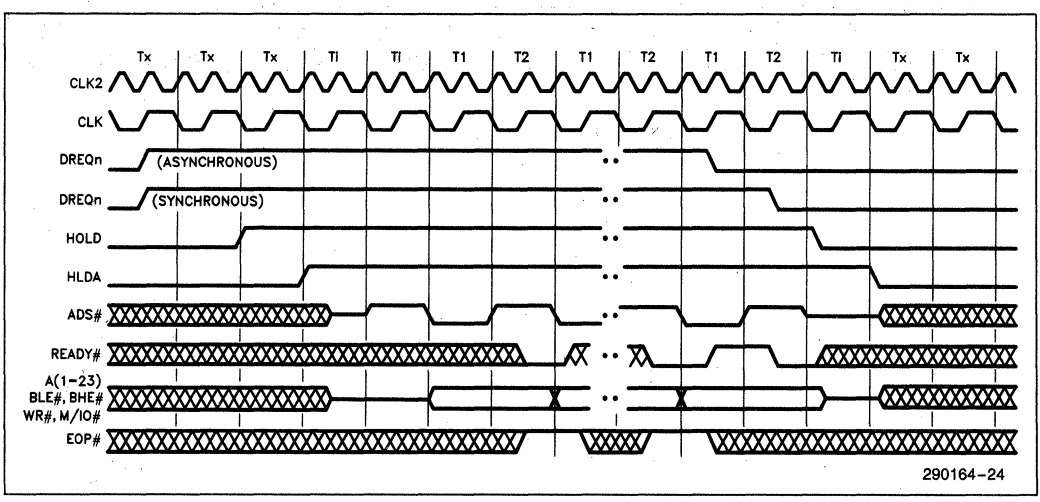


Figure 3-9. Demand Mode Transfers

Using the Demand Transfer Mode allows peripherals to access memory in small, irregular bursts without wasting bus control time. The 82370 is designed to give the best possible bus control latency in the Demand Transfer Mode. Bus control latency is defined here as the time form the last active bus cycle of the previous bus master to the first active bus cycle of the new bus master. The 82370 DMA Controller will perform its first bus access cycle two bus states after HLDA goes active. In the typical configuration, bus control is returned to the host one bus state after the DREQn goes inactive.

There are two cases where there may be more than one bus state of bus control latency at the end of a transfer. The first is at the end of an Auto-Initialize process, and the second is at the end of a process where the source is the Requester and Two-Cycle transfers are used.

When a Buffer Auto-Initialize Porcess is complete, the 82370 requires seven bus states to reload the Current Registers from the Base Registers of the Auto-Initialized channel. The reloading is done while the 82370 is still the bus master so that it is prepared to service the channel immediately after relinquishing the bus, if necessary.

In the case where the Requester is the source, and Two-Cycle transfers are being used, there are two extra idle states at the end of the transfer process. This occurs due to the housekeeping in the DMA's internal pipeline. These two idle states are present only after the very last Requester access, before the DMA Controller de-activates the HOLD signal.

3.3.4 CHANNEL PRIORITY ARBITRATION

DMA channel priority can be programmed into one of two arbitration methods: Fixed or Rotating. The four lower DMA channels and the four upper DMA channels operate as if they were two separate DMA controllers operating in cascade. The lower group of four channels (0–3) is always prioritized between channels 7 and 4 of the upper group of channels (4–7). Figure 3-10 shows a pictorial representation of the priority grouping.

The priority can thus be set up as rotating for one group of channels and fixed for the other, or any other combination. While in Fixed Priority, the programmer can also specify which channel has the lowest priority.

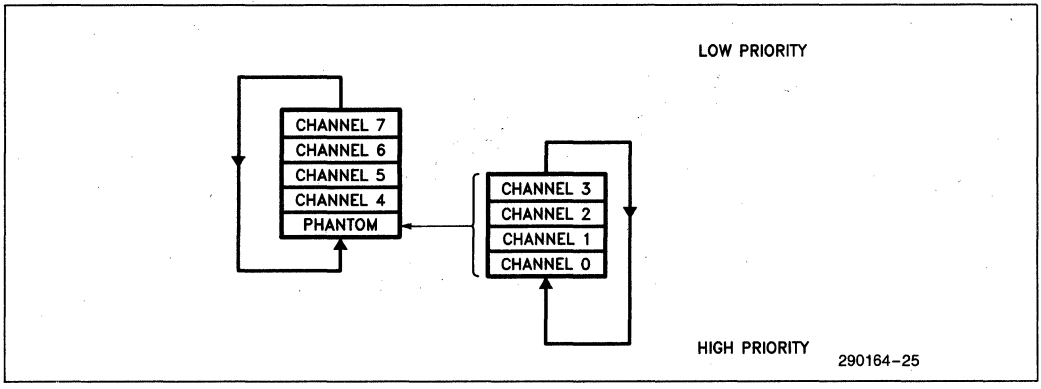


Figure 3-10. DMA Priority Grouping

The 82370 DMA Controller defaults to Fixed Priority. Channel 0 has the highest priority, then 1, 2, 3, 4, 5, 6, 7. Channel 7 has the lowest priority. Any time the DMA Controller arbitrates DMA requests, the requesting channel with the highest priority will be serviced next.

Fixed Priority can be entered into at any time by a software command. The priority levels in effect after the mode switch are determined by the current setting of the Programmable Priority.

Programmable Priority is available for fixing the priority of the DMA channels within a group to levels other than the default. Through a software command, the channel to have the lowest priority in a group can be specified. Each of the two groups of four channels can have the priority fixed in this way. The other channels in the group will follow the natural Fixed Priority sequence. This mode affects only the priority levels while operating with Fixed Priority.

For example, if channel 2 is programmed to have the lowest priority in its group, channel 3 has the highest priority. In descending order, the other channels would have the following priority: (3,0,1,2),4,5,6,7 (channel 2 lowest, channel 3 highest). If the upper group were programmed to have channel 5 as the lowest priority channel, the priority would be (again, highest to lowest): 6,7, (3,0,1,2), 4,5. Figure 3-11 shows this example pictorially. The lower group is always prioritized as a fifth channel of the upper group (between channels 4 and 7).

The DMA Controller will only accept Programmable Priority commands while the addressed group is operating in Fixed Priority. Switching from Fixed to Rotating Priority preserves the current priority levels. Switching from Rotating to Fixed Priority returns the priority levels to those which were last programmed by use of Programmable Priority.

Rotating Priority allows the devices using DMA to share the system bus more evenly. An individual channel does not retain highest priority after being serviced, priority is passed to the next highest priority channel in the group. The channel which was most recently serviced inherits the lowest priority. This rotation occurs each time a channel is serviced. Figure 3-12 shows the sequence of events as priority is passed between channels. Note that the lower group rotates within the upper group, and that servicing a channel within the lower group causes rotation within the group as well as rotation of the upper group.

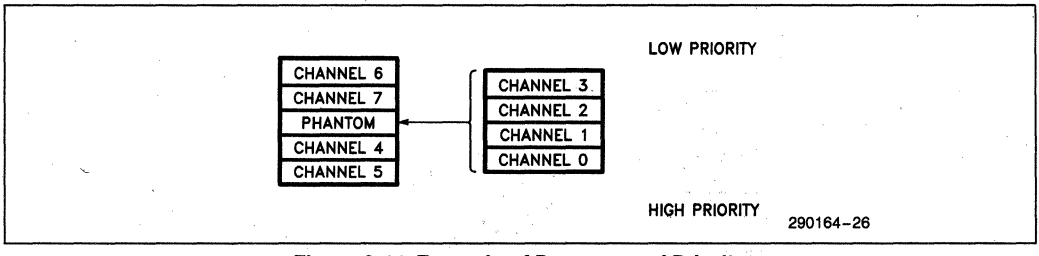


Figure 3-11. Example of Programmed Priority

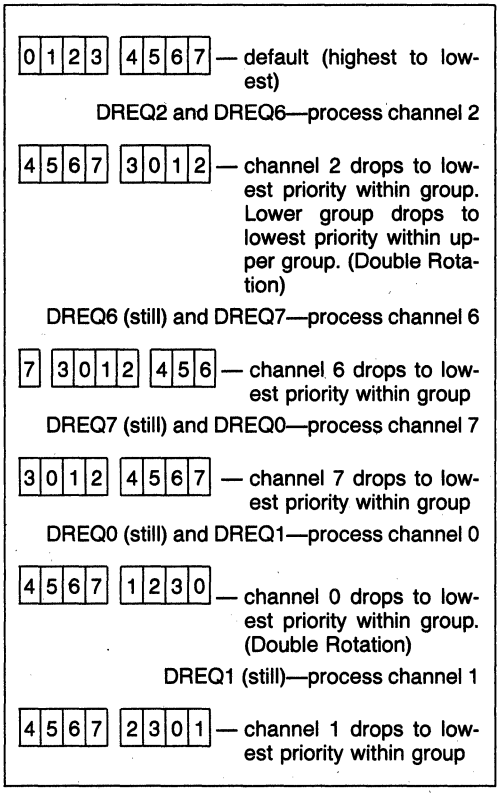


Figure 3-12. Rotating Channel Priority. Lower and upper groups are programmed for the Rotating Priority Mode.

3.3.5 COMBINING PRIORITY MODES

Since the DMA Controller operates as two four-channel controllers in cascade, the overall priority scheme of all eight channels can take on a variety of forms. There are four possible combinations of priority modes between the two groups of channels: Fixed Priority only (default), Fixed Priority upper group/Rotating Priority lower group, Rotating Priority upper group/Fixed Priority lower group, and Rotating Priority only. Figure 3-13 illustrates the operation of the two combined priority methods.

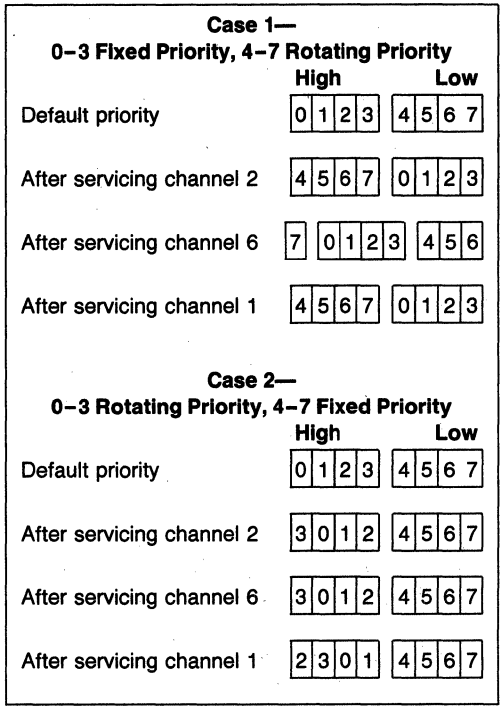


Figure 3-13. Combining Priority Modes

3.3.6 BUS OPERATION

Data may be transferred by the DMA Controller using two different bus cycle operations: Fly-By (one-cycle) and Two-Cycle. These bus handshake methods are selectable independently for each channel through a command register. Device data path widths are independently programmable for both Target and Requester. Also selectable through software is the direction of data transfer. All of these parameters affect the operation of the 82370 on a bus-cycle by bus-cycle basis.

3.3.6.1 Fly-By Transfers

The Fly-By Transfer Mode is the fastest and most efficient way to use the 82370 DMA Controller to transfer data. In this method of transfer, the data is written to the destination device at the same time it is read from the source. Only one bus cycle is used to accomplish the transfer.

In the Fly-By Mode, the DMA acknowledge signal is used to select the Requester. The DMA Controller simultaneously places the address of the Target on the address bus. The state of M/IO# and W/R# during the Fly-By transfer cycle indicate the type of Target and whether the Target is being written to or read from. The Target's Bus Size is used as an incrementer for the Byte Count. The Requester address registers are ignored during Fly-By transfers.

Note that memory-to-memory transfers cannot be done using the Fly-By Mode. Only one memory of I/O address is generated by the DMA Controller at a time during Fly-By transfers. Only one of the devices being accessed can be selected by an address. Also, the Fly-By method of data transfer limits the hardware to accesses of devices with the same data bus width. The Temporary Registers are not affected in the Fly-By Mode.

Fly-By transfers also require that the data paths of the Target and Requester be directly connected. This requires that successive Fly-By access be to word boundaries, or that the Requester be capable of switching its connections to the data bus.

3.3.6.2. Two-Cycle Transfers

Two-Cycle transfers can also be performed by the 82370 DMA Controller. These transfers require at least two bus cycles to execute. The data being transferred is read into the DMA Controller's Temporary Register during the first bus cycle(s). The second bus cycle is used to write the data from the Temporary Register to the destination.

If the addresses of the data being transferred are not word aligned, the 82370 will recognize the situation and read and write the data in groups of bytes, placing them always at the proper destination. This process of collecting the desired bytes and putting them together is called "byte assembly". The reverse process (reading from aligned locations and writing to non-aligned locations) is called "byte disassembly".

The assembly/disassembly process takes place transparent to the software, but can only be done while using the Two-Cycle transfer method. The 82370 will always perform the assembly/disassembly process as necessary for the current data transfer. Any data path widths for either the Requester or Target can be used in the Two-Cycle Mode. This is very convenient for interfacing existing 8- and 16-bit peripherals to the 80376's 16-bit bus.

The 82370 DMA Controller always reads and write data within the word boundaries; i.e. if a word to be read is crossing a word boundary, the DMA Controller will perform two read operations, each reading one byte, to read the 16-bit word into the Temporary Register. Also, the 82370 DMA Controller always attempts to fill the Temporary Register from the source before writing any data to the destination. If the process is terminated before the Temporary Register is filled (TC or EOP#), the 82370 will write the partial data to the destination. If a process is temporarily suspended (such as when DREQn is deactivated during a demand transfer), the contents of a partially filled Temporary Register will be stored within the 82370 until the process is restarted.

For example, if the source is specified as an 8-bit device and the destination as a 32-bit device, there will be four reads as necessary from the 8-bit source to fill the Temporary Register. Then the 82370 will write the 32-bit contents to the destination in two cycles of 16-bit each. This cycle will repeat until the process is terminated or suspended.

With Two-Cycle transfers, the devices that the 82370 accesses can reside at any address within I/O or memory space. The device must be able to decode the byte-enables (BLE#, BHE#). Also, if the device cannot accept data in byte quantities, the programmer must take care not to allow the DMA Controller to access the device on any address other than the device boundary.

3.3.6.3 Data Path Width and Data Transfer Rate Considerations

The number of bus cycles used to transfer a single "word" of data is affected by whether the Two-Cycle or the Fly-By (Single-Cycle) transfer method is used.

The number of bus cycles used to transfer data directly affects the data transfer rate. Inefficient use of bus cycles will decrease the effective data transfer rate that can be obtained. Generally, the data transfer rate is halved by using Two-Cycle transfers instead of Fly-By transfers.

The choice of data path widths of both Target and Requester affects the data transfer rate also. During each bus cycle, the largest pieces of data possible should be transferred.

The data path width of the devices to be accessed must be programmed into the DMA controller. The 82370 defaults after reset to 8-bit-to-8-bit data transfers, but the Target and Requester can have different data path widths, independent of each other and independent of the other channels. Since this is a software programmable function, more discussion of the uses of this feature are found in the section on programming.

3.3.6.4 Read, Write and Verify Cycles

Three different bus cycles types may be used in a data transfer. They are the Read, Write and Verify cycles. These cycle types dictate the way in which the 82370 operates on the data to be transferred.

A Read Cycle transfers data from the Target to the Requester. A Write Cycle transfers data from the Requester to the target. In a Fly-By transfer, the address and bus status signals indicate the access (read of write) to the Target; the access to the Requester is assumed to be the opposite.

The Verify Cycle is used to perform a data read only. No write access is indicated or assumed in a Verify Cycle. The Verify Cycle is useful for validating block fill operations. An external comparator must be provided to do any comparisons on the data read.

3.4 Bus Arbitration and Handshaking

Figure 3-14 shows the flow of events in the DMA request arbitration process. The arbitration sequence starts when the Requester asserts a DREQn (or DMA service is requested by software). Figure 3-15 shows the timing of the sequence of events following a DMA request. This sequence is executed for each channel that is activated. The DREQn signal can be replaced by a software DMA channel request with no change in the sequence.

After the Requester asserts the service request, the 82370 will request control of the bus via the HOLD signal. The 82370 will always assert the HOLD signal one bus state after the service request is asserted. The 80376 responds by asserting the HLDA signal, thus releasing control of the bus to the 82370 DMA Controller.

Priority of pending DMA service requests is arbitrated during the first state after HLDA is asserted by the 80376. The next state will be the beginning of the first transfer access of the highest priority process.

When the 82370 DMA Controller is finished with its current bus activity, it returns control of the bus to the host processor. This is done by driving the HOLD signal inactive. The 82370 does not drive any address or data bus signals after HOLD goes low. It enters the Slave Mode until another DMA process is requested. The processor acknowledges that it has regained control of the bus by forcing the HLDA signal inactive. Note that the 82370's DMA Controller will not re-request control of the bus until the entire HOLD/HLDA handshake sequence is complete.

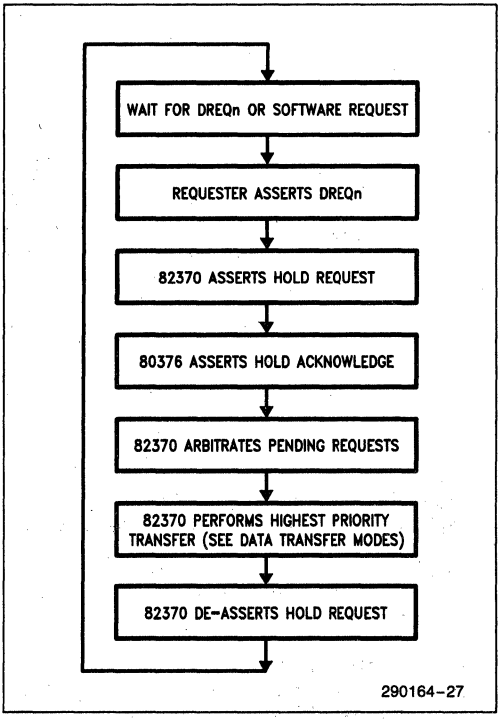
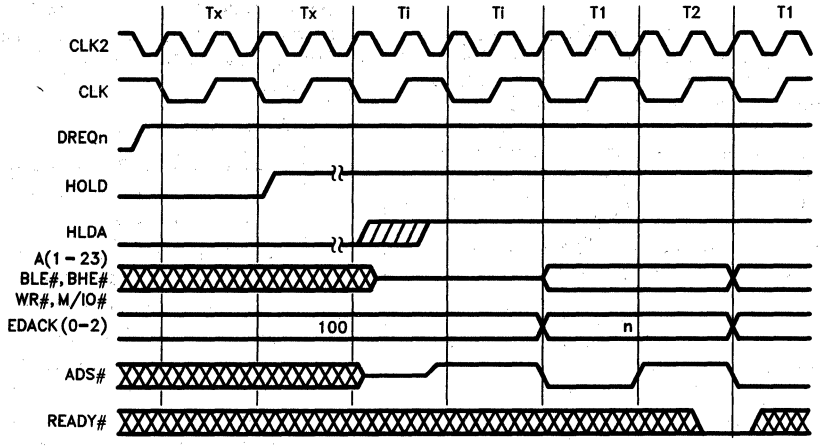


Figure 3-14. Bus Arbitration and DMA Sequence

The 82370 DMA Controller will terminate a current DMA process for one of three reasons: expired byte count, end-of-process command (EOP# activated) from a peripheral, or deactivated DMA request signal. In each case, the controller will de-assert HOLD immediately after completing the data transfer in progress. These three methods of process termination are illustrated in Figures 3-16, 3-19 and 3-18, respectively.

An expired byte count indicates that the current process is complete as programmed and the channel has no further transfers to process. The channel must be restarted according to the currently programmed Buffer Transfer Mode, or reprogrammed completely, including a new Buffer Transfer Mode.



NOTE:
290164-28
Channel priority resolution takes place during the bus state before HOLDA is asserted, allowing the DMA Controller to respond to HLDA without extra idle bus states.

Figure 3-15. Beginning of a DMA process

If the peripheral activates the EOP# signal, it is indicating that it will not accept or deliver any more data for the current buffer. The 82370 DMA Controller considers this as a completion of the channel's current process and interprets the condition the same way as if the byte count expired.

The action taken by the 82370 DMA Controller in response to a de-activated DREQn signal depends on the Data Transfer Mode of the channel. In the Demand Mode, data transfers will take place as long as the DREQn is active and the byte count has not expired. In the Block Mode, the controller will complete the entire block transfer without relinquishing the bus, even if DREQn goes inactive before the transfer is complete. In the Single Mode, the controller will execute single data transfers, relinquishing the bus between each transfer, as long as DREQn is active.

Normal termination of a DMA process due to expiration of the byte count (Terminal Count—TC) is shown if Figure 3-16. The condition of DREQn is ignored until after the process is terminated. If the channel is programmed to auto-initialize, HOLD will be held active for an additional seven clock cycles while the auto-initialization takes place.

Table 3-3 shows the DMA channel activity due to EOP# or Byte Count expiring (Terminal Count).

Table 3-3. DMA Channel Activity Due to Terminal Count or External EOP#

Buffer Process	Single or Chaining-Base Empty		Auto-Initialize		Chaining-Base Loaded	
EVENT						
Terminal Count	True	X	True	X	True	X
EOP#	X	0	X	0	X	0
RESULTS						
Current Registers			Load	Load	Load	Load
Channel Mask	Set	Set				
EOP# Output	0	X	0	X	1	X
Terminal Count Status	Set	Set	Set	Set		
Software Request	CLR	CLR	CLR	CLR		

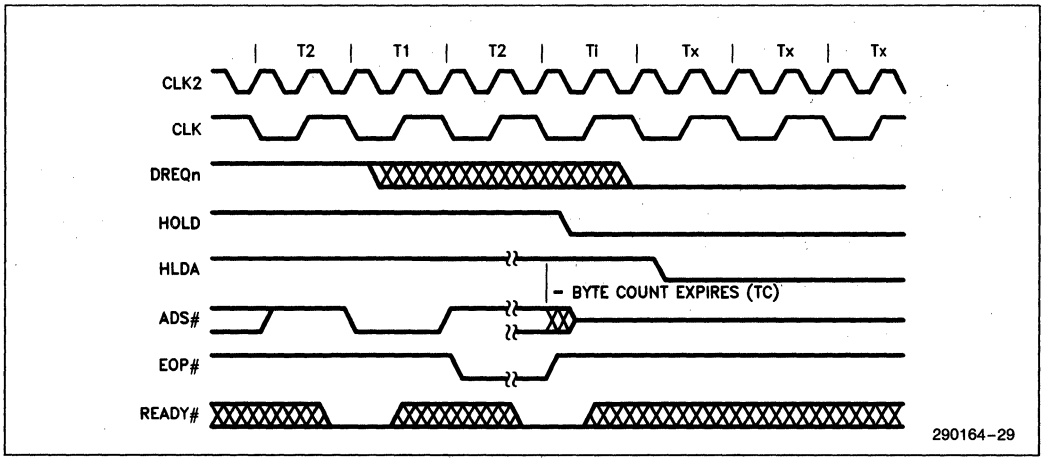


Figure 3-16. Termination of a DMA Process Due to Expiration of Current Byte Count

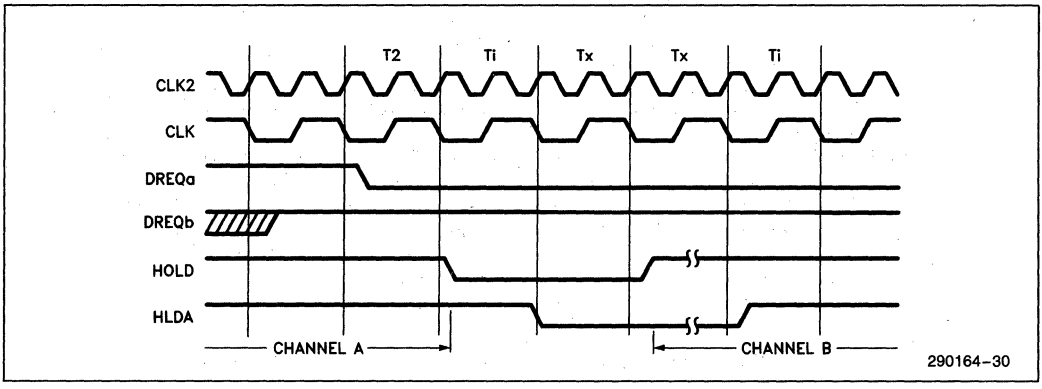


Figure 3-17. Switching between Active DMA Channels

The 82370 always relinquishes control of the bus between channel services. This allows the hardware designer the flexibility to externally arbitrate bus hold requests, if desired. If another DMA request is pending when a higher priority channel service is completed, the 82370 will relinquish the bus until the hold acknowledge is inactive. One bus state after the HLDA signal goes inactive, the 82370 will assert HOLD again. This is illustrated in Figure 3-17.

3.4.1 SYNCHRONOUS AND ASYNCHRONOUS SAMPLING OF DREQn AND EOP#

As an indicator that a DMA service is to be started, DREQn is always sampled asynchronous. It is sampled at the beginning of a bus state and acted upon at the end of the state. Figure 3-15 illustrates the start of a DMA process due to a DREQn input.

The DREQn and EOP# inputs can be programmed to be sampled either synchronously or asynchronously to signal the end of a transfer.

The synchronous mode affords the Requester one bus state of extra time to react to an access. This means the Requester can terminate a process on the current access, without losing any data. The asynchronous mode requires that the input signal be presented prior to the beginning of the last state of the Requester access.

The timing relationships of the DREQn and EOP# signals to the termination of a DMA transfer are shown in Figures 3-18 and 3-19. Figure 3-18 shows the termination of a DMA transfer due to inactive DREQn. Figure 3-19 shows the termination of a DMA process due to an active EOP# input.

In the Synchronous Mode, DREQn and EOP# are sampled at the end of the last state of every Requester data transfer cycle. If EOP# is active or DREQn is inactive at this time, the 82370 recognizes this access to the Requester as the last transfer. At this point, the 82370 completes the transfer in progress, if necessary, and returns bus control to the host.

In the asynchronous mode, the inputs are sampled at the beginning of every state of a Requester access. The 82370 waits until the end of the state to act on the input.

DREQn and EOP# are sampled at the latest possible time when the 82370 can determine if another transfer is required. In the Synchronous Mode, DREQn and EOP# are sampled on the trailing edge of the last bus state before another data access cycle begins. The Asynchronous Mode requires that the signals be valid one clock cycle earlier.

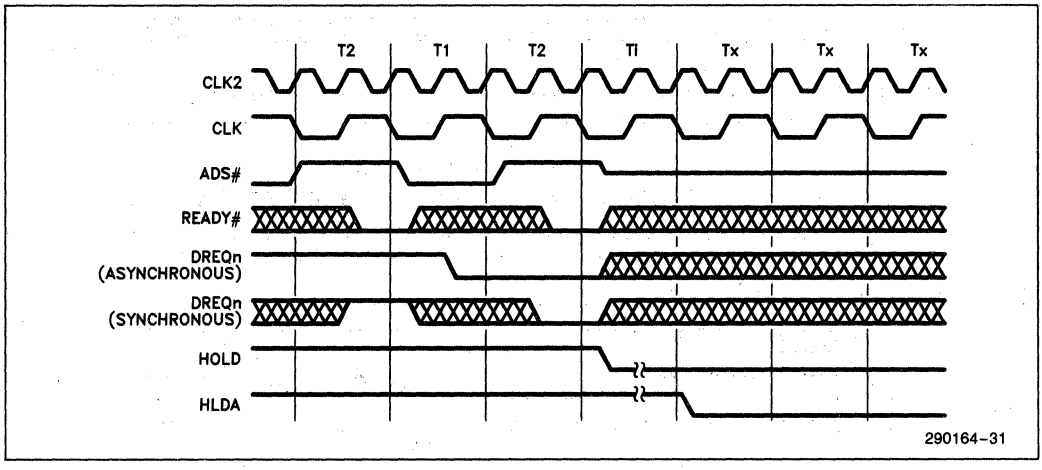


Figure 3-18. Termination of a DMA Process due to De-Asserting DREQn

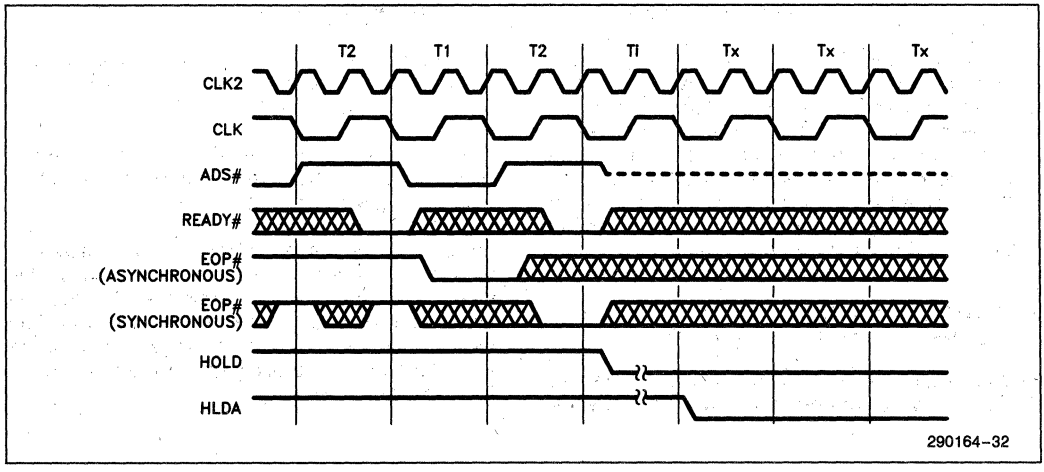


Figure 3-19. Termination of a DMA Process due to an External EOP#

While in the Pipeline Mode, if the NA# signal is sampled active during a transfer, the end of the state where NA# was sampled active is when the 82370 decides whether to commit to another transfer. The device must de-assert DREQn or assert EOP# before NA# is asserted, otherwise the 82370 will commit to another, possibly undesired, transfer.

Synchronous DREQn and EOP# sampling allows the peripheral to prevent the next transfer from occurring by de-activating DREQn or asserting EOP# during the current Requester access, before the 82370 DMA Controller commits itself to another transfer. The DMA Controller will not perform the next transfer if it has not already begun the bus cycle. Asynchronous sampling allows less stringent timing requirements than the Synchronous Mode, but requires that the DREQn signal be valid at the beginning of the next to last bus state of the current Requester access.

Using the Asynchronous Mode with zero wait states can be very difficult. Since the addresses and control signals are driven by the 82370 near half-way through the first bus state of a transfer, and the Asynchronous Mode requires that DREQn be inactive before the end of the state, the peripheral being accessed is required to present DREQn only a few nanoseconds after the control information is available. This means that the peripheral's control logic must be extremely fast (practically non-causal). An alternative is the Synchronous Mode.

3.4.2 ARBITRATION OF CASCADED MASTER REQUESTS

The Cascade Mode allows another DMA-type device to share the bus by arbitrating its bus accesses with the 82370's. Seven of the eight DMA channels (0–3 and 5–7) can be connected to a cascaded device. The cascaded device requests bus control through the DREQn line of the channel which is programmed to operate in Cascade Mode. Bus hold acknowledge is signalled to the cascaded device through the EDACK lines. When the EDACK lines are active with the code for the requested cascade channel, the bus is available to the cascaded master device.

A cascade cycle begins the same way a regular DMA cycle begins. The requesting bus master asserts the DREQn line on the 82370. This bus control request is arbitrated as any other DMA request would be. If any channel receives a DMA request, the 82370 requests control of the bus. When the host acknowledges that it has released bus control, the 82370 acknowledges to the requesting master that it may access the bus. The 82370 enters an idle state until the new master relinquishes control.

A cascade cycle will be terminated by one of two events: DREQn going inactive, or HLDA going inactive. The normal way to terminate the cascade cycle

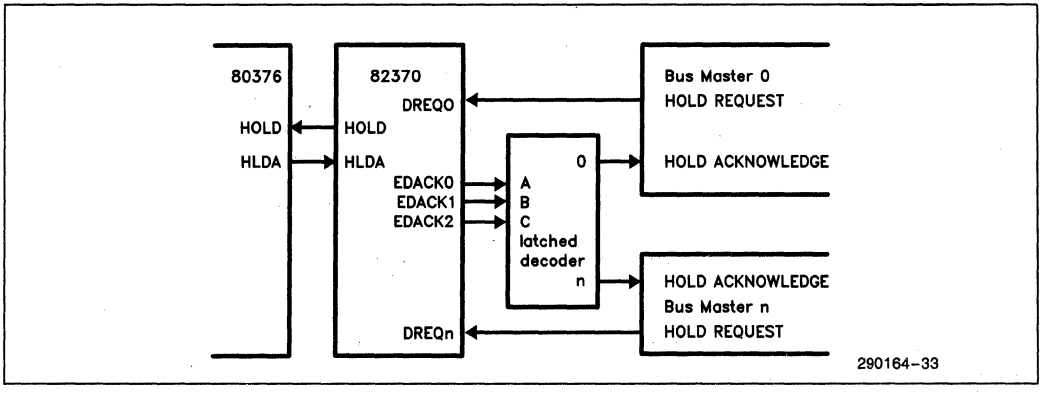


Figure 3-20. Cascaded Bus Master

is for the cascaded master to drop the DREQn signal. Figure 3-21 shows the two cascade cycle termination sequences.

The Refresh Controller may interrupt the cascaded master to perform a refresh cycle. If this occurs, the 82370 DMA Controller will de-assert the EDACK signal (hold acknowledge to cascaded master) and wait for the cascaded master to remove its hold request. When the 82370 regains bus control, it will perform the refresh cycle in its normal fashion. After the refresh cycle has been completed, and if the cascaded device has re-asserted its request, the 82370 will return control to the cascaded master which was interrupted.

The 82370 assumes that it is the only device monitoring the HLDA signal. If the system designer wishes to place other devices on the bus as bus masters, the HLDA from the processor must be intercepted before presenting it to the 82370. Using the Cascade capabililty of the 82370 DMA Controller offers a much better solution.

3.4.3 ARBITRATION OF REFRESH REQUESTS

The arbitration of refresh requests by the DRAM Refresh Controller is slightly different from normal DMA channel request arbitration. The 82370 DRAM Refresh Controller always has the highest priority of any DMA process. It also can interrupt a process in progress. Two types of processes in progress may be encountered: normal DMA, and bus master cascade.

In the event of a refresh request during a normal DMA process, the DMA Controller will complete the data transfer in progress and then execute the refresh cycle before continuing with the current DMA process. The priority of the interrupted process is not lost. If the data transfer cycle interrupted by the Refresh Controller is the last of a DMA process, the refresh cycle will always be executed before control of the bus is transferred back to the host.

When the Refresh Controller request occurs during a cascade cycle, the Refresh Controller must be assured that the cascaded master device has relinquished control of the bus before it can execute the refresh cycle. To do this, the DMA Controller drops the EDACK signal to the cascaded master and waits for the corresponding DREQn input to go inactive. By dropping the DREQn signal, the cascaded master relinquishes the bus. The Refresh Controller then performs the refresh cycle. Control of the bus is returned to the cascaded master if DREQn returns to an active state before the end of the refresh cycle, otherwise control is passed to the processor and the cascaded master loses its priority.

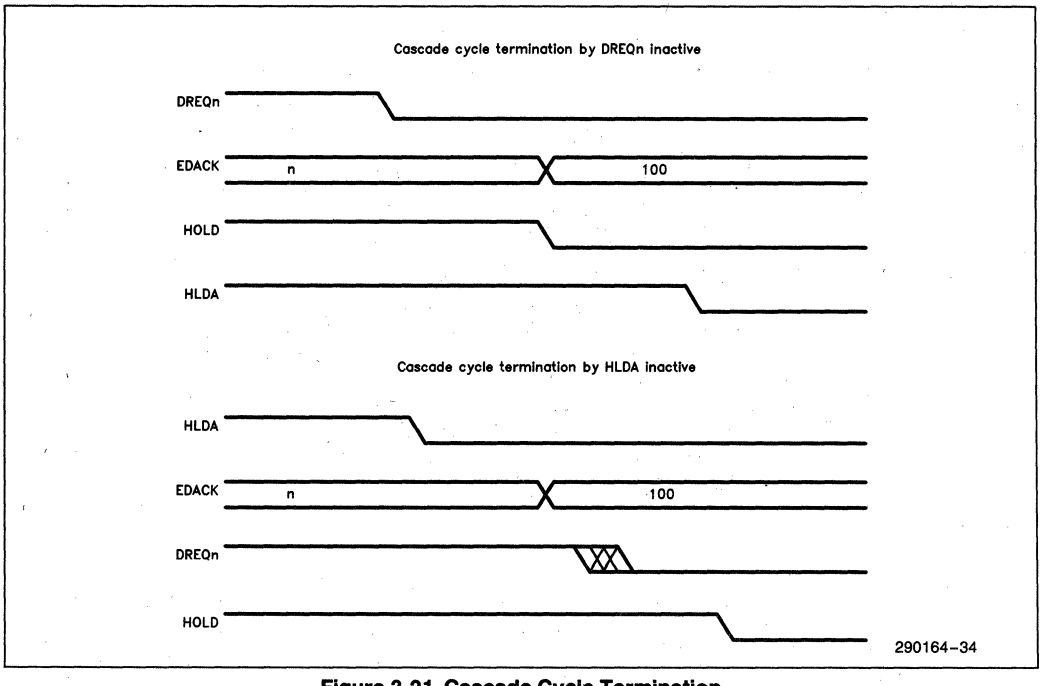


Figure 3-21. Cascade Cycle Termination

3.5 DMA Controller Register Overview

The 82370 DMA Controller contains 44 registers which are accessable to the host processor. Twenty-four of these registers contain the device addresses and data counts for the individual DMA channels (three per channel). The remaining registers are control and status registers for initiating and monitoring the operation of the 82370 DMA Controller. Table 3-4 lists the DMA Controller's registers and their accessability.

Table 3-4. DMA Controller Registers

Register Name	Access
Control/Status Registers—one each per group	
Command Register I	write only
Command Register II	write only
Mode Register I	write only
Mode Register II	write only
Software Request Register	read/write
Mask Set-Reset Register	write only
Mask Read-Write Register	read/write
Status Register	read only
Bus Size Register	write only
Chaining Register	read/write
Channel Registers—one each per channel	
Base Target Address	write only
Current Target Address	read only
Base Requester Address	write only
Current Requester Address	read only
Base Byte Count	write only
Current Byte Count	read only

3.5.1 CONTROL/STATUS REGISTERS

The following registers are available to the host processor for programming the 82370 DMA Controller into its various modes and for checking the operating status of the DMA processes. Each set of four DMA channels has one of each of these registers associated with it.

Command Register I

Enables or disables the DMA channel as a group. Sets the Priority Mode (Fixed or Rotating) of the group. This write-only register is cleared by a hardware reset, defaulting to all channels enabled and Fixed Priority Mode.

Command Register II

Sets the sampling mode of the DREQn and EOP# inputs. Also sets the lowest priority channel for the group in the Fixed Priority Mode. The functions programmed through Command Register II default after a hardware reset to: asynchronous DREQn and EOP#, and channels 3 and 7 lowest priority.

Mode Registers I

Mode Register I is identical in function to the Mode register of the 8237A. It programs the following functions for an individually selected channel:

Type of Transfer—read, write, verify
Auto-Initialize—enable or disable
Target Address Count—increment or decrement
Data Transfer Mode—demand, single, block, cascade

Mode Register I functions default to the following after reset: verify transfer, Auto-Initialize disabled, Increment Target address, Demand Mode.

Mode Register II

Programs the following functions for an individually selected channel:

Target Address Hold—enable or disable
Requester Address Count—increment or decrement
Requester Address Hold—enable or disable
Target Device Type—I/O or Memory
Requester Device Type—I/O or Memory
Transfer Cycles—Two-Cycle or Fly-By

Mode Register II functions are defined as follows after a hardware reset: Disable Target Address Hold, Increment Requester Address, Target (and Requester) in memory, Fly-By Transfer Cycles. Note: Requester Device Type ignored in Fly-By Transfers.

Software Request Register

The DMA Controller can respond to service requests which are initiated by software. Each channel has an internal request status bit associated with it. The host processor can write to this register to set or reset the request bit of a selected channel.

The status of a group's software DMA service requests can be read from this register as well. Each status bit is cleared upon Terminal Count or external EOP#.

The software DMA requests are non-maskable and subject to priority arbitration with all other software and hardware requests. The entire register is cleared by a hardware reset.

Mask Registers

Each channel has associated with it a mask bit which can be set/reset to disable/enable that channel. Two methods are available for setting and clear-

ing the mask bits. The Mask Set/Reset Register is a write-only register which allows the host to select an individual channel and either set or reset the mask bit for that channel only. The Mask Read/Write Register is available for reading the mask bit status and for writing mask bits in groups of four.

The mask bits of a group may be cleared in one step by executing the Clear Mask Command. See the DMA Programming section for details. A hardware reset sets all of the channel mask bits, disabling all channels.

Status Register

The Status register is a read-only register which contains the Terminal Count (TC) and Service Request status for a group. Four bits indicate the TC status and four bits indicate the hardware request status for the four channels in the group. The TC bits are set when the Byte Count expires, or when and external EOP# is asserted. These bits are cleared by reading from the Status Register. The Service Request bit for a channel indicates when there is a hardware DMA request (DREQn) asserted for that channel. When the request has been removed, the bit is cleared.

Bus Size Register

This write-only register is used to define the bus size of the Target and Requester of a selected channel. The bus sizes programmed will be used to dictate the sizes of the data paths accessed when the DMA channel is active. The values programmed into this register affect the operation of the Temporary Register. When 32-bit bus width is programmed, the 82370 DMA Controller will access the device twice through its 16-bit external Data Bus to perform a 32-bit data transfer. Any byte-assembly required to make the transfers using the specified data path widths will be done in the Temporary Register. The Bus Size register of the Target is used as an increment/decrement value for the Byte Counter and Target Address when in the Fly-By Mode. Upon reset, all channels default to 8-bit Targets and 8-bit Requesters.

Chaining Register

As a command or write register, the Chaining register is used to enable or disable the Chaining Mode for a selected channel. Chaining can either be disabled or enabled for an individual channel, independently of the Chaining Mode status of other channels. After a hardware reset, all channels default to Chaining disabled.

When read by the host, the Chaining Register provides the status of the Chaining Interrupt of each of the channels. These interrupt status bits are cleared when the new buffer information has been loaded.

3.5.2 CHANNEL REGISTERS

Each channel has three individually programmable registers necessary for the DMA process; they are the Base Byte Count, Base Target Address, and Base Requester Address registers. The 24-bit Base Byte Count register contains the number of bytes to be transferred by the channel. The 24-bit Base Target Address Register contains the beginning address (memory or I/O) of the Target device. The 24-bit Base Requester Address register contains the base address (memory or I/O) of the device which is to request DMA service.

Three more registers for each DMA channel exist within the DMA Controller which are directly related to the registers mentioned above. These registers contain the current status of the DMA process. They are the Current Byte Count register, the Current Target Address, and the Current Requester Address. It is these registers which are manipulated (incremented, decremented, or held constant) by the 82370 DMA Controller during the DMA process. The Current registers are loaded from the Base registers at the beginning of a DMA process.

The Base registers are loaded when the host processor writes to the respective channel register addresses. Depending on the mode in which the channel is operating, the Current registers are typically loaded in the same operation. Reading from the channel register addresses yields the contents of the corresponding Current register.

To maintain compatibility with software which accesses an 8237A, a Byte Pointer Flip-Flop is used to control access to the upper and lower bytes of some words of the Channel Registers. These words are accessed as byte pairs at single port addresses. The Byte Pointer Flip-Flop acts as a one-bit pointer which is toggled each time a qualifying Channel Register byte is accessed.

It always points to the next logical byte to be accessed of a pair of bytes.

The Channel registers are arranged as pairs of words, each pair with its own port address. Addressing the port with the Byte Pointer Flip-Flop reset accesses the least significant byte of the pair. The most significant byte is accessed when the Byte Pointer is set.

For compatibility with existing 8237A designs, there is one exception to the above statements about the Byte Pointer Flip-Flop. The third byte (bits 16-23) of

the Target Address is accessed through its own port address. The Byte Pointer Flip-Flop is not affected by any accesses to this byte.

The upper eight bits of the Byte Count Register are cleared when the least significant byte of the register is loaded. This provides compatibility with software which accesses an 8237A. The 8237A has 16-bit Byte Count Registers.

3.5.3 TEMPORARY REGISTERS

Each channel has a 32-bit Temporary Register used for temporary data storage during two-cycle DMA transfers. It is this register in which any necessary byte assembly and disassembly of non-aligned data is performed. Figure 3-22 shows how a block of data will be moved between memory locations with different boundaries. Note that the order of the data does not change.

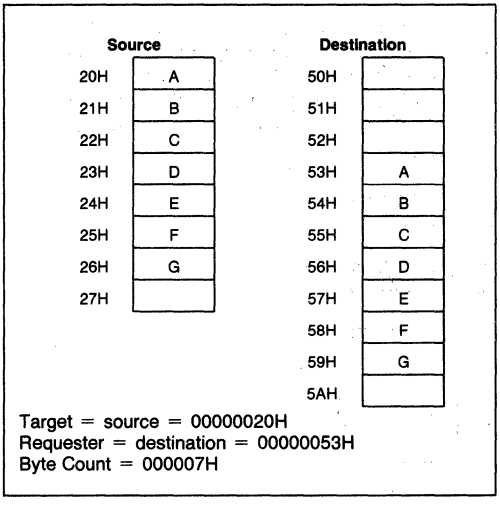


Figure 3-22. Transfer of data between memory locations with different boundaries. This will be the result, independent of data path width.

If the destination is the Requester and an early process termination has been indicated by the EOP# signal or DREQn inactive in the Demand Mode, the Temporary Register is not affected. If data remains in the Temporary Register due to differences in data path widths of the Target and Requester, it will not be transferred or otherwise lost, but will be stored for later transfer.

If the destination is the Target and the EOP# signal is sensed active during the Requester access of a transfer, the DMA Controller will complete the transfer by sending to the Target whatever information is in the Temporary Register at the time of process termination. This implies that the Target could be accessed with partial data in two accesses. For this reason it is advisable to have an I/O device designated as a Requester, unless it is capable of handling partial data transfers.

3.6 DMA Controller Programming

Programming a DMA Channel to perform a needed DMA function is in general a four step process. First the global attributes of the DMA Controller are programmed via the two Command Registers. These global attributes include: priority levels, channel group enables, priority mode, and DREQn/EOP# input sampling.

The second step involves setting the operating modes of the particular channel. The Mode Registers are used to define the type of transfer and the handshaking modes. The Bus Size Register and Chaining Register may also need to be programmed in this step.

The third step in setting up the channel is to load the Base Registers in accordance with the needs of the operating modes chosen in step two. The Current Registers are automatically loaded from the Base Registers, if required by the Buffer Transfer Mode in effect. The information loaded and the order in which it is loaded depends on the operating mode. A channel used for cascading, for example, needs no buffer information and this step can be skipped entirely.

The last step is to enable the newly programmed channel using one of the Mask Registers. The channel is then available to perform the desired data transfer. The status of the channel can be observed at any time through the Status Register, Mask Register, Chaining Register, and Software Request register.

Once the channel is programmed and enabled, the DMA process may be initiated in one of two ways, either by a hardware DMA request (DREQn) or a software request (Software Request Register).

Once programmed to a particular Process/Mode configuration, the channel will operate in that configuration until programmed otherwise. For this reason, restarting a channel after the current buffer expires does not require complete reprogramming of the channel. Only those parameters which have changed need to be reprogrammed. The Byte Count Register is always changed and must be reprogrammed. A Target or Requester Address Register which is incremented or decremented should be reprogrammed also.

3.6.1 BUFFER PROCESSES

The Buffer Process is determined by the Auto-Initialize bit of Mode Register I and the Chaining Register. If Auto-Initialize is enabled, Chaining should not be used.

3.6.1.1 Single Buffer Process

The Single Buffer Process is programmed by disabling Chaining via the Chaining Register and programming Mode Register I for non-Auto-Initialize.

3.6.1.2 Buffer Auto-Initialize Process

Setting the Auto-Initialize bit in Mode Register I is all that is necessary to place the channel in this mode. Buffer Auto-Initialize must not be enabled simultaneous to enabling the Buffer Chaining Mode as this will have unpredictable results.

Once the Base Registers are loaded, the channel is ready to be enabled. The channel will reload its Current Registers from the Base Registers each time the Current Buffer expires, either by an expired Byte Count or an external EOP#.

3.6.1.3 Buffer Chaining

The Buffer Chaining Process is entered into from the Single Buffer Process. The Mode Registers should be programmed first, with all of the Transfer Modes defined as if the channel were to operate in the Single Buffer Process. The channel's Base Registers are then loaded. When the channel has been set up in this way, and the chaining interrupt service routine is in place, the Chaining Process can be entered by programming the Chaining Register. Figure 3-23 illustrates the Buffer Chaining Process.

An interrupt (IRQ1) will be generated immediately after the Chaining Process is entered, as the channel then perceives the Base Registers as empty and in need of reloading. It is important to have the interrupt service routine in place at the time the Chaining Process is entered into. The interrupt request is removed when the most significant byte of the Base Target Address is loaded.

The interrupt will occur again when the first buffer expires and the Current Registers are loaded from the Base Registers. The cycle continues until the Chaining Process is disabled, or the host fails to respond to IRQ1 before the Current Buffer expires.

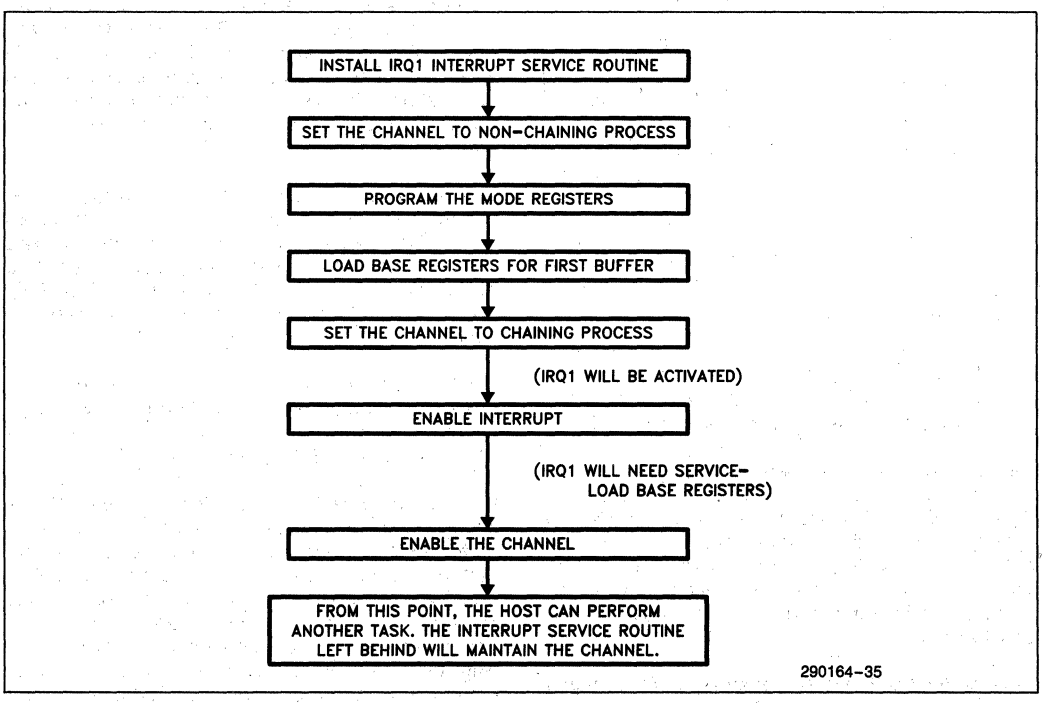


Figure 3-23. Flow of Events in the Buffer Chaining Process

Exiting the Chaining Process can be done by resetting the Chaining Mode Register. If an interrupt is pending for the channel when the Chaining Register is reset, the interrupt request will be removed. The Chaining Process can be temporarily disabled by setting the channel's Mask bit in the Mask Register.

The interrupt service routine for IRQ1 has the responsibility of reloading the Base Registers as necessary. It should check the status of the channel to determine the cause of channel expiration, etc. It should also have access to operating system information regarding the channel, if any exists. The IRQ1 service routine should be capable of determining whether the chain should be continued or terminated and act on that information.

3.6.2 DATA TRANSFER MODES

The Data Transfer Modes are selected via Mode Register I. The Demand, Single, and Block Modes are selected by bits D6 and D7. The individual transfer type (Fly-By vs Two-Cycle, Read-Write-Verify, and I/O vs Memory) is programmed through both of the Mode registers.

3.6.3 CASCADED BUS MASTERS

The Cascade Mode is set by writing ones to D7 and D6 of Mode Register I. When a channel is programmed to operate in the Cascade Mode, all of the other modes associated with Mode Registers I and II are ignored. The priority and DREQn/EOP# definitions of the Command Registers will have the same effect on the channel's operation as any other mode.

3.6.4 SOFTWARE COMMANDS

There are five port addresses which, when written to, command certain operations to be performed by the 82370 DMA Controller. The data written to these locations is not of consequence, writing to the location is all that is necessary to command the 82370 to perform the indicated function. Following are descriptions of the command functions.

Clear Byte Pointer Flip-Flop—Location 000CH

Resets the Byte Pointer Flip-Flop. This command should be performed at the beginning of any access to the channel registers in order to be assured of beginning at a predictable place in the register programming sequence.

Master Clear—Location 000DH

All DMA functions are set to their default states. This command is the equivalent of a hardware reset to the DMA Controller. Functions other than those in the DMA Controller section of the 82370 are not affected by this command.

Clear Mask Register—Channels 0-3
— Location 000EH
Channels 4-7
— Location 00CEH

This command simultaneously clears the Mask Bits of all channels in the addressed group, enabling all of the channels in the group.

Clear TC Interrupt Request—Location 001EH

This command resets the Terminal Count Interrupt Request Flip-Flop. It is provided to allow the program which made a software DMA request to acknowledge that it has responded to the expiration of the requested channel(s).

3.7 Register Definitions

The following diagrams outline the bit definitions and functions of the 82370 DMA Controller's Status and Control Registers. The function and programming of the registers is covered in the previous section on DMA Controller Programming. An entry of "X" as a bit value indicates "don't care."

Channel Registers (read Current, write Base)

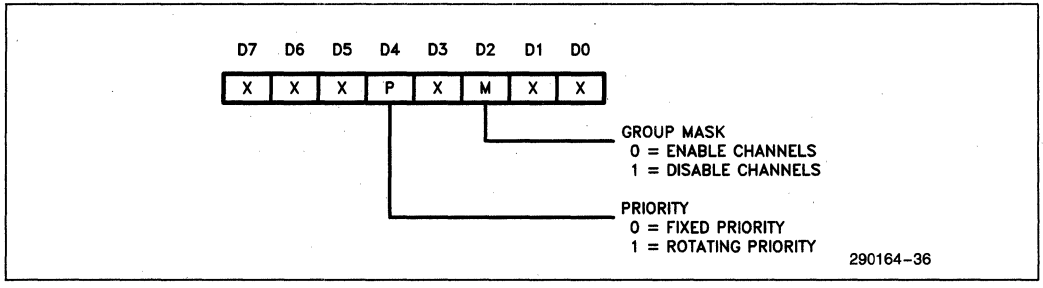
Channel	Register Name	Address (hex)	Byte Pointer	Bits Accessed
Channel 0	Target Address	00	0	0–7
			1	8–15
		87	x	16–23
	Byte Count	01	0	0–7
			1	8–15
		11	0	16–23
	Requester Address	90	0	0–7
			1	8–15
		91	0	16–23
Channel 1	Target Address	02	0	0–7
			1	8–15
		83	x	16–23
	Byte Count	03	0	0–7
			1	8–15
		13	0	16–23
	Requester Address	92	0	0–7
			1	8–15
		93	0	16–23
Channel 2	Target Address	04	0	0–7
			1	8–15
		81	x	16–23
	Byte Count	05	0	0–7
			1	8–15
		15	0	16–23
	Requester Address	94	0	0–7
			1	8–15
		95	0	16–23
Channel 3	Target Address	06	0	0–7
			1	8–15
		82	x	16–23
	Byte Count	07	0	0–7
			1	8–15
		17	0	16–23
	Requester Address	96	0	0–7
			1	8–15
		97	0	16–23
Channel 4	Target Address	C0	0	0–7
			1	8–15
		8F	x	16–23
	Byte Count	C1	0	0–7
			1	8–15
		D1	0	16–23
	Requester Address	98	0	0–7
			1	8–15
		99	0	16–23

Channel Registers (read Current, write Base) (Continued)

Channel	Register Name	Address (hex)	Byte Pointer	Bits Accessed
Channel 5	Target Address	C2	0	0–7
			1	8–15
		8B	x	16–23
	Byte Count	C3	0	0–7
			1	8–15
		D3	0	16–23
	Requester Address	9A	0	0–7
			1	8–15
		9B	0	16–23
Channel 6	Target Address	C4	0	0–7
			1	8–15
		89	x	16–23
	Byte Count	C5	0	0–7
			1	8–15
		D5	0	16–23
	Requester Address	9C	0	0–7
			1	8–15
		9D	0	16–23
Channel 7	Target Address	C6	0	0–7
			1	8–15
		8A	x	16–23
	Byte Count	C7	0	0–7
			1	8–15
		D7	0	16–23
	Requester Address	9E	0	0–7
			1	8–15
		9F	0	16–23

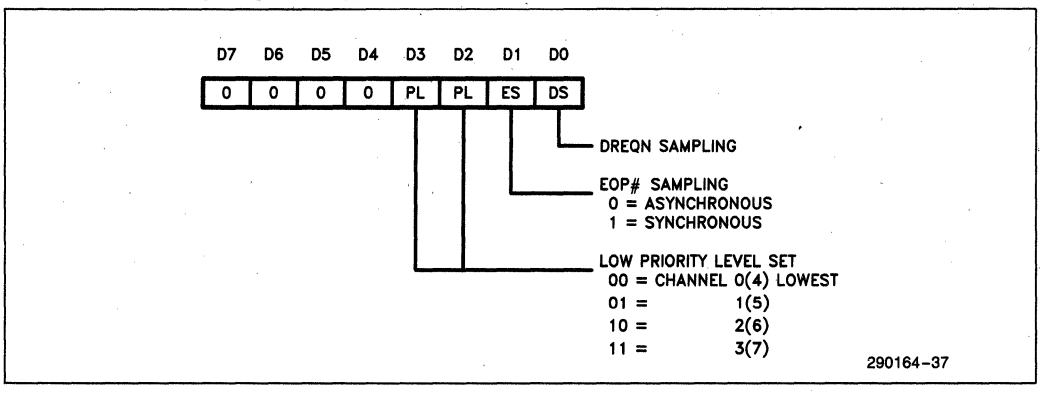
Command Register I (write only)

Port Addresses— Channels 0–3—0008H
Channels 4–7—00C8H



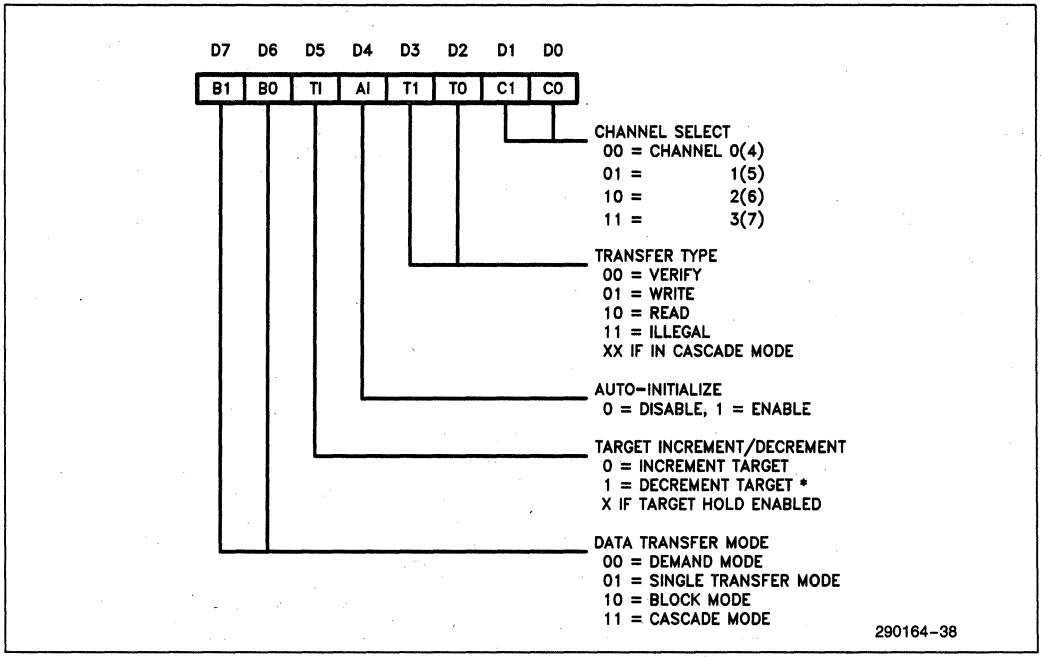
Command Register II (write only)

Port Addresses— Channels 0-3—001AH
Channels 4-7—00DAH



Mode Register I (write only)

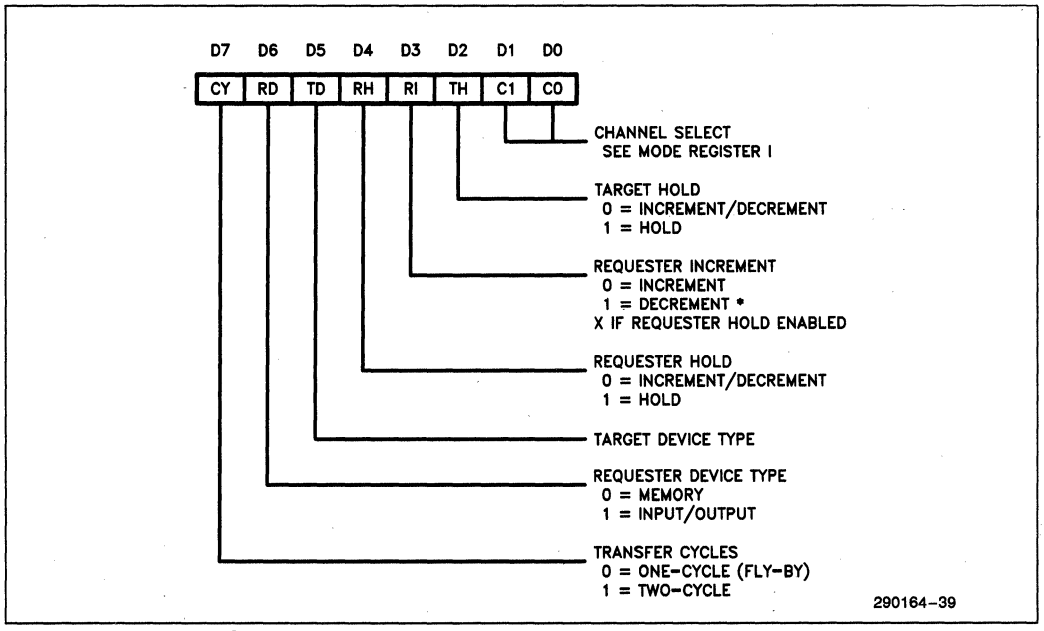
Port Addresses— Channels 0-3—000BH
Channels 4-7—00CBH



*Target and Requester DECREMENT is allowed only for byte transfers.

Mode Register II (write only)

Port Addresses— Channels 0-3—001BH
Channels 4-7—00DBH

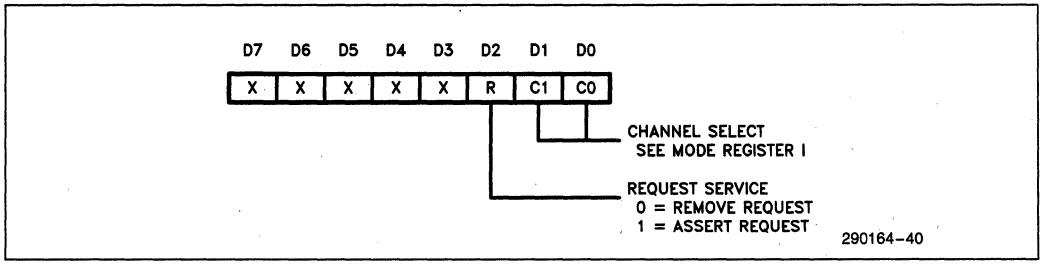


*Target and Requester DECREMENT is allowed only for byte transfers.

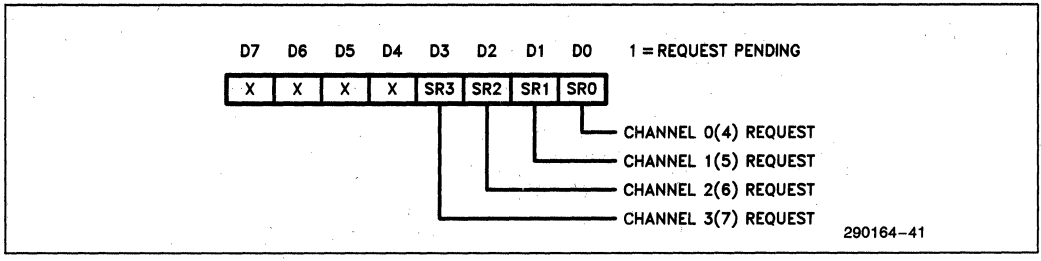
Software Request Register (read/write)

Port Addresses— Channels 0-3—0009H
Channels 4-7—00C9H

Write Format: Software DMA Service Request

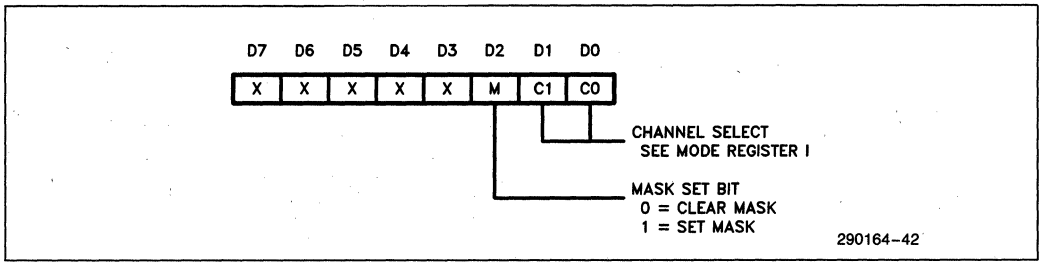


Read Format: Software Requests Pending



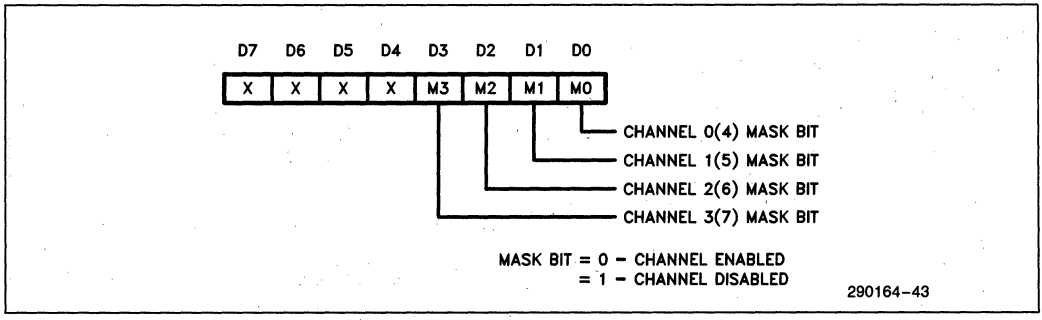
Mask Set/Reset Register Individual Channel Mask (write only)

Port Addresses— Channels 0–3—000AH
Channels 4–7—00CAH



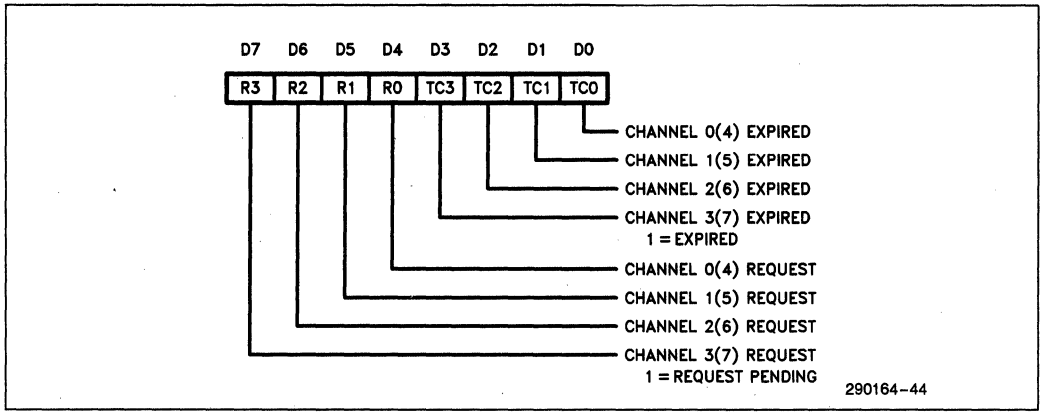
Mask Read/Write Register Group Channel Mask (read/write)

Port Addresses— Channels 0–3—000FH
Channels 4–7—00CFH



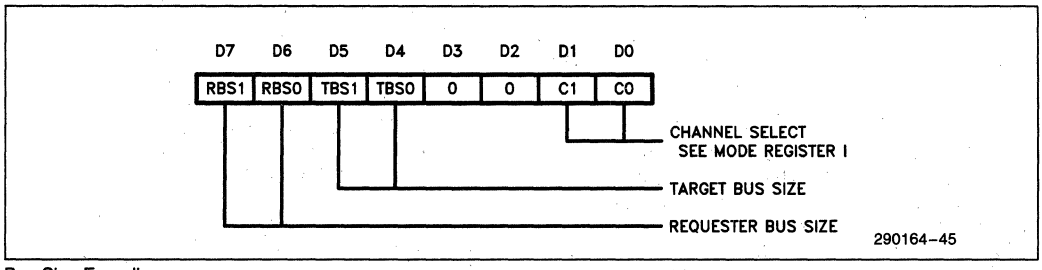
Status Register Channel Process Status (read only)

Port Addresses— Channels 0–3—0008H
Channels 4–7—00C8H



Bus Size Register Set Data Path Width (write only)

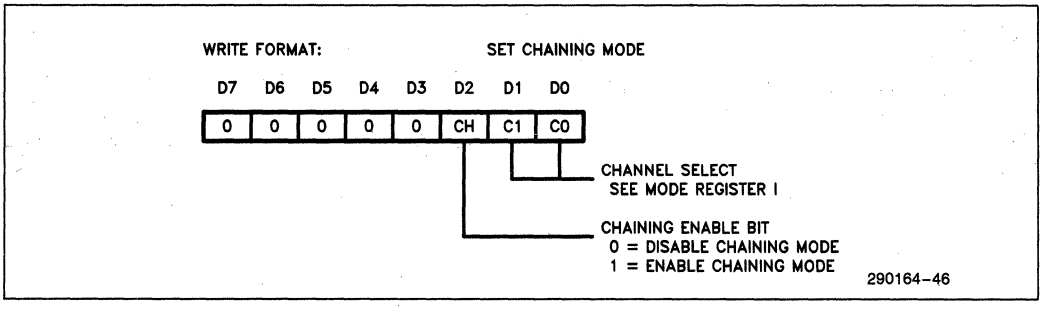
Port Addresses— Channels 0–3—0018H
Channels 4–7—00D8H

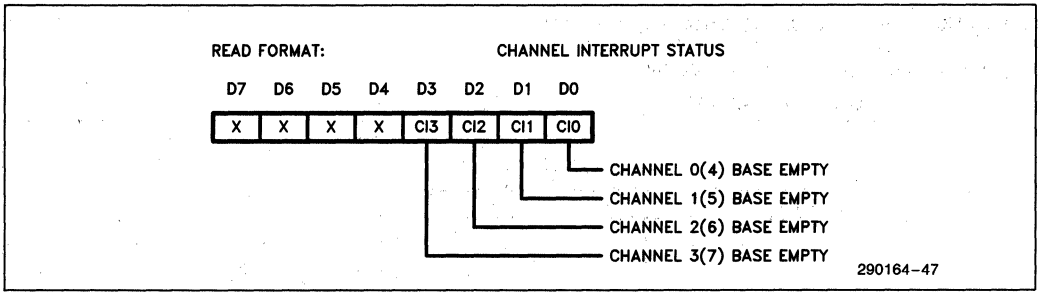


Bus Size Encoding:
00 = Reserved by Intel 10 = 16-bit Bus
01 = 32-bit Bus* 11 = 8-bit Bus
*If programmed as 32-bit bus width, the corresponding device will be accessed in two 16-bit cycles provided that the data is aligned within word boundary.

Chaining Register (read/write)

Port Addresses— Channels 0–3—0019H
Channels 4–7—00D9H





3.8 8237A Compatibility

The register arrangement of the 82370 DMA Controller is a superset of the 8237A DMA Controller. Functionally the 82370 DMA Controller is very different from the 8237A. Most of the functions of the 8237A are performed also by the 82370. The following discussion points out the differences between the 8237A and the 82370.

The 8237A is limited to transfers between I/O and memory only (except in one special case, where two channels can be used to perform memory-to-memory transfers). The 82370 DMA Controller can transfer between any combination of memory and I/O. Several other features of the 8237A are enhanced or expanded in the 82370 and other features are added.

The 8237A is an 8-bit only DMA device. For programming compatibility, all of the 8-bit registers are preserved in the 82370. The 82370 is programmed via 8-bit registers. The address registers in the 82370 are 24-bit registers in order to support the 80376's 24-bit bus. The Byte Count Registers are 24-bit registers, allowing support of larger data blocks than possible with the 8237A.

All of the 8237A's operating modes are supported by the 82370 (except the cumbersome two-channel memory-to-memory transfer). The 82370 performs memory-to-memory transfers using only one channel. The 82370 has the added features of buffer pipelining (Buffer Chaining Process) and programmable priority levels.

The 82370 also adds the feature of address registers for both destination and source. These addresses may be incremented, decremented, or held constant, as required by the application of the individual channel. This allows any combination of destination and source device.

Each DMA channel has associated with it a Target and a Requester. In the 8237A, the Target is the device which can be accessed by the address register, the Requester is the device which is accessed by the DMA Acknowledge signals and must be an I/O device.

4.0 PROGRAMMABLE INTERRUPT CONTROLLER (PIC)

4.1 Functional Description

The 82370 Programmable Interrupt Controller (PIC) consists of three enhanced 82C59A Interrupt Controllers. These three controllers together provide 15 external and 5 internal interrupt request inputs. Each external request input can be cascaded with an additional 82C59A slave controller. This scheme allows the 82370 to support a maximum of 120 (15 x 8) external interrupt request inputs.

Following one or more interrupt requests, the 82370 PIC issues an interrupt signal to the 80376. When the 80376 host processor responds with an interrupt acknowledge signal, the PIC will arbitrate between the pending interrupt requests and place the interrupt vector associated with the highest priority pending request on the data bus.

The major enhancement in the 82370 PIC over the 82C59A is that each of the interrupt request inputs can be individually programmed with its own interrupt vector, allowing more flexibility in interrupt vector mapping.

4.1.1 INTERNAL BLOCK DIAGRAM

The block diagram of the 82370 Programmable Interrupt Controller is shown in Figure 4-1. Internally,

the PIC consists of three 82C59A banks: A, B and C. The three banks are cascaded to one another: C is cascaded to B, B is cascaded to A. The INT output of Bank A is used externally to interrupt the 80376.

Bank A has nine interrupt request inputs (two are unused), and Banks B and C have eight interrupt request inputs. Of the fifteen external interrupt request inputs, two are shared by other functions. Specifically, the Interrupt Request 3 input (IRQ3#) can be used as the Timer 2 output (TOUT2#). This pin can be used in three different ways: IRQ3# input only, TOUT2# output only, or using TOUT2# to generate an IRQ3# interrupt request. Also, the Interrupt Request 9 input (IRQ9#) can be used as DMA Request 4 input (DREQ 4). Typically, only IRQ9# or DREQ4 can be used at a time.

4.1.2 INTERRUPT CONTROLLER BANKS

All three banks are identical, with the exception of the IRQ1.5 on Bank A. Therefore, only one bank will be discussed. In the 82370 PIC, all external requests can be cascaded into and each interrupt controller bank behaves like a master. As compared to the 82C59A, the enhancements in the banks are:

— All interrupt vectors are individually programmable. (In the 82C59A, the vectors must be programmed in eight consecutive interrupt vector locations.)

— The cascade address is provided on the Data Bus (D0-D7). (In the 82C59A, three dedicated control signals (CAS0, CAS1, CAS2) are used for master/slave cascading.)

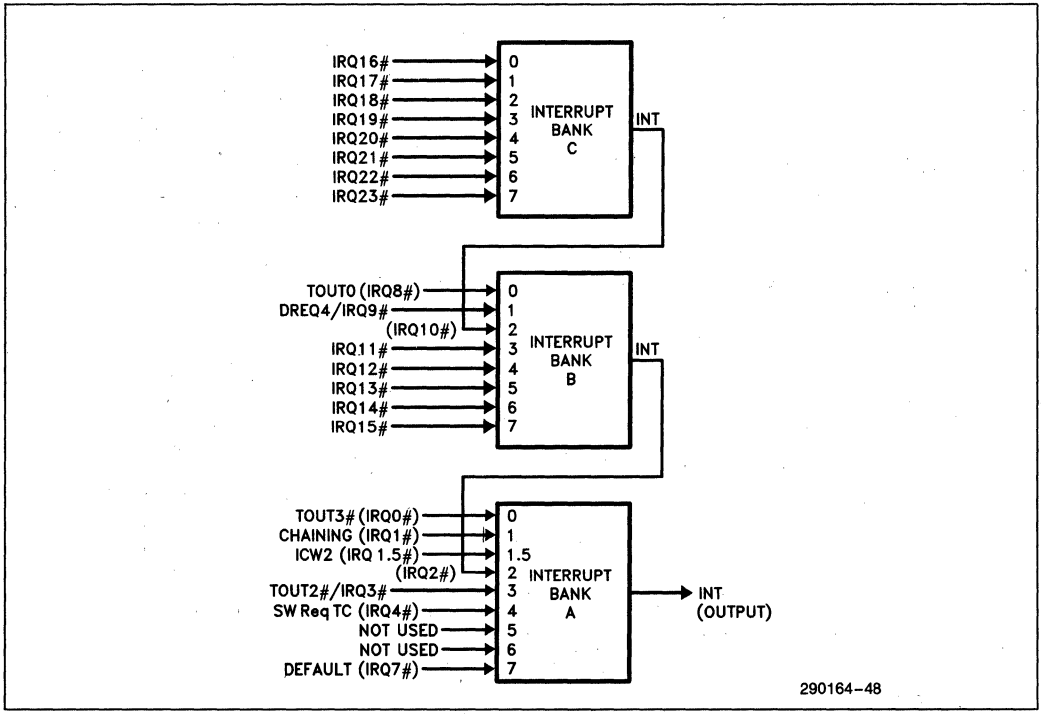


Figure 4-1. Interrupt Controller Block Diagram

The block diagram of a bank is shown in Figure 4-2. As can be seen from this figure, the bank consists of six major blocks: the Interrupt Request Register (IRR), the In-Service Register (ISR), the Interrupt Mask Register (IMR), the Priority Resolver (PR), the Vector Registers (VR), and the Control Logic. The functional description of each block is included below.

INTERRUPT REQUEST (IRR) AND IN-SERVICE REGISTER (ISR)

The interrupts at the Interrupt Request (IRQ) input lines are handled by two registers in cascade, the Interrupt Request Register (IRR) and the In-Service Register (ISR). The IRR is used to store all interrupt levels which are requesting service; and the ISR is used to store all interrupt levels which are being serviced.

PRIORITY RESOLVER (PR)

This logic block determines the priorities of the bits set in the IRR. The highest priority is selected and strobed into the corresponding bit of the ISR during an Interrupt Acknowledge cycle.

INTERRUPT MASK REGISTER (IMR)

The IMR stores the bits which mask the interrupt lines to be masked (disabled). The IMR operates on the IRR. Masking of a higher priority input will not affect the interrupt request lines of lower priority.

VECTOR REGISTERS (VR)

This block contains a set of Vector Registers, one for each interrupt request line, to store the pre-programmed interrupt vector number. The corresponding vector number will be driven onto the Data Bus of the 82370 during the Interrupt Acknowledge cycle.

CONTROL LOGIC

The Control Logic coordinates the overall operations of the other internal blocks within the same bank. This logic will drive the Interrupt Output signal (INT) HIGH when one or more unmasked interrupt inputs are active (LOW). The INT output signal goes directly to the 80376 (in bank A) or to another bank to which this bank is cascaded (see Figure 4-1). Also,

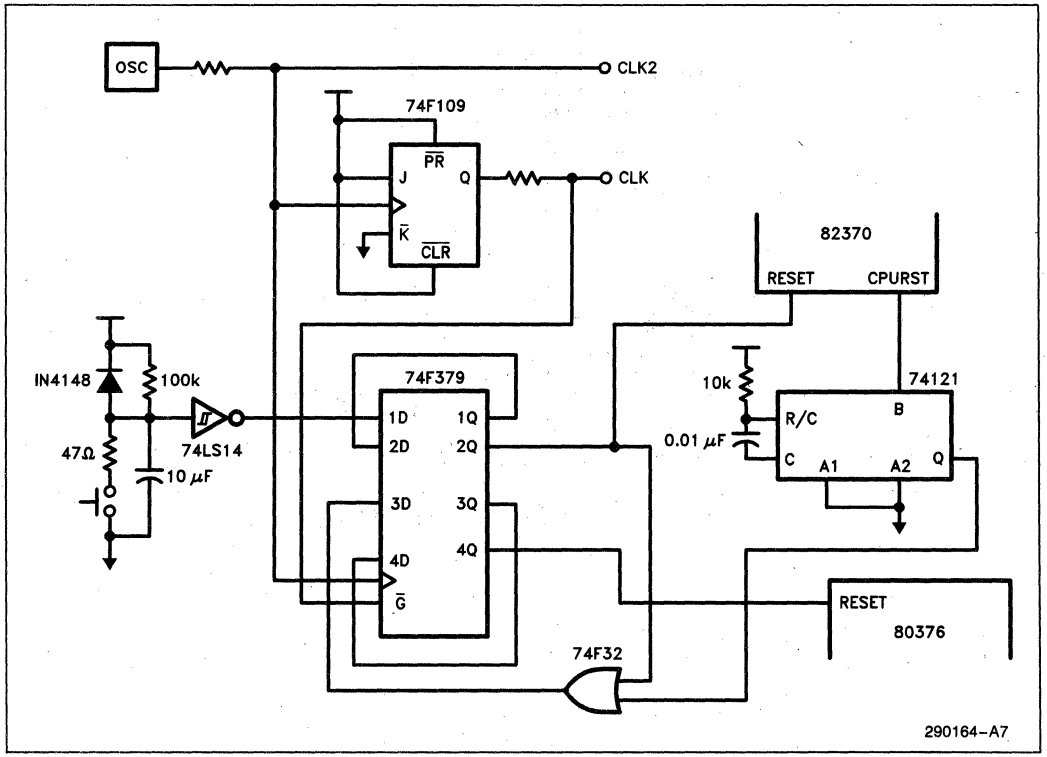


Figure 4-2. Interrupt Bank Block Diagram

this logic will recognize an Interrupt Acknowledge cycle (via M/IO#, D/C# and W/R# signals). During this bus cycle, the Control Logic will enable the corresponding Vector Register to drive the interrupt vector onto the Data Bus.

In bank A, the Control Logic is also responsible for handling the special ICW2 interrupt request input (IRQ1.5).

4.2 Interface Signals

4.2.1 INTERRUPT INPUTS

There are 15 external Interrupt Request inputs and 5 internal Interrupt Requests. The external request inputs are: IRQ3#, IRQ9#, IRQ11# to IRQ23#. They are shown in bold arrows in Figure 4-1. All IRQ inputs are active LOW and they can be programmed (via a control bit in the Initialization Command Word 1 (ICW1)) to be either edge-triggered or level-triggered. In order to be recognized as a valid interrupt request, the interrupt input must be active (LOW) until the first INTA cycle (see Bus Functional Description). Note that all 15 external Interrupt Request inputs have weak internal pull-up resistors.

As mentioned earlier, an 82C59A can be cascaded to each external interrupt input to expand the interrupt capacity to a maximum of 120 levels. Also, two of the interrupt inputs are dual functions: IRQ3# can be used as Timer 2 output (TOUT2#) and IRQ9# can be used as DREQ4 input. IRQ3# is a bidirectional dual function pin. This interrupt request input is wired-OR with the output of Timer 2 (TOUT2#). If only IRQ3# function is to be used, Timer 2 should be programmed so that OUT2 is LOW. Note that TOUT2# can also be used to generate an interrupt request to IRQ3# input.

The five internal interrupt requests serve special system functions. They are shown in Table 4-1. The following paragraphs describe these interrupts.

Table 4-1. 82370 Internal Interrupt Requests

Interrupt Request	Interrupt Source
IRQ0#	Timer 3 Output (TOUT3)
IRQ8#	Timer 0 Output (TOUT0)
IRQ1#	DMA Chaining Request
IRQ4#	DMA Terminal Count
IRQ1.5#	ICW2 Written

TIMER 0 AND TIMER 3 INTERRUPT REQUESTS

IRQ8# and IRQ0# interrupt requests are initiated by the output of Timers 0 and 3, respectively. Each of these requests is generated by an edge-detector flip-flop.

The flip-flops are activated by the following conditions:

Set — Rising edge of timer output (TOUT);

Clear — Interrupt acknowledge for this request; OR Request is masked (disabled); OR Hardware Reset.

CHAINING AND TERMINAL COUNT INTERRUPTS

These interrupt requests are generated by the 82370 DMA Controller. The chaining request (IRQ1#) indicates that the DMA Base Register is not loaded. The Terminal Count request (IRQ4#) indicates that a software DMA request was cleared.

ICW2 INTERRUPT REQUEST

Whenever an Initialization Control Word 2 (ICW2) is written to a Bank, a special ICW2 interrupt request is generated. The interrupt will be cleared when the newly programmed ICW2 Register is read. This interrupt request is in Bank A at level 1.5. This interrupt request is internally ORed with the Cascaded Request from Bank B and is always assigned a higher priority than the Cascaded Request.

This special interrupt is provided to support compatibility with the original 82C59A. A detailed description of this interrupt is discussed in the Programming section.

DEFAULT INTERRUPT (IRQ7#)

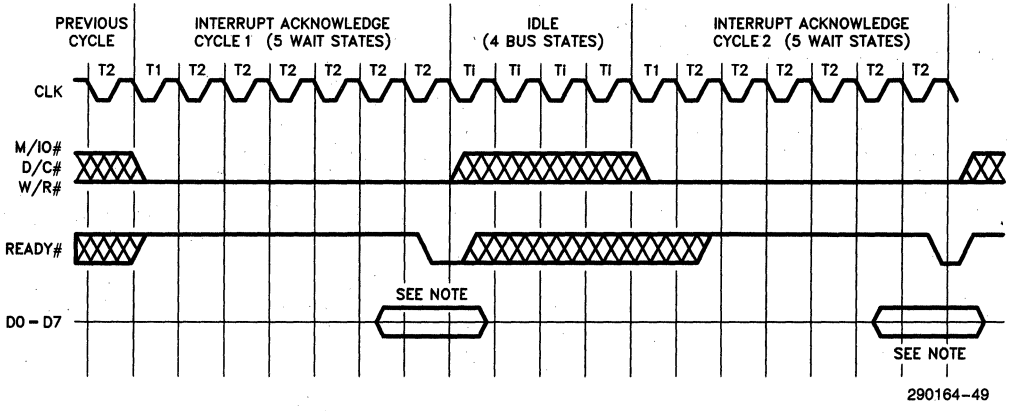
During an Interrupt Acknowledge cycle, if there is no active pending request, the PIC will automatically generate a default vector. This vector corresponds to the IRQ7# vector in bank A.

4.2.2 INTERRUPT OUTPUT (INT)

The INT output pin is taken directly from bank A. This signal should be tied to the Maskable Interrupt Request (INTR) of the 80376. When this signal is active (HIGH), it indicates that one or more internal/external interrupt requests are pending. The 80376 is expected to respond with an interrupt acknowledge cycle.

4.3 Bus Functional Description

The INT output of bank A will be activated as a result of any unmasked interrupt request. This may be a non-cascaded or cascaded request. After the PIC has driven the INT signal HIGH, the 80376 will respond by performing two interrupt acknowledge cycles. The timing diagram in Figure 4-3 shows a typical interrupt acknowledge process between the 82370 and the 80376 CPU.



NOTE:
What is actually driven on the Data Bus depends on if the current interrupt request is a Slave Request.

	INTA Cycle 1	INTA Cycle 2
NON-SLAVE REQUEST	00H	Vector
SLAVE REQUEST	Slave Address	High Impedence*

*Slave will place a vector at this time.

Figure 4-3. Interrupt Acknowledge Cycle

After activating the INT signal, the 82370 monitors the status lines (M/IO#, D/C#, W/R#) and waits for the 80376 to initiate the first interrupt acknowledge cycle. In the 80376 environment, two successive interrupt acknowledge cycles (INTA) marked by M/IO# = LOW, D/C# = LOW, and W/R# = LOW are performed. During the first INTA cycle, the PIC will determine the highest priority request. Assuming this interrupt input has no external Slave Controller cascaded to it, the 82370 will drive the Data Bus with 00H in the first INTA cycle. During the second INTA cycle, the 82370 PIC will drive the Data Bus with the corresponding pre-programmed interrupt vector.

If the PIC determines (from the ICW3) that this interrupt input has an external Slave Controller cascaded to it, it will drive the Data Bus with the specific Slave Cascade Address (instead of 00H) during the first INTA cycle. This Slave Cascade Address is the pre-programmed content in the corresponding Vector Register. This means that no Slave Address should be chosen to be 00H. Note that the Slave Address and Interrupt Vector are different interpretations of the same thing. They are both the contents of the programmable Vector Register. During the second INTA cycle, the Data Bus will be floated so that the external Slave Controller can drive its interrupt vector on the bus. Since the Slave Interrupt Controller resides on the system bus, bus transceiver enable and direction control logic must take this into consideration.

In order to have a successful interrupt service, the interrupt request input must be held valid (LOW) until the beginning of the first interrupt acknowledge cycle. If there is no pending interrupt request when the first INTA cycle is generated, the PIC will generate a default vector, which is the IRQ7 vector (Bank A, level 7).

According to the Bus Cycle definition of the 80376, there will be four Bus Idle States between the two interrupt acknowledge cycles. These idle bus cycles will be initiated by the 80376. Also, during each interrupt acknowledge cycle, the internal Wait State Generator of the 82370 will automatically generate the required number of wait states for internal delays.

4.4 Modes of Operation

A variety of modes and commands are available for controlling the 82370 PIC. All of them are programmable; that is, they may be changed dynamically under software control. In fact, each bank can be programmed individually to operate in different modes. With these modes and commands, many possible configurations are conceivable, giving the user enough versatility for almost any interrupt controlled application.

This section is not intended to show how the 82370 PIC can be programmed. Rather, it describes the operation in different modes.

4.4.1 END-OF-INTERRUPT

Upon completion of an interrupt service routine, the interrupted bank needs to be notified so its ISR can be updated. This allows the PIC to keep track of which interrupt levels are in the process of being serviced and their relative priorities. Three different End-Of-Interrupt (EOI) formats are available. They are: Non-Specific EOI Command, Specific EOI Command, and Automatic EOI Mode. Selection of which EOI to use is dependent upon the interrupt operations the user wishes to perform.

If the 82370 is NOT programmed in the Automatic EOI Mode, an EOI command must be issued by the 80376 to the specific 82370 PIC Controller Bank. Also, if this controller bank is cascaded to another internal bank, an EOI command must also be sent to the bank to which this bank is cascaded. For example, if an interrupt request of Bank C in the 82370 PIC is serviced, an EOI should be written into Bank C, Bank B and Bank A. If the request comes from an external interrupt controller cascaded to Bank C, then an EOI should be written into the external controller as well.

NON-SPECIFIC EOI COMMAND

A Non-Specific EOI command sent from the 80376 lets the 82370 PIC bank know when a service routine has been completed, without specification of its exact interrupt level. The respective interrupt bank automatically determines the interrupt level and resets the correct bit in the ISR.

To take advantage of the Non-Specific EOI, the interrupt bank must be in a mode of operation in which it can predetermine its in-service routine levels. For this reason, the Non-Specific EOI command should only be used when the most recent level acknowledged and serviced is always the highest priority level (i.e. in the Fully Nested Mode structure to be described below). When the interrupt bank receives a Non-Specific EOI command, it simply resets the highest priority ISR bit to indicate that the highest priority routine in service is finished.

Special consideration should be taken when deciding to use the Non-Specific EOI command. Here are two operating conditions in which it is best NOT used since the Fully Nested Mode structure will be destroyed:

— Using the Set Priority command within an interrupt service routine.

— Using a Special Mask Mode.

These conditions are covered in more detail in their own sections, but are listed here for reference.

SPECIFIC EOI COMMAND

Unlike a Non-Specific EOI command which automatically resets the highest priority ISR bit, a Specific EOI command specifies an exact ISR bit to be reset. Any one of the IRQ levels of an interrupt bank can be specified in the command.

The Specific EOI command is needed to reset the ISR bit of a completed service routine whenever the interrupt bank is not able to automatically determine it. The Specific EOI command can be used in all conditions of operation, including those that prohibit Non-Specific EOI command usage mentioned above.

AUTOMATIC EOI MODE

When programmed in the Automatic EOI Mode, the 80376 no longer needs to issue a command to notify the interrupt bank it has completed an interrupt routine. The interrupt bank accomplishes this by performing a Non-Specific EOI automatically at the end of the second INTA cycle.

Special consideration should be taken when deciding to use the Automatic EOI Mode because it may disturb the Fully Nested Mode structure. In the Automatic EOI Mode, the ISR bit of a routine in service is reset right after it is acknowledged, thus leaving no designation in the ISR that a service routine is being executed. If any interrupt request within the same bank occurs during this time and interrupts are enabled, it will get serviced regardless of its priority. Therefore, when using this mode, the 80376 should keep its interrupt request input disabled during execution of a service routine. By doing this, higher priority interrupt levels will be serviced only after the completion of a routine in service. This guideline restores the Fully Nested Mode structure. However, in this scheme, a routine in service cannot be interrupted since the host's interrupt request input is disabled.

4.4.2 INTERRUPT PRIORITIES

The 82370 PIC provides various methods for arranging the interrupt priorities of the interrupt request inputs to suit different applications. The following subsections explain these methods in detail.

4.4.2.1 Fully Nested Mode

The Fully Nested Mode of operation is a general purpose priority mode. This mode supports a multi-level interrupt structure in which all of the Interrupt Request (IRQ) inputs within one bank are arranged from highest to lowest.

Unless otherwise programmed, the Fully Nested Mode is entered by default upon initialization. At this time, IRQ0# is assigned the highest priority (priority=0) and IRQ7# the lowest (priority=7). This default priority can be changed, as will be explained later in the Rotating Priority Mode.

When an interrupt is acknowledged, the highest priority request is determined from the Interrupt Request Register (IRR) and its vector is placed on the bus. In addition, the corresponding bit in the In-Service Register (ISR) is set to designate the routine in service. This ISR bit will remain set until the 80376 issues an End Of Interrupt (EOI) command immediately before returning from the service routine; or alternately, if the Automatic End Of Interrupt (AEOI) bit is set, the ISR bit will be reset at the end of the second INTA cycle.

While the ISR bit is set, all further interrupts of the same or lower priority are inhibited. Higher level interrupts can still generate an interrupt, which will be acknowledged only if the 80376 internal interrupt enable flip-flop has been reenabled (through software inside the current service routine).

4.4.2.2 Automatic Rotation–Equal Priority Devices

Automatic rotation of priorities serves in applications where the interrupting devices are of equal priority within an interrupt bank. In this kind of environment, once a device is serviced, all other equal priority peripherals should be given a chance to be serviced before the original device is serviced again. This is accomplished by automatically assigning a device the lowest priority after being serviced. Thus, in the worst case, the device would have to wait until all other peripherals connected to the same bank are serviced before it is serviced again.

There are two methods of accomplishing automatic rotation. One is used in conjunction with the Non-Specific EOI command and the other is used with the Automatic EOI mode. These two methods are discussed below.

ROTATE ON NON-SPECIFIC EOI COMMAND

When the Rotate On Non-Specific EOI command is issued, the highest ISR bit is reset as in a normal Non-Specific EOI command. However, after it is reset, the corresponding Interrupt Request (IRQ) level is assigned the lowest priority. Other IRQ priorities rotate to conform to the Fully Nested Mode based on the newly assigned low priority.

Figure 4-4 shows how the Rotate On Non-Specific EOI command affects the interrupt priorities. Assume the IRQ priorities were assigned with IRQ0 the highest and IRQ7 the lowest. IRQ6 and IRQ4 are

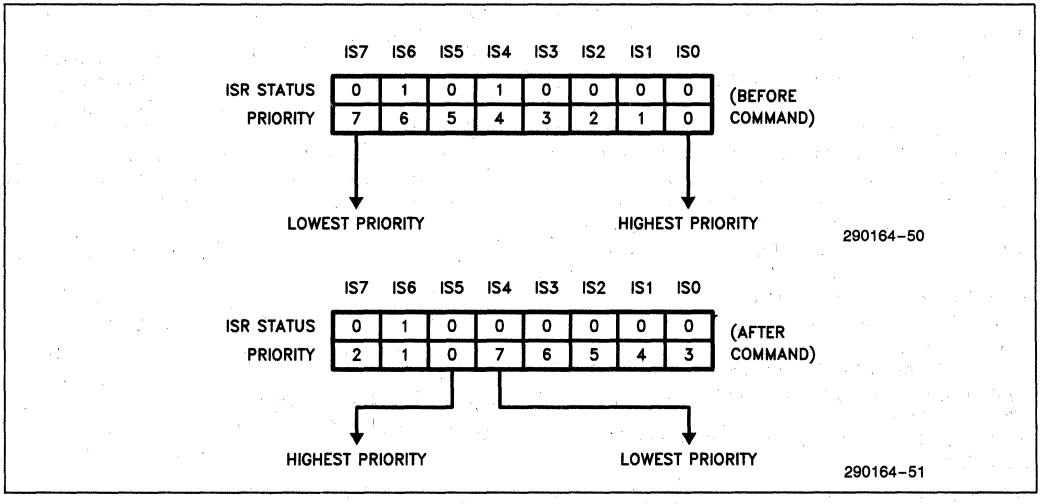


Figure 4-4. Rotate On Non-Specific EOI Command

already in service but neither is completed. Being the higher priority routine, IRQ4 is necessarily the routine being executed. During the IRQ4 routine, a rotate on Non-Specific EOI command is executed. When this happens, Bit 4 in the ISR is reset. IRQ4 then becomes the lowest priority and IRQ5 becomes the highest.

ROTATE ON AUTOMATIC EOI MODE

The Rotate On Automatic EOI Mode works much like the Rotate On Non-Specific EOI Command. The main difference is that priority rotation is done automatically after the second INTA cycle of an interrupt request. To enter or exit this mode, a Rotate-On-Automatic-EOI Set Command and Rotate-On-Automatic-EOI Clear Command is provided. After this mode is entered, no other commands are needed as in the normal Automatic EOI Mode. However, it must be noted again that when using any form of the Automatic EOI Mode, special consideration should be taken. The guideline presented in the Automatic EOI Mode also applies here.

4.4.2.3 Specific Rotation–Specific Priority

Specific rotation gives the user versatile capabilities in interrupt controlled operations. It serves in those applications in which a specific device's interrupt priority must be altered. As opposed to Automatic Rotation which will automatically set priorities after each interrupt request is serviced, specific rotation is completely user controlled. That is, the user selects which interrupt level is to receive the lowest or the highest priority. This can be done during the main program or within interrupt routines. Two specific rotation commands are available to the user: Set Priority Command and Rotate On Specific EOI Command.

SET PRIORITY COMMAND

The Set Priority Command allows the programmer to assign an IRQ level the lowest priority. All other interrupt levels will conform to the Fully Nested Mode based on the newly assigned low priority.

ROTATE ON SPECIFIC EOI COMMAND

The Rotate On Specific EOI Command is literally a combination of the Set Priority Command and the Specific EOI Command. Like the Set Priority Command, a specified IRQ level is assigned lowest priority. Like the Specific EOI Command, a specified level will be reset in the ISR. Thus, this command accomplishes both tasks in one single command.

4.4.2.4 Interrupt Priority Mode Summary

In order to simplify understanding the many modes of interrupt priority, Table 4-2 is provided to bring out their summary of operations.

4.4.3 INTERRUPT MASKING

VIA INTERRUPT MASK REGISTER

Each bank in the 82370 PIC has an Interrupt Mask Register (IMR) which enhances interrupt control ca-

Table 4-2. Interrupt Priority Mode Summary

Interrupt Priority Mode	Operation Summary	Effect On Priority After EOI	
		Non-Specific/Automatic	Specific
Fully-Nested Mode	IRQ0# - Highest Priority IRQ7# - Lowest Priority	No change in priority. Highest ISR bit is reset.	Not Applicable.
Automatic Rotation (Equal Priority Devices)	Interrupt level just serviced is the lowest priority. Other priorities rotate to conform to Fully-Nested Mode.	Highest ISR bit is reset and the corresponding level becomes the lowest priority.	Not Applicable.
Specific Rotation (Specific Priority Devices)	User specifies the lowest priority level. Other priorities rotate to conform to Fully-Nested Mode.	Not Applicable.	As described under "Operation Summary".

pabilities. This IMR allows individual IRQ masking. When an IRQ is masked, its interrupt request is disabled until it is unmasked. Each bit in the 8-bit IMR disables one interrupt channel if it is set (HIGH). Bit 0 masks IRQ0, Bit 1 masks IRQ1 and so forth. Masking an IRQ channel will only disable the corresponding channel and does not affect the others' operations.

The IMR acts only on the output of the IRR. That is, if an interrupt occurs while its IMR bit is set, this request is not "forgotten". Even with an IRQ input masked, it is still possible to set the IRR. Therefore, when the IMR bit is reset, an interrupt request to the 80376 will then be generated, providing that the IRQ request remains active. If the IRQ request is removed before the IMR is reset, the Default Interrupt Vector (Bank A, level 7) will be generated during the interrupt acknowledge cycle.

SPECIAL MASK MODE

In the Fully Nested Mode, all IRQ levels of lower priority than the routine in service are inhibited. However, in some applications, it may be desirable to let a lower priority interrupt request to interrupt the routine in service. One method to achieve this is by using the Special Mask Mode. Working in conjunction with the IMR, the Special Mask Mode enables interrupts from all levels except the level in service. This is usually done inside an interrupt service routine by masking the level that is in service and then issuing the Special Mask Mode Command. Once the Special Mask Mode is enabled, it remains in effect until it is disabled.

4.4.4 EDGE OR LEVEL INTERRUPT TRIGGERING

Each bank in the 82370 PIC can be programmed independently for either edge or level sensing for the interrupt request signals. Recall that all IRQ inputs are active LOW. Therefore, in the edge triggered mode, an active edge is defined as an input transition from an inactive (HIGH) to active (LOW) state. The interrupt input may remain active without generating another interrupt. During level triggered mode, an interrupt request will be recognized by an active (LOW) input, and there is no need for edge detection. However, the interrupt request must be removed before the EOI Command is issued, or the 80376 must be disabled to prevent a second false interrupt from occurring.

In either modes, the interrupt request input must be active (LOW) during the first INTA cycle in order to be recognized. Otherwise, the Default Interrupt Vector will be generated at level 7 of Bank A.

4.4.5 INTERRUPT CASCADING

As mentioned previously, the 82370 allows for external Slave interrupt controllers to be cascaded to any of its external interrupt request pins. The 82370 PIC indicates that an external Slave Controller is to be serviced by putting the contents of the Vector Register associated with the particular request on the 80376 Data Bus during the first INTA cycle (instead of 00H during a non-slave service). The external logic should latch the vector on the Data Bus using the INTA status signals and use it to select the external Slave Controller to be serviced (see Figure 4-5). The selected Slave will then respond to the second INTA cycle and place its vector on the Data Bus. This method requires that if external Slave Controllers are used in the system, no vector should be programmed to 00H.

Since the external Slave Cascade Address is provided on the Data Bus during INTA cycle 1, an external latch is required to capture this address for the Slave Controller. A simple scheme is depicted in Figure 4-5 below.

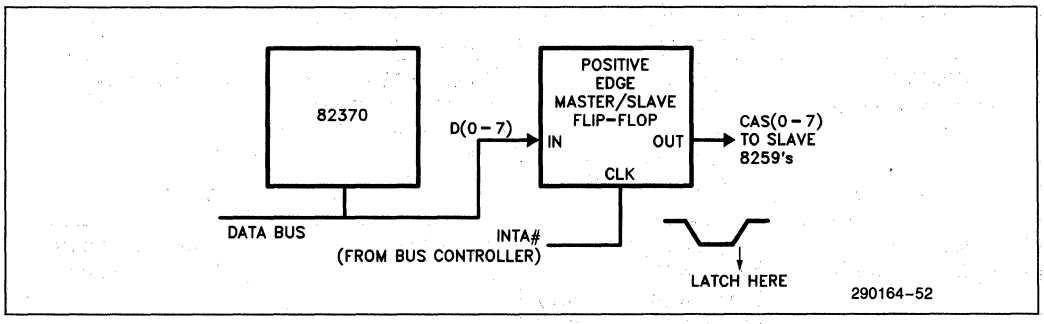


Figure 4-5. Slave Cascade Address Capturing

4.4.5.1 Special Fully Nested Mode

This mode will be used where cascading is employed and the priority is to be conserved within each Slave Controller. The Special Fully Nested Mode is similar to the "regular" Fully Nested Mode with the following exceptions:

— When an interrupt request from a Slave Controller is in service, this Slave Controller is not locked out from the Master's priority logic. Further interrupt requests from the higher priority logic within the Slave Controller will be recognized by the 82370 PIC and will initiate interrupts to the 80376. In comparing to the "regular" Fully Nested Mode, the Slave Controller is masked out when its request is in service and no higher requests from the same Slave Controller can be serviced.

— Before exiting the interrupt service routine, the software has to check whether the interrupt serviced was the only request from the Slave Controller. This is done by sending a Non-Specific EOI Command to the Slave Controller and then reading its In Service Register. If there are no requests in the Slave Controller, a Non-Specific EOI can be sent to the corresponding 82370 PIC bank also. Otherwise, no EOI should be sent.

4.4.6 READING INTERRUPT STATUS

The 82370 PIC provides several ways to read different status of each interrupt bank for more flexible interrupt control operations. These include polling the highest priority pending interrupt request and reading the contents of different interrupt status registers.

4.4.6.1 Poll Command

The 82370 PIC supports status polling operations with the Poll Command. In a Poll Command, the pending interrupt request with the highest priority can be determined. To use this command, the INT output is not used, or the 80376 interrupt is disabled. Service to devices is achieved by software using the Poll Command.

This mode is useful if there is a routine command common to several levels so that the INTA sequence is not needed. Another application is to use the Poll Command to expand the number of priority levels.

Notice that the ICW2 mechanism is not supported for the Poll Command. However, if the Poll Command is used, the programmable Vector Registers are of no concern since no INTA cycle will be generated.

4.4.6.2 Reading Interrupt Registers

The contents of each interrupt register (IRR, ISR, and IMR) can be read to update the user's program on the present status of the 82370 PIC. This can be a versatile tool in the decision making process of a service routine, giving the user more control over interrupt operations.

The reading of the IRR and ISR contents can be performed via the Operation Control Word 3 by using a Read Status Register Command and the content of IMR can be read via a simple read operation of the register itself.

4.5 Register Set Overview

Each bank of the 82370 PIC consists of a set of 8-bit registers to control its operations. The address map of all the registers is shown in Table 4-3 below. Since all three register sets are identical in functions, only one set will be described.

Functionally, each register set can be divided into five groups. They are: the four Initialization Command Words (ICW's), the three Operation Control Words (OCW's), the Poll/Interrupt Request/In-Service Register, the Interrupt Mask Register, and the Vector Registers. A description of each group follows.

Table 4-3. Interrupt Controller Register Address Map

Port Address	Access	Register Description
20H	Write	Bank B ICW1, OCW2, or OCW3
	Read	Bank B Poll, Request or In-Service Status Register
21H	Write	Bank B ICW2, ICW3, ICW4, OCW1
	Read	Bank B Mask Register
22H	Read	Bank B ICW2
28H	Read/Write	IRQ8 Vector Register
29H	Read/Write	IRQ9 Vector Register
2AH	Read/Write	Reserved
2BH	Read/Write	IRQ11 Vector Register
2CH	Read/Write	IRQ12 Vector Register
2DH	Read/Write	IRQ13 Vector Register
2EH	Read/Write	IRQ14 Vector Register
2FH	Read/Write	IRQ15 Vector Register
A0H	Write	Bank C ICW1, OCW2, or OCW3
	Read	Bank C Poll, Request or In-Service Status Register
A1H	Write	Bank C ICW2, ICW3, ICW4, OCW1
	Read	Bank C Mask Register
A2H	Read	Bank C ICW2
A8H	Read/Write	IRQ16 Vector Register
A9H	Read/Write	IRQ17 Vector Register
AAH	Read/Write	IRQ18 Vector Register
ABH	Read/Write	IRQ19 Vector Register
ACH	Read/Write	IRQ20 Vector Register
ADH	Read/Write	IRQ21 Vector Register
AEH	Read/Write	IRQ22 Vector Register
AFH	Read/Write	IRQ23 Vector Register
30H	Write	Bank A ICW1, OCW2, or OCW3
	Read	Bank A Poll, Request or In-Service Status Register
31H	Write	Bank A ICW2, ICW3, ICW4, OCW1
	Read	Bank A Mask Register
32H	Read	Bank ICW2
38H	Read/Write	IRQ0 Vector Register
39H	Read/Write	IRQ1 Vector Register
3AH	Read/Write	IRQ1.5 Vector Register
3BH	Read/Write	IRQ3 Vector Register
3CH	Read/Write	IRQ4 Vector Register
3DH	Read/Write	Reserved
3EH	Read/Write	Reserved
3FH	Read/Write	IRQ7 Vector Register

4.5.1 INITIALIZATION COMMAND WORDS (ICW)

Before normal operation can begin, the 82370 PIC must be brought to a known state. There are four 8-bit Initialization Command Words in each interrupt bank to setup the necessary conditions and modes for proper operation. Except for the second command word (ICW2) which is a read/write register, the other three are write-only registers. Without going into detail of the bit definitions of the command words, the following subsections give a brief description of what functions each command word controls.

ICW1

The ICW1 has three major functions. They are:

— To select between the two IRQ input triggering modes (edge- or level-triggered);

— To designate whether or not the interrupt bank is to be used alone or in the cascade mode. If the cascade mode is desired, the interrupt bank will accept ICW3 for further cascade mode programming. Otherwise, no ICW3 will be accepted;

— To determine whether or not ICW4 will be issued; that is, if any of the ICW4 operations are to be used.

ICW2

ICW2 is provided for compatibility with the 82C59A only. Its contents do not affect the operation of the interrupt bank in any way. Whenever the ICW2 of any of the three banks is written into, an interrupt is generated from bank A at level 1.5. The interrupt request will be cleared after the ICW2 register has been read by the 80376. The user is expected to program the corresponding vector register or to use it as an indicator that an attempt was made to alter the contents. Note that each ICW2 register has different addresses for read and write operations.

ICW3

The interrupt bank will only accept an ICW3 if programmed in the external cascade mode (as indicated in ICW1). ICW3 is used for specific programming within the cascade mode. The bits in ICW3 indicate which interrupt request inputs have a Slave cascaded to them. This will subsequently affect the interrupt vector generation during the interrupt acknowledge cycles as described previously.

ICW4

The ICW4 is accepted only if it was selected in ICW1. This command word register serves two functions:

— To select either the Automatic EOI mode or software EOI mode;

— To select if the Special Nested mode is to be used in conjunction with the cascade mode.

4.5.2 OPERATION CONTROL WORDS (OCW)

Once initialized by the ICW's, the interrupt banks will be operating in the Fully Nested Mode by default and they are ready to accept interrupt requests. However, the operations of each interrupt bank can be further controlled or modified by the use of OCW's. Three OCW's are available for programming various modes and commands. Note that all OCW's are 8-bit write-only registers.

The modes and operations controlled by the OCW's are:

— Fully Nested Mode;

— Rotating Priority Mode;

— Special Mask Mode;

— Poll Mode;

— EOI Commands;

— Read Status Commands.

OCW1

OCW1 is used solely for masking operations. It provides a direct link to the Internal Mask Register (IMR). The 80376 can write to this OCW register to enable or disable the interrupt inputs. Reading the pre-programmed mask can be done via the Interrupt Mask Register which will be discussed shortly.

OCW2

OCW2 is used to select End-Of-Interrupt, Automatic Priority Rotation, and Specific Priority Rotation operations. Associated commands and modes of these operations are selected using the different combinations of bits in OCW2.

Specifically, the OCW2 is used to:

— Designate an interrupt level (0–7) to be used to reset a specific ISR bit or to set a specific priority. This function can be enabled or disabled;

— Select which software EOI command (if any) is to be executed (i.e. Non-Specific or Specific EOI);

— Enable one of the priority rotation operations (i.e. Rotate On Non-Specific EOI, Rotate On Automatic EOI, or Rotate On Specific EOI).

OCW3

There are three main categories of operation that OCW3 controls. They are summarized as follows:

— To select and execute the Read Status Register Commands, either reading the Interrupt Request Register (IRR) or the In-Service Register (ISR);

— To issue the Poll Command. The Poll Command will override a Read Register Command if both functions are enabled simultaneously;

— To set or reset the Special Mask Mode.

4.5.3 POLL/INTERRUPT REQUEST/IN-SERVICE STATUS REGISTER

As the name implies, this 8-bit read-only register has multiple functions. Depending on the command issued in the OCW3, the content of this register reflects the result of the command executed. For a Poll Command, the register read contains the binary code of the highest priority level requesting service (if any). For a Read IRR Command, the register content will show the current pending interrupt request(s). Finally, for a Read ISR Command, this register will specify all interrupt levels which are being serviced.

4.5.4 INTERRUPT MASK REGISTER (IMR)

This is a read-only 8-bit register which, when read, will specify all interrupt levels within the same bank that are masked.

4.5.5 VECTOR REGISTERS (VR)

Each interrupt request input has an 8-bit read/write programmable vector register associated with it. The registers should be programmed to contain the interrupt vector for the corresponding request. The contents of the Vector Register will be placed on the Data Bus during the INTA cycles as described previously.

4.6 Programming

Programming the 82370 PIC is accomplished by using two types of command words: ICW's and OCW's. All modes and commands explained in the previous sections are programmable using the ICW's and OCW's. The ICW's are issued from the 80376 in a sequential format and are used to setup the banks in the 82370 PIC in an initial state of operation. The OCW's are issued as needed to vary and control the 82370 PIC's operations.

Both ICW's and OCW's are sent by the 80376 to the interrupt banks via the Data Bus. Each bank distinguishes between the different ICW's and OCW's by the I/O address map, the sequence they are issued (ICW's only), and by some dedicated bits among the ICW's and OCW's.

An example of programming the 82370 interrupt controllers is given in Appendix C (Programming the 82370 Interrupt Controllers).

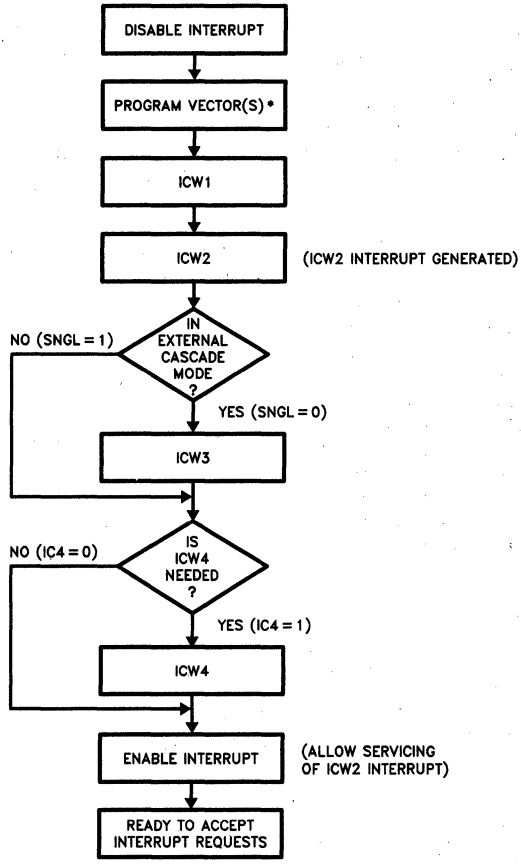
All three interrupt banks are programmed in a similar way. Therefore, only a single bank will be described in the following sections.

4.6.1 INITIALIZATION (ICW)

Before normal operation can begin, each bank must be initialized by programming a sequence of two to four bytes written into the ICW's.

Figure 4-6 shows the initialization flow for an interrupt bank. Both ICW1 and ICW2 must be issued for any form of operation. However, ICW3 and ICW4 are used only if designated in ICW1. Once initialized, if any programming changes within the ICW's are to be made, the entire ICW sequence must be reprogrammed, not just an individual ICW.

Note that although the ICW2's in the 82370 PIC do not effect the Bank's operation, they still must be programmed in order to preserve the compatibility with the 82C59A. The contents programmed are not relevant to the overall operations of the interrupt banks. Also, whenever one of the three ICW2's is programmed, an interrupt level 1.5 in Bank A will be generated. This interrupt request will be cleared upon reading of the ICW2 registers. Since the three ICW2's share the same interrupt level and the system may not know the origin of the interrupt, all three ICW2's must be read.



*ICW2 vector address must be programmed now.
Other vector addresses may be programmed via ICW2 interrupt service routine.

Figure 4-6. Initialization Sequence

Certain internal setup conditions occur automatically within the interrupt bank after the first ICW (ICW1) has been issued. These are:

— The edge sensitive circuit is reset, which means that following initialization, an interrupt request input must make a HIGH-to-LOW transition to generate an interrupt;

— The Interrupt Mask Register (IMR) is cleared; that is, all interrupt inputs are enabled;

— IRQ7 input of each bank is assigned priority 7 (lowest);

— Special Mask Mode is cleared and Status Read is set to IRR;

— If no ICW4 is needed, then no Automatic-EOI is selected.

4.6.2 VECTOR REGISTERS (VR)

Each interrupt request input has a separate Vector Register. These Vector Registers are used to store the pre-programmed vector number corresponding to their interrupt sources. In order to guarantee proper interrupt handling, all Vector Registers must be programmed with the predefined vector numbers. Since an interrupt request will be generated whenever an ICW2 is written during the initialization sequence, it is important that the Vector Register of IRQ1.5 in Bank A should be initialized and the interrupt service routine of this vector is set up before the ICW's are written.

4.6.3 OPERATION CONTROL WORDS (OCW)

After the ICW's are programmed, the operations of each interrupt controller bank can be changed by writing into the OCW's as explained before. There is no special programming sequence required for the OCW's. Any OCW may be written at any time in order to change the mode of or to perform certain operations on the interrupt banks.

4.6.3.1 Read Status and Poll Commands (OCW3)

Since the reading of IRR and ISR status as well as the result of a Poll Command are available on the same read-only Status Register, a special Read Status/Poll Command must be issued before the Poll/Interrupt Request/In-Service Status Register is read. This command can be specified by writing the required control word into OCW3. As mentioned earlier, if both the Poll Command and the Status Read Command are enabled simultaneously, the Poll Command will override the Status Read. That is, after the command execution, the Status Register will contain the result of the Poll Command.

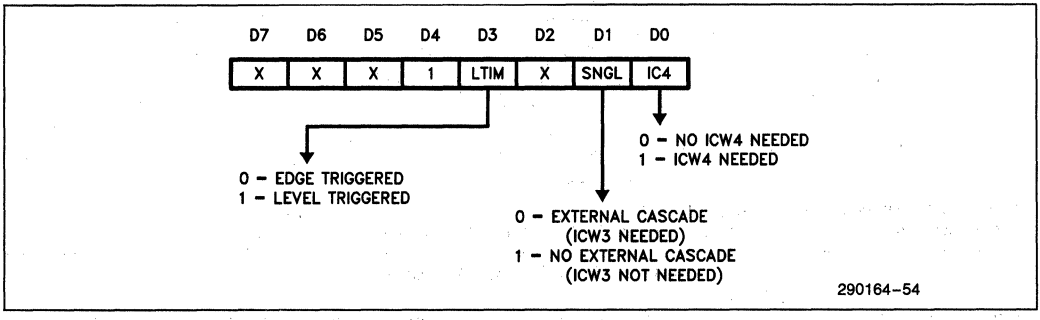
Note that for reading IRR and ISR, there is no need to issue a Read Status Command to the OCW3 every time the IRR or ISR is to be read. Once a Read Status Command is received by the interrupt bank, it "remembers" which register is selected. However, this is not true when the Poll Command is used.

In the Poll Command, after the OCW3 is written, the 82370 PIC treats the next read to the Status Register as an interrupt acknowledge. This will set the appropriate IS bit if there is a request and read the priority level. Interrupt Request input status remains unchanged from the Poll Command to the Status Read.

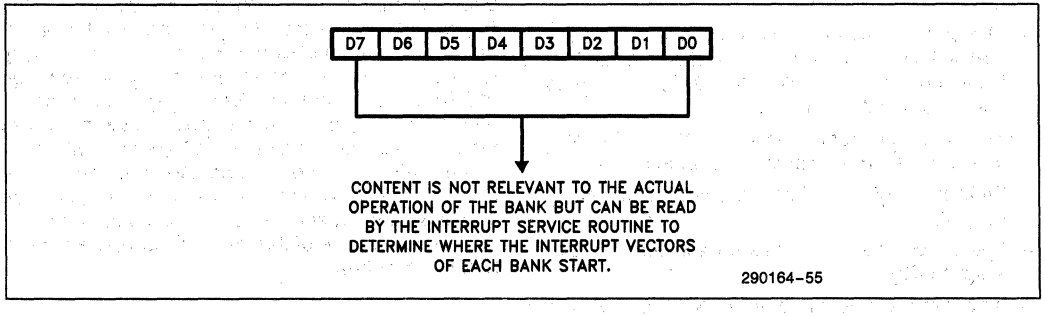
In addition to the above read commands, the Interrupt Mask Register (IMR) can also be read. When read, this register reflects the contents of the preprogrammed OCW1 which contains information on which interrupt request(s) is(are) currently disabled.

4.7 Register Bit Definition

INITIALIZATION COMMAND WORD 1 (ICW1)

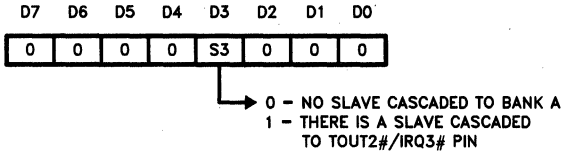


INITIALIZATION COMMAND WORD 2 (ICW2)

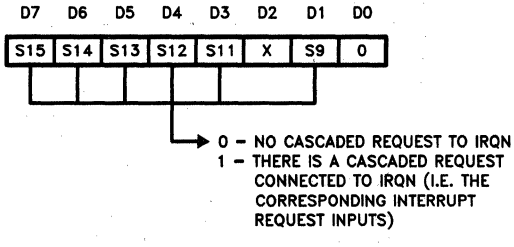


INITIALIZATION COMMAND WORD 3 (ICW3)

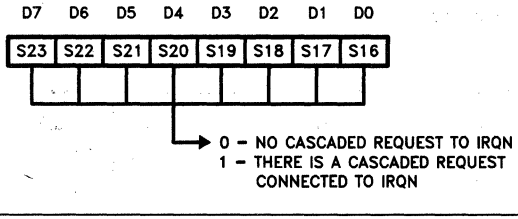
ICW3 for Bank A:



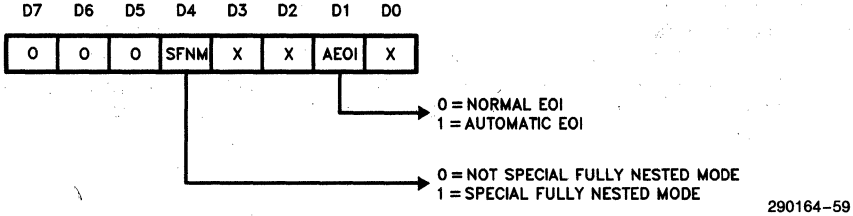
ICW3 for Bank B:



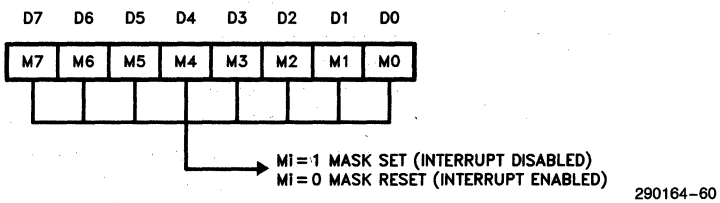
ICW3 for Bank C:



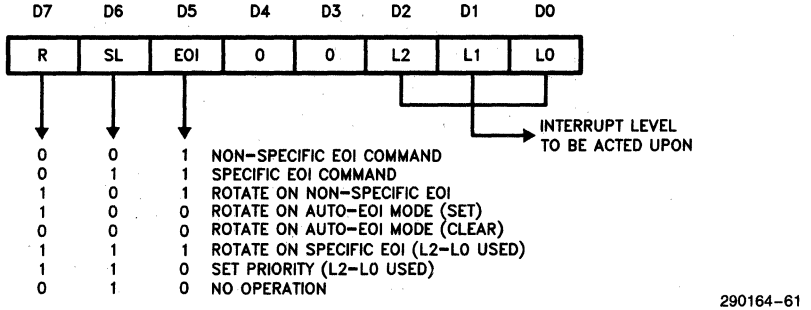
INITIALIZATION COMMAND WORD 4 (ICW4)



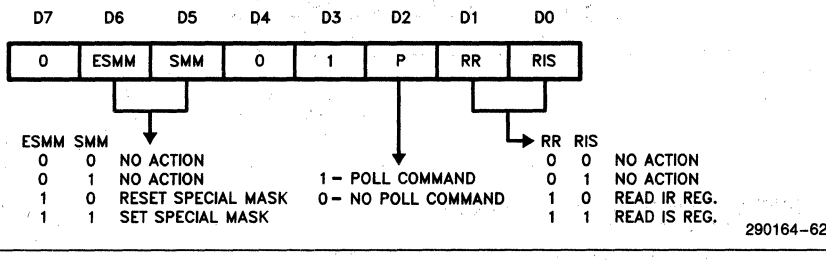
OPERATION CONTROL WORD 1 (OCW1)



OPERATION CONTROL WORD 2 (OCW2)



OPERATION CONTROL WORD 3 (OCW3)

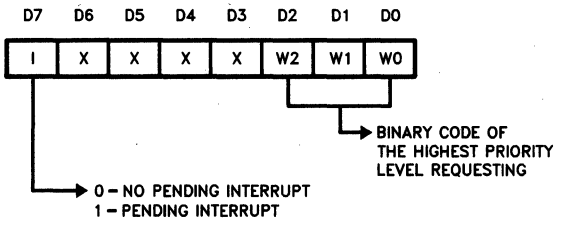


ESMM — Enable Special Mask Mode. When this bit is set to 1, it enables the SMM bit to set or reset the Special Mask Mode. When this bit is set to 0, SMM bit becomes don't care.

SMM — Special Mask Mode. If ESMM = 1 and SMM = 1, the interrupt controller bank will enter Special Mask Mode. If ESMM = 1 and SMM = 0, the bank will revert to normal mask mode. When ESMM = 0, SMM has no effect.

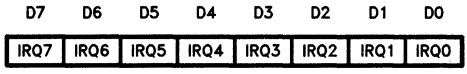
POLL/INTERRUPT REQUEST/IN-SERVICE STATUS REGISTER

Poll Command Status



290164-63

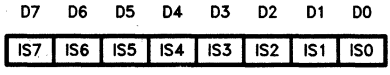
Interrupt Request Status



290164-64

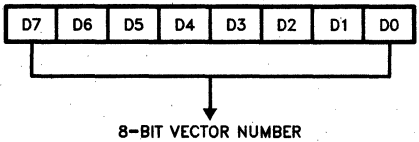
NOTE:
Although all Interrupt Request inputs are active LOW, the internal logical will invert the state of the pins so that when there is a pending interrupt request at the input, the corresponding IRQ bit will be set to HIGH in the Interrupt Request Status register.

In-Service Status



290164-65

VECTOR REGISTER (VR)



290164-66

Table 4-4. Register Operational Summary

Operational Description	Command Words	Bits
Fully Nested Mode	OCW-Default	
Non-specific EOI Command	OCW2	EOI
Specific EOI Command	OCW2	SL, EOI, L0–L2
Automatic EOI Mode	ICW1, ICW4	IC4, AEOI
Rotate On Non-Specific EOI Command	OCW2	EOI
Rotate On Automatic EOI Mode	OCW2	R, SL, EOI
Set Priority Command	OCW2	L0–L2
Rotate On Specific EOI Command	OCW2	R, SL, EOI
Interrupt Mask Register	OCW1	M0–M7
Special Mask Mode	OCW3	ESMM, SMM
Level Triggered Mode	ICW1	LTIM
Edge Triggered Mode	ICW1	LTIM
Read Register Command, IRR	OCW3	RR, RIS
Read Register Command, ISR	OCW3	RR, RIS
Read IMR	IMR	M0–M7
Poll Command	OCW3	P
Special Fully Nested Mode	ICW1, ICW4	IC4, SFNM

4.8 Register Operational Summary

For ease of reference, Table 4-4 gives a summary of the different operating modes and commands with their corresponding registers.

5.0 PROGRAMMABLE INTERVAL TIMER

5.1 Functional Description

The 82370 contains four independently Programmable Interval Timers: Timer 0–3. All four timers are functionally compatible to the Intel 82C54. The first three timers (Timer 0–2) have specific functions. The fourth timer, Timer 3, is a general purpose timer. Table 5-1 depicts the functions of each timer. A brief description of each timer's function follows.

Table 5-1. Programmable Interval Timer Functions

Timer	Output	Function
0	IRQ8	Event Based IRQ8 Generator
1	TOUT1/REF#	Gen. Purpose/DRAM Refresh Req.
2	TOUT2/IRQ3#	Gen. Purpose/Speaker Out/IRQ3#
3	TOUT3#	Gen. Purpose/IRQ0 Generator

TIMER 0—Event Based Interrupt Request 8 Generator

Timer 0 is intended to be used as an Event Counter. The output of this timer will generate an Interrupt Request 8 (IRQ8) upon a rising edge of the timer output (TOUT0). Normally, this timer is used to implement a time-of-day clock or system tick. The Timer 0 output is not available as an external signal.

TIMER 1—General Purpose/DRAM Refresh Request

The output of Timer 1, TOUT1, can be used as a general purpose timer or as a DRAM Refresh Request signal. The rising edge of this output creates a DRAM refresh request to the 82370 DRAM Refresh Controller. Upon reset, the Refresh Request function is disabled, and the output pin is the Timer 1 output.

TIMER 2—General Purpose/Speaker Out/IRQ3#

The Timer 2 output, TOUT2#, could be used to support tone generation to an external speaker. This pin is a bidirectional signal. When used as an input, a logic LOW asserted at this pin will generate an Interrupt Request 3 (IRQ3#) (see Programmable Interrupt Controller).

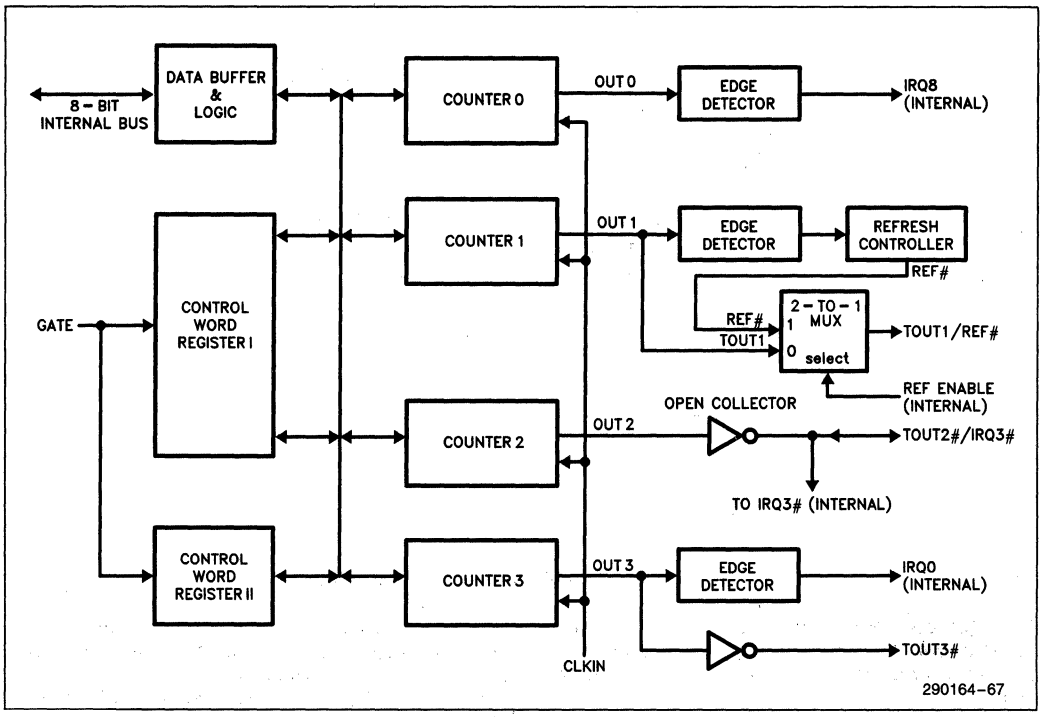


Figure 5-1. Block Diagram of Programmable Interval Timer

TIMER 3—General Purpose/Interrupt Request 0 Generator

The output of Timer 3 is fed to an edge detector and generates an Interrupt Request 0 (IRQ0) in the 82370. The inverted output of this timer (TOUT3#) is also available as an external signal for general purpose use.

5.1.1 INTERNAL ARCHITECTURE

The functional block diagram of the Programmable Interval Timer section is shown in Figure 5-1. Following is a description of each block.

DATA BUFFER & READ/WRITE LOGIC

This part of the Programmable Interval Timer is used to interface the four timers to the 82370 internal bus. The Data Buffer is for transferring commands and data between the 8-bit internal bus and the timers. The Read/Write Logic accepts inputs from the internal bus and generates signals to control other functional blocks within the timer section.

CONTROL WORD REGISTERS I & II

The Control Word Registers are write-only registers. They are used to control the operating modes of the timers. Control Word Register I controls Timers 0, 1 and 2, and Control Word Register II controls Timer 3. Detailed description of the Control Word Registers will be included in the Register Set Overview section.

COUNTER 0, COUNTER 1, COUNTER 2, COUNTER 3

Counters 0, 1, 2, and 3 are the major parts of Timers 0, 1, 2, and 3, respectively. These four functional blocks are identical in operation, so only a single counter will be described. The internal block diagram of one counter is shown in Figure 5-2.

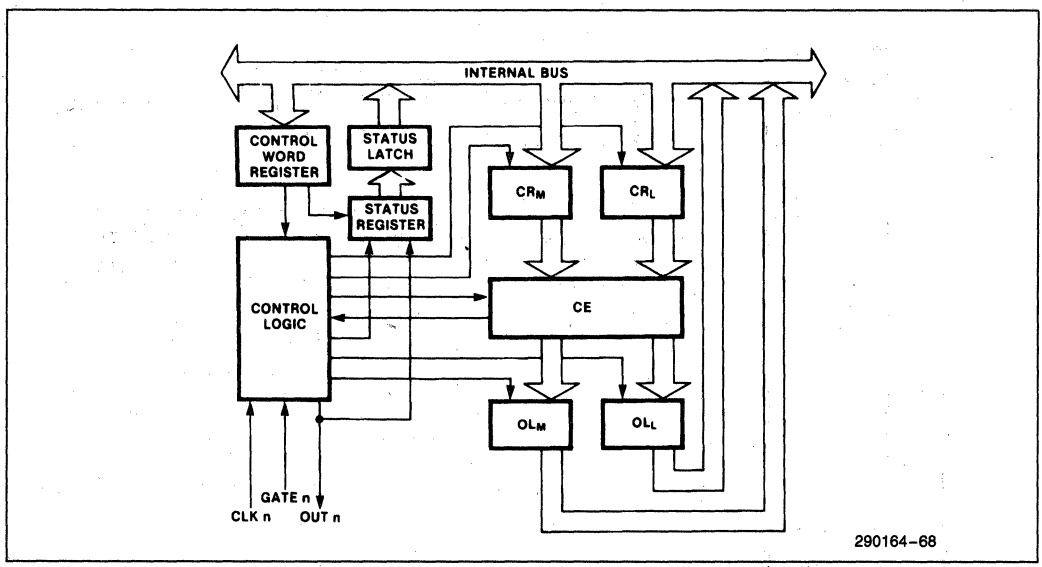


Figure 5-2. Internal Block Diagram of a Counter

The four counters share a common clock input (CLKIN), but otherwise are fully independent. Each counter is programmable to operate in a different mode.

Although the Control Word Register is shown in the figure, it is not part of the counter itself. Its programmed contents are used to control the operations of the counters.

The Status Register, when latched, contains the current contents of the Control Word Register and status of the output and Null Count Flag (see Read Back Command).

The Counting Element (CE) is the actual counter. It is a 16-bit presettable synchronous down counter.

The Output Latches (OL) contain two 8-bit latches (OLM and OLL). Normally, these latches "follow" the content of the CE. OLM contains the most significant byte of the counter and OLL contains the least significant byte. If the Counter Latch Command is sent to the counter, OL will latch the present count until read by the 80376 and then return to follow the CE. One latch at a time is enabled by the timer's Control Logic to drive the internal bus. This is how the 16-bit Counter communicates over the 8-bit internal bus. Note that CE cannot be read. Whenever the count is read, it is one of the OL's that is being read.

When a new count is written into the counter, the value will be stored in the Count Registers (CR), and transferred to CE. The transferring of the contents from CR's to CE is defined as "loading" of the counter. The Count Register contains two 8-bit registers: CRM (which contains the most significant byte) and CRL (which contains the least significant byte). Similar to the OL's, the Control Logic allows one register at a time to be loaded from the 8-bit internal bus. However, both bytes are transferred from the CR's to the CE simultaneously. Both CR's are cleared when the Counter is programmed. This way, if the Counter has been programmed for one byte count (either the most significant or the least significant byte only), the other byte will be zero. Note that CE cannot be written into directly. Whenever a count is written, it is the CR that is being written.

As shown in the diagram, the Control Logic consists of three signals: CLKIN, GATE, and OUT. CLKIN and GATE will be discussed in detail in the section that follows. OUT is the internal output of the counter. The external outputs of some timers (TOUT) are the inverted version of OUT (see TOUT1, TOUT2#, TOUT3#). The state of OUT depends on the mode of operation of the timer.

5.2 Interface Signals

5.2.1 CLKIN

CLKIN is an input signal used by all four timers for internal timing reference. This signal can be independent of the 82370 system clock, CLK2. In the following discussion, each "CLK Pulse" is defined as the time period between a rising edge and a falling edge, in that order, of CLKIN.

During the rising edge of CLKIN, the state of GATE is sampled. All new counts are loaded and counters are decremented on the falling edge of CLKIN.

5.2.2 TOUT1, TOUT2#, TOUT3#

TOUT1, TOUT2# and TOUT3# are the external output signals of Timer 1, Timer 2 and Timer 3, respectively. TOUT2# and TOUT3# are the inverted signals of their respective counter outputs, OUT. There is no external output for Timer 0.

If Timer 2 is to be used as a tone generator of a speaker, external buffering must be used to provide sufficient drive capability.

The Outputs of Timer 2 and 3 are dual function pins. The output pin of Timer 2 (TOUT2#/IRQ3#), which is a bidirectional open-collector signal, can also be used as interrupt request input. When the interrupt function is enabled (through the Programmable Interrupt Controller), a LOW on this input will generate an Interrupt Request 3# to the 82370 Programmable Interrupt Controller. This pin has a weak internal pull-up resistor. To use the IRQ3# function, Timer 2 should be programmed so that OUT2 is LOW. Additionally, OUT3 of Timer 3 is connected to an edge detector which will generate an Interrupt Request 0 (IRQ0) to the 82370 after the rising edge of OUT3 (see Figure 5-1).

5.2.3 GATE

GATE is not an externally controllable signal. Rather, it can be software controlled with the Internal Control Port. The state of GATE is always sampled on the rising edge of CLKIN. Depending on the mode of operation, GATE is used to enable/disable counting or trigger the start of an operation.

For Timer 0 and 1, GATE is always enabled (HIGH). For Timer 2 and 3, GATE is connected to Bit 0 and 6, respectively, of an Internal Control Port (at address 61H) of the 82370. After a hardware reset, the state of GATE of Timer 2 and 3 is disabled (LOW).

5.3 Modes of Operation

Each timer can be independently programmed to operate in one of six different modes. Timers are programmed by writing a Control Word into the Control Word Register followed by an Initial Count (see Programming).

The following are defined for use in describing the different modes of operation.

CLK Pulse— A rising edge, then a falling edge, in that order, of CLKIN.

Trigger— A rising edge of a timer's GATE input.

Timer/Counter Loading— The transfer of a count from Count Register (CR) to Count Element (CE).

5.3.1 MODE 0–INTERRUPT ON TERMINAL COUNT

Mode 0 is typically used for event counting. After the Control Word is written, OUT is initially LOW, and will remain LOW until the counter reaches zero. OUT then goes HIGH and remains HIGH until a new count or a new Mode 0 Control Word is written into the counter.

In this mode, GATE=HIGH enables counting; GATE = LOW disables counting. However, GATE has no effect on OUT.

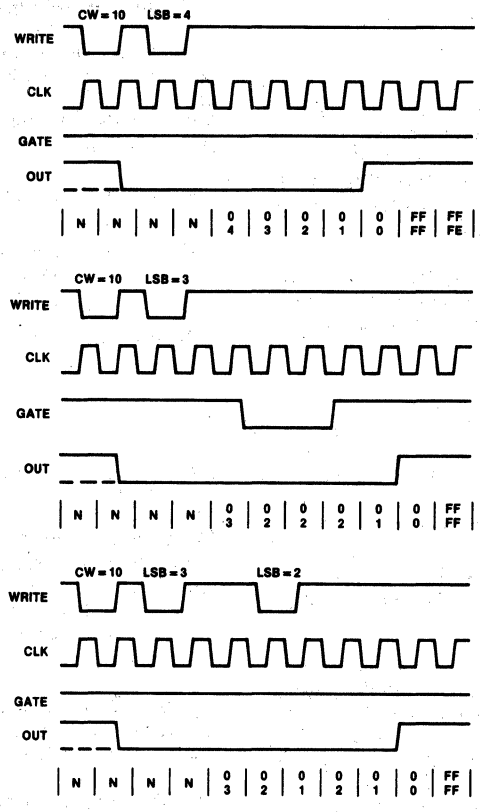
After the Control Word and initial count are written to a timer, the initial count will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not go HIGH until N+1 CLK pulses after the initial count is written.

If a new count is written to the timer, it will be loaded on the next CLK pulse and counting will continue from the new count. If a two-byte count is written, the following happens:

1. Writing the first byte disables counting, OUT is set LOW immediately (i.e. no CLK pulse required).

2. Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the counting sequence to be synchronized by software. Again, OUT does not go HIGH until N+1 CLK pulses after the new count of N is written.



NOTES:
The following conventions apply to all mode timing diagrams.
1. Counters are programmed for binary (not BCD) counting and for reading/writing least significant byte (LSB) only.
2. The counter is always selected (CS# always low).
3. CW stands for "Control Word"; CW = 10 means a control word of 10, Hex is written to the counter.
4. LSB stands for "Least significant byte" of count.
5. Numbers below diagrams are count values.
The lower number is the least significant byte.
The upper number is the most significant byte. Since the counter is programmed to read/write LSB only, the most significant byte cannot be read.
N stands for an undefined count.
Vertical lines show transitions between count values.

Figure 5-3. Mode 0

If an initial count is written while GATE is LOW, the counter will be loaded on the next CLK pulse. When GATE goes HIGH, OUT will go HIGH N CLK pulses later; no CLK pulse is needed to load the counter as this has already been done.

5.3.2 MODE 1–GATE RETRIGGERABLE ONE-SHOT

In this mode, OUT will be initially HIGH. OUT will go LOW on the CLK pulse following a trigger to start the one-shot operation. The OUT signal will then remain LOW until the timer reaches zero. At this point, OUT will stay HIGH until the next trigger comes in. Since the state of GATE signals of Timer 0 and 1 are internally set to HIGH.

After writing the Control Word and initial count, the timer is considered "armed". A trigger results in loading the timer and setting OUT LOW on the next CLK pulse. Therefore, an initial count of N will result in a one-shot pulse width of N CLK cycles. Note

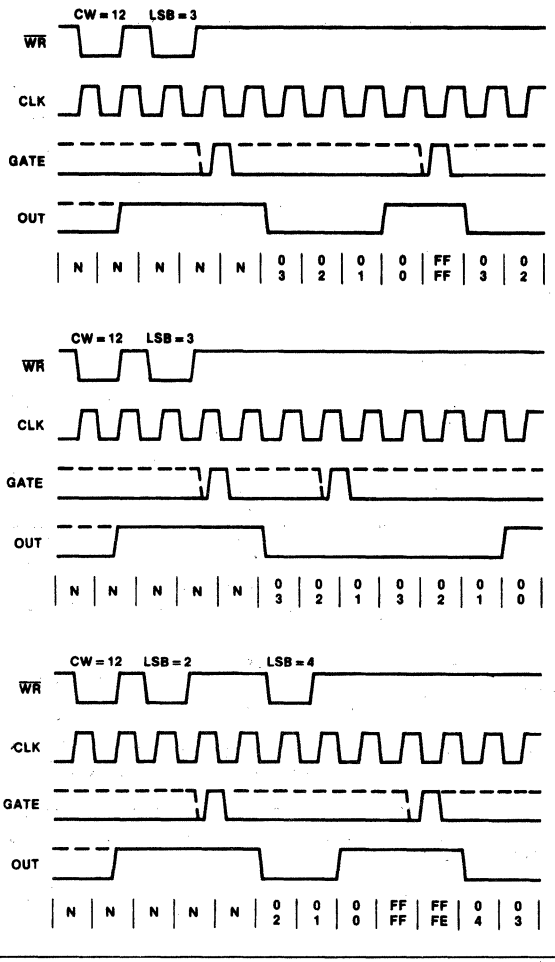


Figure 5-4. Mode 1

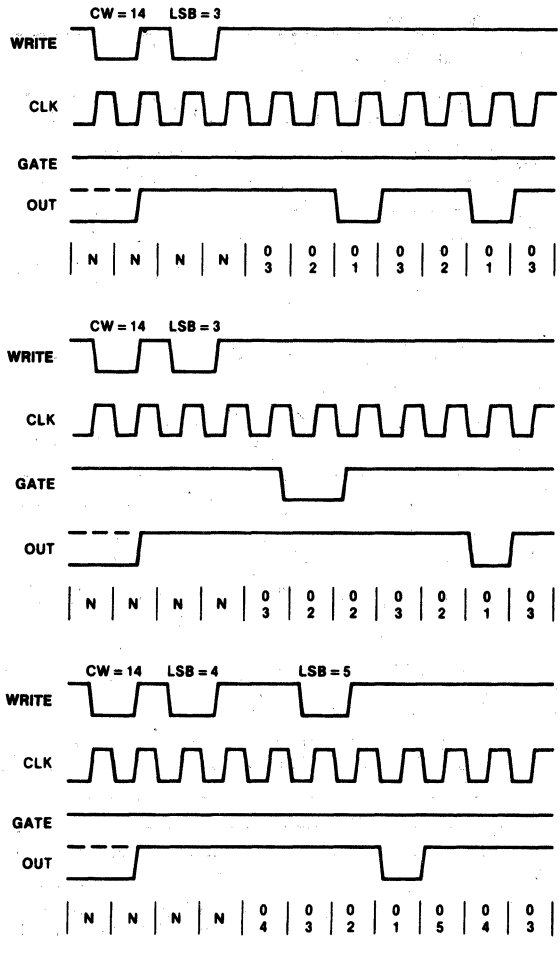
that this one-shot operation is retriggerable; i.e. OUT will remain LOW for N CLK pulses after every trigger. The one-shot operation can be repeated without rewriting the same count into the timer.

If a new count is written to the timer during a one-shot operation, the current one-shot pulse width will not be affected until the timer is retriggered. This is because loading of the new count to CE will occur only when the one-shot is triggered.

5.3.3 MODE 2–RATE GENERATOR

This mode is a divide-by-N counter. It is typically used to generate a Real Time Clock interrupt. OUT will initially be HIGH. When the initial count has decremented to 1, OUT goes LOW for one CLK pulse, then OUT goes HIGH again. Then the timer reloads the initial count and the process is repeated. In other words, this mode is periodic since the same sequence is repeated itself indefinitely. For an initial count of N, the sequence repeats every N CLK cycles.

Similar to Mode 0, GATE = HIGH enables counting, where GATE = LOW disables counting. If GATE goes LOW during an output pulse (LOW), OUT is set HIGH immediately. A trigger (rising edge on GATE) will reload the timer with the initial count on the next CLK pulse. Then, OUT will go LOW (for one CLK pulse) N CLK pulses after the new trigger. Thus, GATE can be used to synchronize the timer.



NOTE:
A GATE transition should not occur one clock prior to terminal count.

Figure 5-5. Mode 2

After writing a Control Word and initial count, the timer will be loaded on the next CLK pulse. OUT goes LOW (for one CLK pulse) N CLK pulses after the initial count is written. This is another way the timer may be synchronized by software.

Writing a new count while counting does not affect the current counting sequence because the new count will not be loaded until the end of the current counting cycle. If a trigger is received after writing a new count but before the end of the current period, the timer will be loaded with the new count on the next CLK pulse after the trigger, and counting will continue with the new count.

5.3.4 MODE 3–SQUARE WAVE GENERATOR

Mode 3 is typically used for Baud Rate generation. Functionally, this mode is similar to Mode 2 except

for the duty cycle of OUT. In this mode, OUT will be initially HIGH. When half of the initial count has expired, OUT goes low for the remainder of the count. The counting sequence will be repeated, thus this mode is also periodic. Note that an initial count of N results in a square wave with a period of N CLK pulses.

The GATE input can be used to synchronize the timer. GATE = HIGH enables counting; GATE = LOW disables counting. If GATE goes LOW while OUT is LOW, OUT is set HIGH immediately (i.e. no CLK pulse is required). A trigger reloads the timer with the initial count on the next CLK pulse.

After writing a Control Word and initial count, the timer will be loaded on the next CLK pulse. This allows the timer to be synchronized by software.

Writing a new count while counting does not affect the current counting sequence. If a trigger is received after writing a new count but before the end of the current half-cycle of the square wave, the timer will be loaded with the new count on the next CLK pulse and counting will continue from the new count. Otherwise, the new count will be loaded at the end of the current half-cycle.

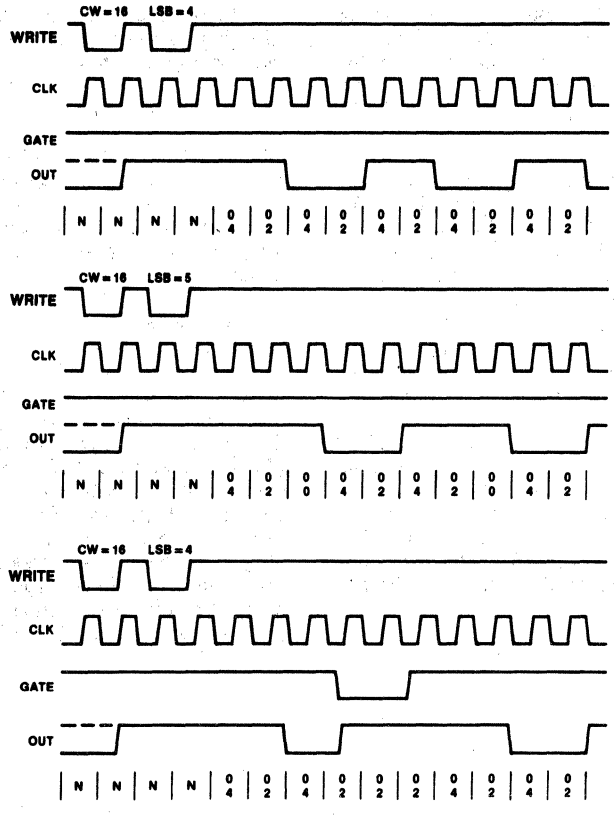
There is a slight difference in operation depending on whether the initial count is EVEN or ODD. The following description is to show exactly how this mode is implemented.

EVEN COUNTS:

OUT is initially HIGH. The initial count is loaded on one CLK pulse and is decremented by two on succeeding CLK pulses. When the count expires (decremented to 2), OUT changes to LOW and the timer is reloaded with the initial count. The above process is repeated indefinitely.

ODD COUNTS:

OUT is initially HIGH. The initial count minus one (which is an even number) is loaded on one CLK pulse and is decremented by two on succeeding CLK pulses. One CLK pulse after the count expires (decremented to 2), OUT goes LOW and the timer is loaded with the initial count minus one again. Succeeding CLK pulses decrement the count by two. When the count expires, OUT goes HIGH immediately and the timer is reloaded with the initial count minus one. The above process is repeated indefinitely. So for ODD counts, OUT will HIGH or $(N+1)/2$ counts and LOW for $(N-1)/2$ counts.



NOTE:
A GATE transition should not occur one clock prior to terminal count.

Figure 5-6. Mode 3

5.3.5 MODE 4–INITIAL COUNT TRIGGERED STROBE

This mode allows a strobe pulse to be generated by writing an initial count to the timer. Initially, OUT will be HIGH. When a new initial count is written into the timer, the counting sequence will begin. When the initial count expires (decremented to 1), OUT will go LOW for one CLK pulse and then go HIGH again.

Again, GATE = HIGH enables counting while GATE = LOW disables counting. GATE has no effect on OUT.

After writing the Control Word and initial count, the timer will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not strobe LOW until N + 1 CLK pulses after initial count is written.

If a new count is written during counting, it will be loaded in the next CLK pulse and counting will continue from the new count.

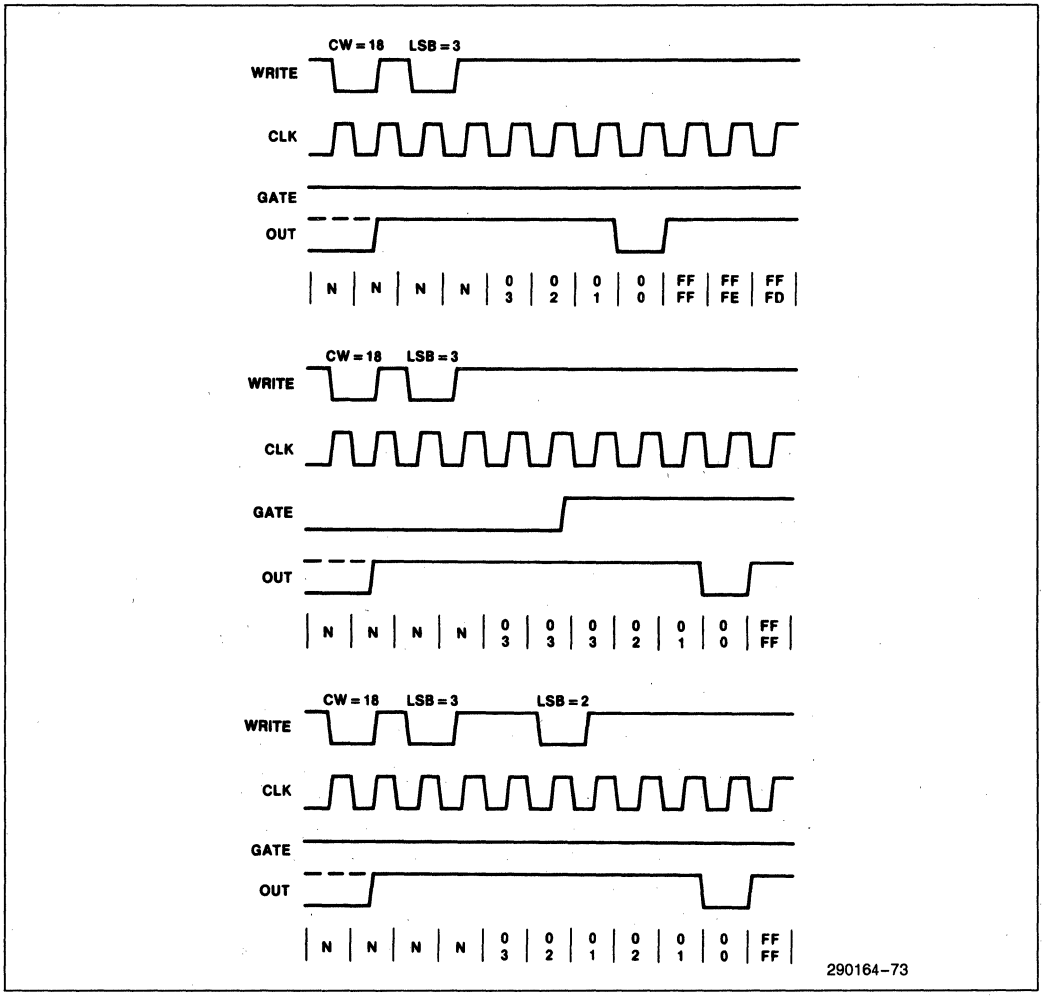


Figure 5-7. Mode 4

If a two-byte count is written, the following will occur:

1. Writing the first byte has no effect on counting.

2. Writing the second byte allows the new count to be loaded on the next CLK pulse.

OUT will strobe LOW N+1 CLK pulses after the new count of N is written. Therefore, when the strobe pulse will occur after a trigger depends on the value of the initial count loaded.

5.3.6 MODE 5–GATE RETRIGGERABLE STROBE

Mode 5 is very similar to Mode 4 except the count sequence is triggered by the gate signal instead of by writing an initial count. Initially, OUT will be HIGH. Counting is triggered by a rising edge of GATE. When the initial count has expired (decremented to 1), OUT will go LOW for one CLK pulse and then go HIGH again.

After loading the Control Word and initial count, the Count Element will not be loaded until the CLK pulse after a trigger. This CLK pulse does not decrement the count. Therefore, for an initial count of N, OUT does not strobe LOW until N+1 CLK pulses after a trigger.

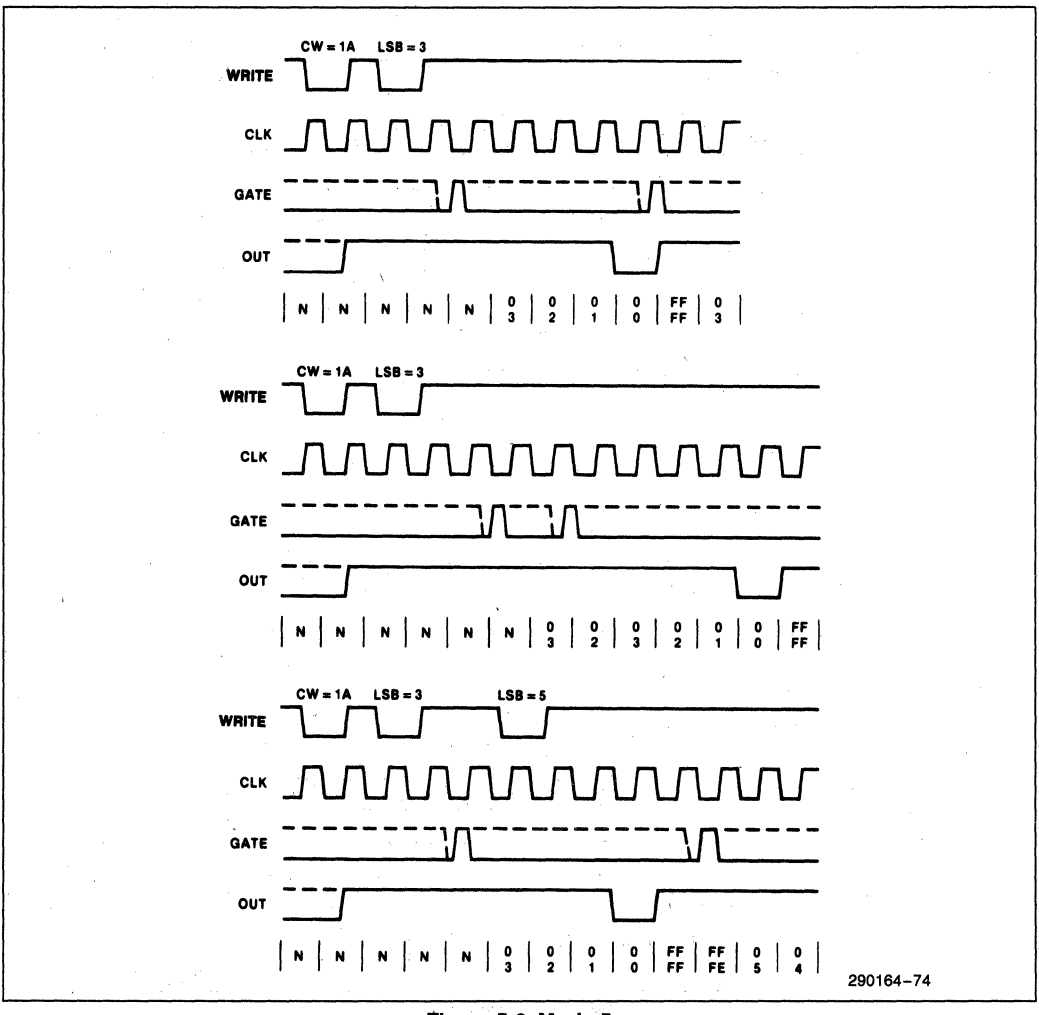


Figure 5-8. Mode 5

The counting sequence is retriggerable. Every trigger will result in the timer being loaded with the initial count on the next CLK pulse.

If the new count is written during counting, the current counting sequence will not be affected. If a trigger occurs after the new count is written but before the current count expires, the timer will be loaded with the new count on the next CLK pulse and a new count sequence will start from there.

5.3.7 OPERATION COMMON TO ALL MODES

5.3.7.1 GATE

The GATE input is always sampled on the rising edge of CLKIN. In Modes 0, 2, 3 and 4, the GATE input is level sensitive. The logic level is sampled on the rising edge of CLKIN. In Modes 1, 2, 3 and 5, the GATE input is rising edge sensitive. In these modes,

Summary of Gate Operations

Mode	GATE LOW or Going LOW	GATE Rising	HIGH
0	Disable count	No Effect	Enable count
1	No Effect	1. Initiate count 2. Reset output after next clock	No Effect
2	1. Disable count 2. Sets output HIGH immediately	Initiate count	Enable count
3	1. Disable count 2. Sets output HIGH immediately	Initiate count	Enable count
4	Disable count	No Effect	Enable count
5	No Effect	Initiate count	No Effect

a rising edge of GATE (trigger) sets an edge sensitive flip-flop in the timer. The flip-flop is reset immediately after it is sampled. This way, a trigger will be detected no matter when it occurs; i.e. a HIGH logic level does not have to be maintained until the next rising edge of CLKIN. Note that in Modes 2 and 3, the GATE input is both edge and level sensitive.

5.3.7.2 Counter

New counts are loaded and counters are decremented on the falling edge of CLKIN. The largest possible initial count is 0. This is equivalent to 2**16 for binary counting and 10**4 for BCD counting.

Note that the counter does not stop when it reaches zero. In Modes 0, 1, 4 and 5, the counter 'wraps around' to the highest count: either FFFF Hex for binary counting or 9999 for BCD counting, and continues counting. Modes 2 and 3 are periodic. The counter reloads itself with the initial count and continues counting from there.

The minimum and maximum initial count in each counter depends on the mode of operation. They are summarized below.

Mode	Min	Max
0	1	0
1	1	0
2	2	0
3	2	0
4	1	0
5	1	0

5.4 Register Set Overview

The Programmable Interval Timer module of the 82370 contains a set of six registers. The port address map of these registers is shown in Table 5-2.

Table 5-2. Timer Register Port Address Map

Port Address	Description
40H	Counter 0 Register (read/write)
41H	Counter 1 Register (read/write)
42H	Counter 2 Register (read/write)
43H	Control Word Register I (Counter 0, 1 & 2) (write-only)
44H	Counter 3 Register (read/write)
45H	Reserved
46H	Reserved
47H	Control Word Register II (Counter 3) (write-only)

5.4.1 COUNTER 0, 1, 2, 3 REGISTER

These four 8-bit registers are functionally identical. They are used to write the initial count value into the respective timer. Also, they can be used to read the latched count value of a timer. Since they are 8-bit registers, reading and writing of the 16-bit initial count must follow the count format specified in the Control Word Registers; i.e. least significant byte only, most significant byte only, or least significant byte then most significant byte (see Programming).

5.4.2 CONTROL WORD REGISTER I & II

There are two Control Word Registers associated with the Timer section. One of the two registers (Control Word Register I) is used to control the operations of Counters 0, 1 and 2 and the other (Control Word Register II) is for Counter 3. The major functions of both Control Word Registers are listed below:

— Select the timer to be programmed.
— Define which mode the selected timer is to operate in.
— Define the count sequence; i.e. if the selected timer is to count as a Binary Counter or a Binary Coded Decimal (BCD) Counter.
— Select the byte access sequence during timer read/write operations; i.e. least significant byte only, most significant only, or least significant byte first, then most significant byte.

Also, the Control Word Registers can be programmed to perform a Counter Latch Command or a Read Back Command which will be described later.

5.5 Programming

5.5.1 INITIALIZATION

Upon power-up or reset, the state of all timers is undefined. The mode, count value, and output of all timers are random. From this point on, how each timer operates is determined solely by how it is programmed. Each timer must be programmed before it can be used. Since the outputs of some timers can generate interrupt signals to the 82370, all timers should be initialized to a known state.

Counters are programmed by writing a Control Word into their respective Control Word Registers. Then, an Initial Count can be written into the corresponding Count Register. In general, the programming procedure is very flexible. Only two conventions need to be remembered:

1. For each timer, the Control Word must be written before the initial count is written.

2. The 16-bit initial count must follow the count format specified in the Control Word (least significant byte only, most significant byte only, or least significant byte first, followed by most significant byte).

Since the two Control Word Registers and the four Counter Registers have separate addresses, and each timer can be individually selected by the appropriate Control Word Register, no special instruction sequence is required. Any programming sequence that follows the conventions above is acceptable.

A new initial count may be written to a timer at any time without affecting the timer's programmed mode in any way. Count sequence will be affected as described in the Modes of Operation section. Note that the new count must follow the programmed count format.

If a timer is previously programmed to read/write two-byte counts, the following precaution applies. A program must not transfer control between writing the first and second byte to another routine which also writes into the same timer. Otherwise, the read/write will result in incorrect count.

Whenever a Control Word is written to a timer, all control logic for that timer(s) is immediately reset (i.e. no CLK pulse is required). Also, the corresponding output in, TOUT#, goes to a known initial state.

5.5.2 READ OPERATION

Three methods are available to read the current count as well as the status of each timer. They are: Read Counter Registers, Counter Latch Command and Read Back Command. Below is a description of these methods.

READ COUNTER REGISTERS

The current count of a timer can be read by performing a read operation on the corresponding Counter Register. The only restriction of this read operation is that the CLKIN of the timers must be inhibited by using external logic. Otherwise, the count may be in the process of changing when it is read, giving an undefined result. Note that since all four timers are sharing the same CLKIN signal, inhibiting CLKIN to read a timer will unavoidably disable the other timers also. This may prove to be impractical. Therefore, it is suggested that either the Counter Latch Command or the Read Back Command can be used to read the current count of a timer.

Another alternative is to temporarily disable a timer before reading its Counter Register by using the GATE input. Depending on the mode of operation, GATE = LOW will disable the counting operation. However, this option is available on Timer 2 and 3 only, since the GATE signals of the other two timers are internally enabled all the time.

COUNTER LATCH COMMAND

A Counter Latch Command will be executed whenever a special Control Word is written into a Control Word Register. Two bits written into the Control Word Register distinguish this command from a 'regular' Control Word (see Register Bit Definition). Also, two other bits in the Control Word will select which counter is to be latched.

Upon execution of this command, the selected counter's Output Latch (OL) latches the count at the time the Counter Latch Command is received. This

count is held in the latch until it is read by the 80376, or until the timer is reprogrammed. The count is then unlatched automatically and the OL returns to "following" the Counting Element (CE). This allows reading the contents of the counters "on the fly" without affecting counting in progress. Multiple Counter Latch Commands may be used to latch more than one counter. Each latched count is held until it is read. Counter Latch Commands do not affect the programmed mode of the timer in any way.

If a counter is latched, and at some time later, it is latched again before the prior latched count is read, the second Counter Latch Command is ignored. The count read will then be the count at the time the first command was issued.

In any event, the latched count must be read according to the programmed format. Specifically, if the timer is programmed for two-byte counts, two bytes must be read. However, the two bytes do not have to be read right after the other. Read/write or programming operations of other timers may be performed between them.

Another feature of this Counter Latch Command is that read and write operations of the same timer may be interleaved. For example, if the timer is programmed for two-byte counts, the following sequence is valid.

1. Read least significant byte.
2. Write new least significant byte.
3. Read most significant byte.
4. Write new most significant byte.

If a timer is programmed to read/write twc-byte counts, the following precaution applies. A program must not transfer control between reading the first and second byte to another routine which also reads from that same timer. Otherwise, an incorrect count will be read.

READ BACK COMMAND

The Read Back Command is another special Command Word operation which allows the user to read the current count value and/or the status of the selected timer(s). Like the Counter Latch Command, two bits in the Command Word identify this as a Read Back Command (see Register Bit Definition).

The Read Back Command may be used to latch multiple counter Output Latches (OL's) by selecting more than one timer within a Command Word. This single command is functionally equivalent to several Counter Latch Commands, one for each counter to be latched. Each counter's latched count will be held until it is read by the 80376 or until the timer is reprogrammed. The counter is automatically unlatched when read, but other counters remain latched until they are read. If multiple Read Back commands are issued to the same timer without reading the count, all but the first are ignored; i.e. the count read will correspond to the very first Read Back Command issued.

As mentioned previously, the Read Back Command may also be used to latch status information of the selected timer(s). When this function is enabled, the status of a timer can be read from the Counter Register after the Read Back Command is issued. The status information of a timer includes the following:

1. Mode of timer:

   This allows the user to check the mode of operation of the timer last programmed.

2. State of TOUT pin of the timer:

   This allows the user to monitor the counter's output pin via software, possibly eliminating some hardware from a system.

3. Null Count/Count available:

   The Null Count Bit in the status byte indicates if the last count written to the Count Register (CR) has been loaded into the Counting Element (CE). The exact time this happens depends on the mode of the timer and is described in the Programming section. Until the count is loaded into the Counting Element (CE), it cannot be read from the timer. If the count is latched or read before this occurs, the count value will not reflect the new count just written.

If multiple status latch operations of the timer(s) are performed without reading the status, all but the first command are ignored; i.e. the status read in will correspond to the first Read Back Command issued.

Both the current count and status of the selected timer(s) may be latched simultaneously by enabling both functions in a single Read Back Command. This is functionally the same as issuing two separate Read Back Commands at once. Once again, if multiple read commands are issued to latch both the count and status of a timer, all but the first command will be ignored.

If both count and status of a timer are latched, the first read operation of that timer will return the latched status, regardless of which was latched first. The next one or two (if two count bytes are to be read) read operations return the latched count. Note that subsequent read operations on the Counter Register will return the unlatched count (like the first read method discussed).

5.6 Register Bit Definitions

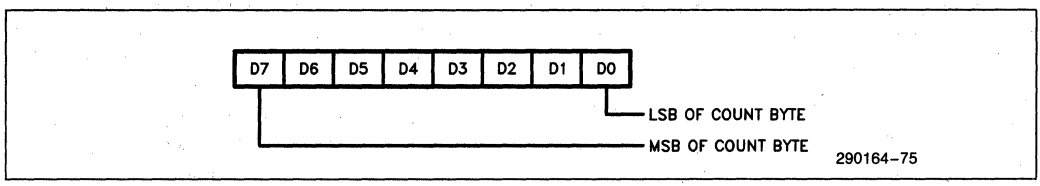
COUNTER 0, 1, 2, 3 REGISTER (READ/WRITE)

Port Address	Description
40H	Counter 0 Register (read/write)
41H	Counter 1 Register (read/write)
42H	Counter 2 Register (read/write)
44H	Counter 3 Register (read/write)
45H	Reserved
46H	Reserved

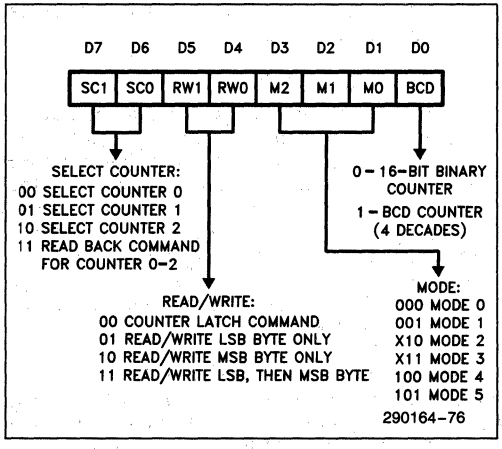
Note that these 8-bit registers are for writing and reading of one byte of the 16-bit count value, either the most significant or the least significant byte.

CONTROL WORD REGISTER I & II (WRITE-ONLY)

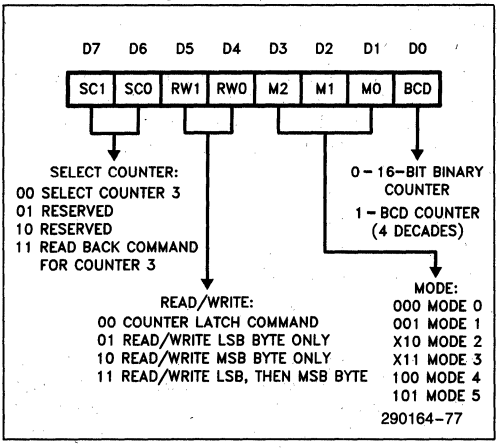
Port Address	Description
43H	Control Word Register I (Counter 0, 1, 2 (write-only)
47H	Control Word Register II (Counter 3) (write-only)



Control Word Register I

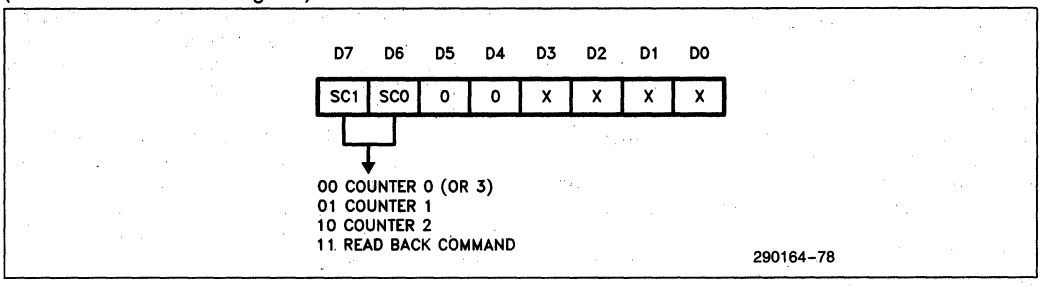


Control Word Register II



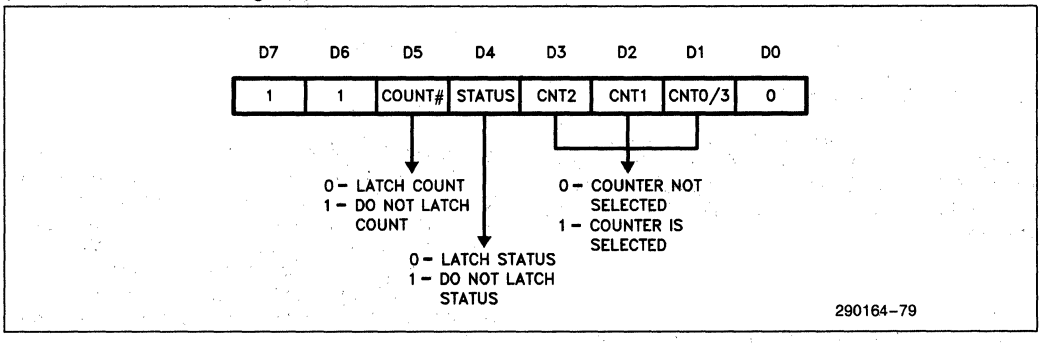
COUNTER LATCH COMMAND FORMAT

(Write to Control Word Register)



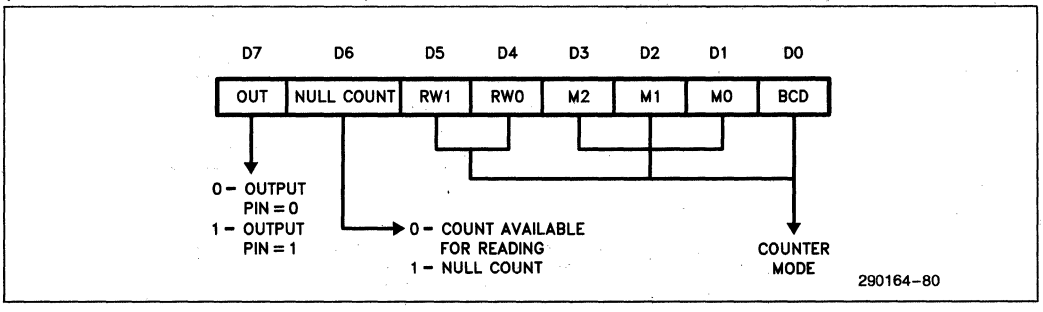
READ BACK COMMAND FORMAT

(Write to Control Word Register)



STATUS FORMAT

(Returned from Read Back Command)



6.0 WAIT STATE GENERATOR

6.1 Functional Description

The 82370 contains a programmable Wait State Generator which can generate a pre-programmed number of wait states during both CPU and DMA initiated bus cycles. This Wait State Generator is capable of generating 1 to 16 wait states in non-pipelined mode, and 0 to 15 wait states in pipelined mode. Depending on the bus cycle type and the two Wait State Control inputs (WSC 0–1), a pre-programmed number of wait states in the selected Wait State Register will be generated.

The Wait State Generator can also be disabled to allow the use of devices capable of generating their own READY# signals. Figure 6-1 is a block diagram of the Wait State Generator.

6.2 Interface Signals

The following describes the interface signals which affect the operation of the Wait State Generator. The READY#, WSC0 and WSC1 signals are inputs. READYO# is the ready output signal to the host processor.

6.2.1 READY#

READY# is an active LOW input signal which indicates to the 82370 the completion of a bus cycle. In the Master mode (e.g. 82370 initiated DMA transfer), this signal is monitored to determine whether a peripheral or memory needs wait states inserted in the current bus cycle. In the Slave mode, it is used (together with the ADS# signal) to trace CPU bus cycles to determine if the current cycle is pipelined.

6.2.2 READYO#

READYO# (Ready Out#) is an active LOW output signal and is the output of the Wait State Generator. The number of wait states generated depends on the WSC(0–1) inputs. Note that special cases are handled for access to the 82370 internal registers and for the Refresh cycles. For 82370 internal register access, READYO# will be delayed to take into the command recovery time of the register. One or more wait states will be generated in a pipelined cycle. During refresh, the number of wait states will be determined by the preprogrammed value in the Refresh Wait State Register.

In the simplest configuration, READYO# can be connected to the READY# input of the 82370 and the 80376 CPU. This is, however, not always the case. If external circuitry is to control the READY# inputs as well, additional logic will be required (see Application Issues).

6.2.3 WSC(0–1)

These two Wait State Control inputs, together with the M/IO# input, select one of the three pre-programmed 8-bit Wait State Registers which determines the number of wait states to be generated. The most significant half of the three Wait State Registers corresponds to memory accesses, the least significant half to I/O accesses. The combination WSC(0–1) = 11 disables the Wait State Generator.

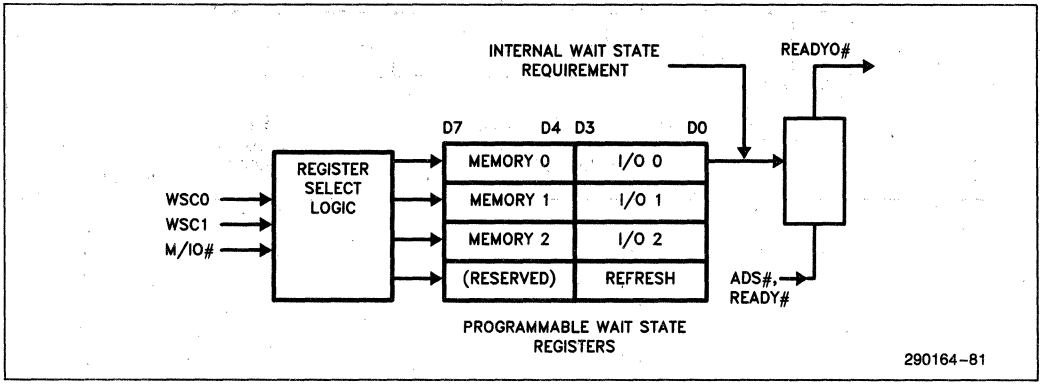


Figure 6-1. Wait State Generator Block Diagram

6.3 Bus Function

6.3.1 WAIT STATES IN NON-PIPELINED CYCLE

The timing diagram of two typical non-pipelined cycles with 82370 generated wait states is shown in Figure 6-2. In this diagram, it is assumed that the internal registers of the 82370 are not addressed. During the first T2 state of each bus cycle, the Wait State Control and the M/IO# inputs are sampled to determine which Wait State Register (if any) is selected. If the WSC inputs are active (i.e. not both are driven HIGH), the pre-programmed number of wait states corresponding to the selected Wait State Register will be requested. This is done by driving the READYO# output HIGH during the end of each T2 state.

The WSC (0–1) inputs need only be valid during the very first T2 state of each non-pipelined cycle. As a general rule, the WSC inputs are sampled on the rising edge of the next clock (82384 CLK) after the last state when ADS# (Address Status) is asserted.

The number of wait states generated depends on the type of bus cycle, and the number of wait states requested. The various combinations are discussed below.

1. Access the 82370 internal registers: 2 to 5 wait states, depending upon the specific register addressed. Some back-to-back sequences to the Interrupt Controller will require 7 wait states.

2. Interrupt Acknowledge to the 82370: 5 wait states.

3. Refresh: As programmed in the Refresh Wait State Register (see Register Set Overview). Note that if WCS (0–1) = 11, READYO# will stay inactive.

4. Other bus cycles: Depending on WCS (0–1) and M/IO# inputs, these inputs select a Wait State Register in which the number of wait states will be equal to the pre-programmed wait state count in the register plus 1. The Wait State Register selection is defined as follows (Table 6-1).

Table 6-1. Wait State Register Selection

M/IO#	WSC(0–1)	Register Selected
0	00	WAIT REG 0 (I/O half)
0	01	WAIT REG 1 (I/O half)
0	10	WAIT REG 2 (I/O half)
1	00	WAIT REG 0 (MEM half)
1	01	WAIT REG 1 (MEM half)
1	10	WAIT REG 2 (MEM half)
X	11	Wait State Gen. Disabled

The Wait State Control signals, WSC (0–1), can be generated with the address decode and the Read/Write control signals as shown in Figure 6-3.

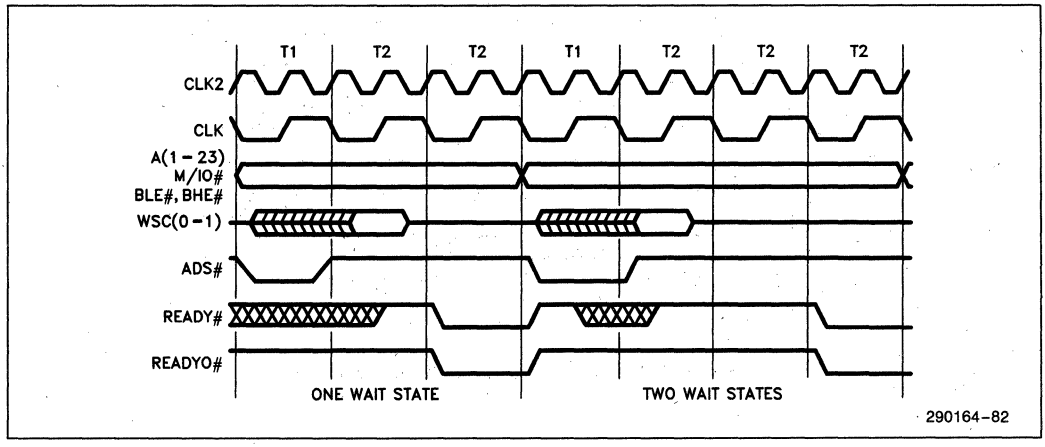


Figure 6-2. Wait States in Non-Pipelined Cycles

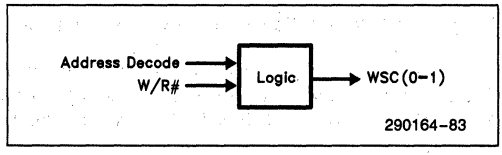


Figure 6-3. WSC (0-1) Generation

Note that during HALT and SHUTDOWN, the number of wait states will depend on the WSC (0-1) inputs, which will select the memory half of one of the Wait State Registers (see CPU Reset and Shutdown Detect).

6.3.2 WAIT STATES IN PIPELINED CYCLES

The timing diagram of two typical pipelined cycles with 82370 generated wait states is shown in Figure 6-4. Again, in this diagram, it is assumed that the 82370 internal registers are not addressed. As defined in the timing of the 80376 processor, the Address (A1-23), Byte Enable (BHE#, BLE#), and other control signals (M/IO#, ADS#) are asserted one T-state earlier than in a non-pipelined cycle; i.e. they are asserted at T2P. Similar to the non-pipelined case, the Wait State Control (WSC) inputs are sampled in the middle of the state after the last state the ADS# signal is asserted. Therefore, the WSC inputs should be asserted during the T1P state of each pipelined cycle (which is one T-state earlier than in the non-pipelined cycle).

The number of wait states generated in a pipelined cycle is selected in a similar manner as in the non-pipelined case discussed in the previous section. The only difference here is that the actual number of wait states generated will be one less than that of the non-pipelined cycle. This is done automatically by the Wait State Generator.

6.3.3 EXTENDING AND EARLY TERMINATING BUS CYCLE

The 82370 allows external logic to either add wait states or cause early termination of a bus cycle by controlling the READY# input to the 82370 and the host processor. A possible configuration is shown in Figure 6-5.

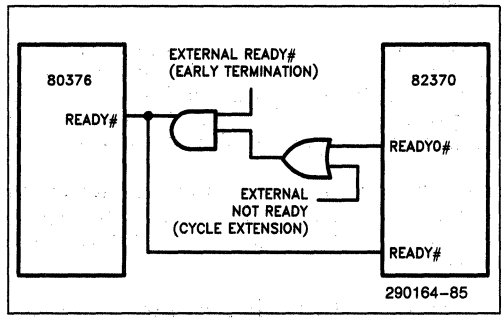


Figure 6-5. External 'READY' Control Logic

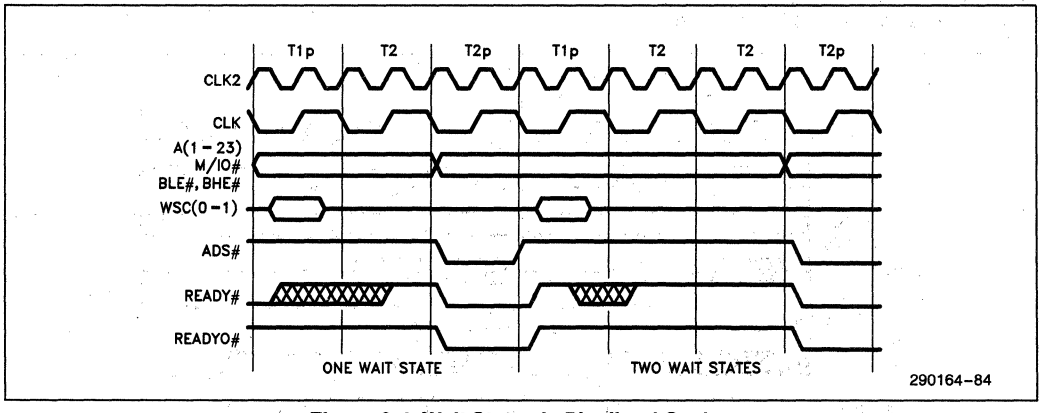


Figure 6-4. Wait States in Pipelined Cycles

The EXT. RDY# (External Ready) signal of Figure 6-5 allows external devices to cause early termination of a bus cycle. When this signal is asserted LOW, the output of the circuit will also go LOW (even though the READYO# of the 82370 may still be HIGH). This output is fed to the READY# input of the 80376 and the 82370 to indicate the completion of the current bus cycle.

Similarly, the EXT. NOT READY (External Not Ready) signal is used to delay the READY# input of the processor and the 82370. As long as this signal is driven HIGH, the output of the circuit will drive the READY# input HIGH. This will effectively extend the duration of a bus cycle. However, it is important to note that if the two-level logic is not fast enough to satisfy the READY# setup time, the OR gate should be eliminated. Instead, the 82370 Wait State Generator can be disabled by driving both WSC (0–1) HIGH. In this case, the addressed memory or I/O device should activate the external READY# input whenever it is ready to terminate the current bus cycle.

Figures 6-6 and 6-7 show the timing relationships of the ready signals for the early termination and extension of the bus cycles. Section 6-7, Application Issues, contains a detailed timing analysis of the external circuit.

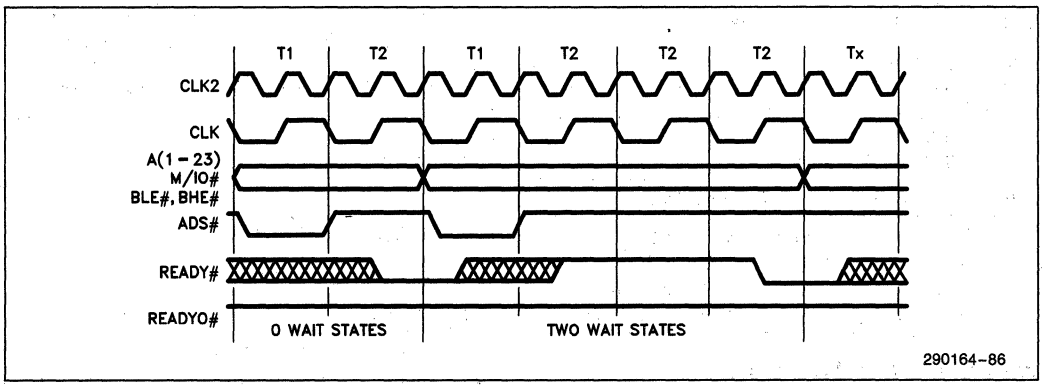


Figure 6-6. Early Termination of Bus Cycle By 'READY#'

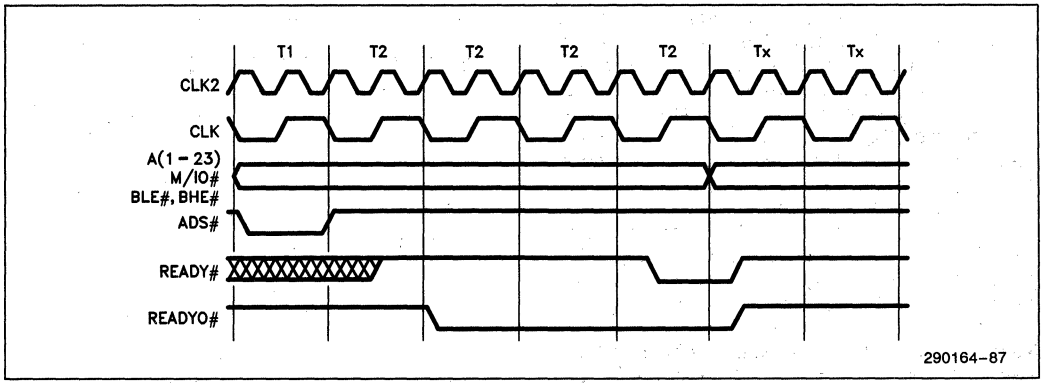


Figure 6-7. Extending Bus Cycle by 'READY#'

Due to the following implications, it should be noted that early termination of bus cycles in which 82370 internal registers are accessed is not recommended.

1. Erroneous data may be read from or written into the addressed register.

2. The 82370 must be allowed to recover either before HLDA (Hold Acknowledge) is asserted or before another bus cycle into an 82370 internal register is initiated.

The recovery time, in clock periods, equals the remaining wait states that were avoided plus 4.

6.4 Register Set Overview

Altogether, there are four 8-bit internal registers associated with the Wait State Genertor. The port address map of these registers is shown below in Table 6-2. A detailed description of each follows.

Table 6-2. Register Address Map

Port Address	Description
72H	Wait State Reg 0 (read/write)
73H	Wait State Reg 1 (read/write)
74H	Wait State Reg 2 (read/write)
75H	Ref. Wait State Reg (read/write)

WAIT STATE REGISTER 0, 1, 2

These three 8-bit read/write registers are functionally identical. They are used to store the pre-programmed wait state count. One half of each register contains the wait state count for I/O accesses while the other half contains the count for memory accesses. The total number of wait states generated will depend on the type of bus cycle. For a non-pipelined cycle, the actual number of wait states requested is equal to the wait state count plus 1. For a pipelined cycle, the number of wait states will be equal to the wait state count in the selected register. Therefore, the Wait State Generator is capable of generating 1 to 16 wait states in non-pipelined mode, and 0 to 15 wait states in pipelined mode.

Note that the minimum wait state count in each register is 0. This is equivalent to 0 wait states for a pipelined cycle and 1 wait state for a non-pipelined cycle.

REFRESH WAIT STATE REGISTER

Similar to the Wait State Registers discussed above, this 4-bit register is used to store the number of wait states to be generated during a DRAM refresh cycle.

Note that the Refresh Wait State Register is not selected by the WSC inputs. It will automatically be chosen whenever a DRAM refresh cycle occurs. If the Wait State Generator is disabled during the refresh cycle (WSC (0–1) = 11), READYO# will stay inactive and the Refresh Wait State Register is ignored.

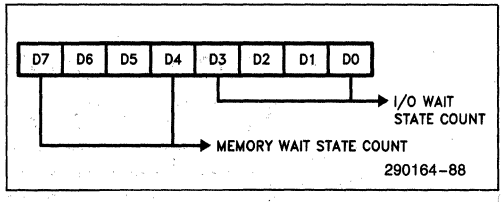
6.5 Programming

Using the Wait State Generator is relatively straightforward. No special programming sequence is required. In order to ensure the expected number of wait states will be generated when a register is selected, the registers to be used must be programmed after power-up by writing the appropriate wait state count into each register. Note that upon hardware reset, all Wait State Registers are initialized with the value FFH, giving the maximum number of wait states possible. Also, each register can be read to check the wait state count previously stored in the register.

6.6 Register Bit Definition

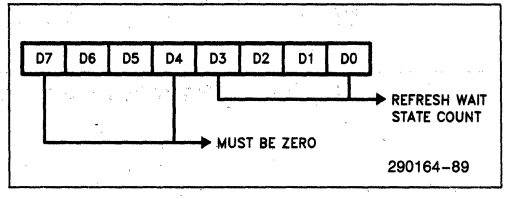
WAIT STATE REGISTER 0, 1, 2

Port Address	Description
72H	Wait State Register 0 (read/write)
73H	Wait State Register 1 (read/write)
74H	Wait State Register 2 (read/write)



REFRESH WAIT STATE REGISTER

Port Address: 75H (Read/Write)



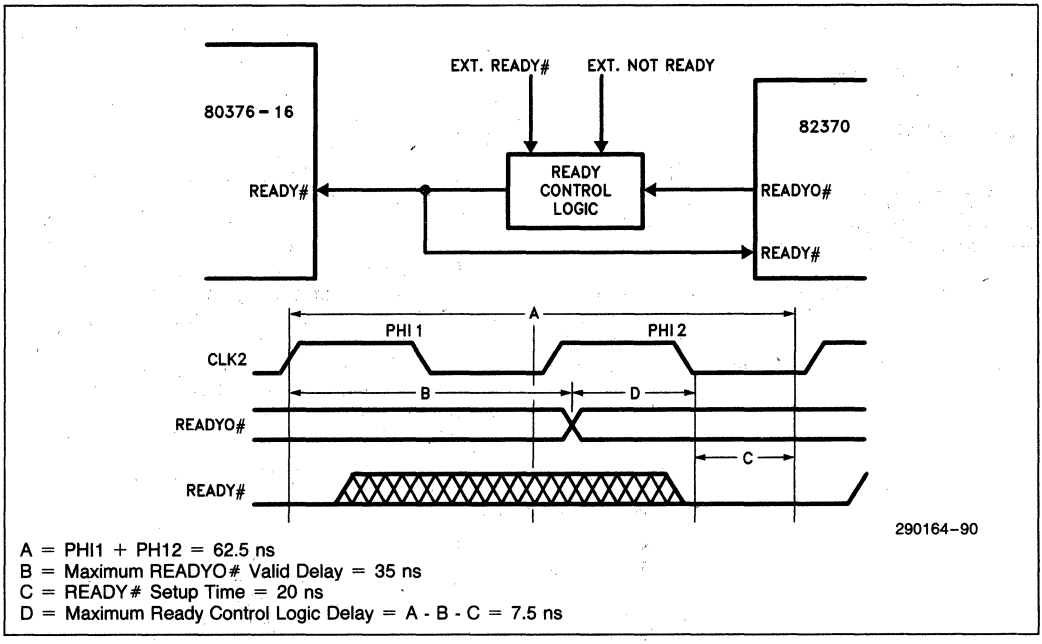
6.7 Application Issues

6.7.1 EXTERNAL 'READY' CONTROL LOGIC

As mentioned in section 6.3.3, wait state cycles generated by the 82370 can be terminated early or extended longer by means of additional external logic (see Figure 6-5). In order to ensure that the READY# input timing requirement of the 80376 and the 82370 is satisfied, special care must be taken when designing this external control logic. This section addresses the design requirements.

A simplified block diagram of the external logic along with the READY# timing diagram is shown in Figure 6-8. The purpose is to determine the maximum delay time allowed in the external control logic in order to satisfy the READY# setup time.

First, it will be assumed that the 80376 is running at 16 MHz (i.e. CLK2 is 32 MHz). Therefore, one bus state (two CLK2 periods) will be equivalent to 62.5 ns. According to the AC specifications of the 82370, the maximum delay time for valid READYO# signal is 31 ns after the rising edge of CLK2 in the beginning of T2 (for non-pipelined cycle) or T2P (for pipelined cycle). Also, the minimum READY# setup time of the 80376 and the 82370 should be 19 ns before the rising edge of CLK2 at the beginning of the next bus state. This limits the total delay time for the external READY# control logic to be 12.5 ns (62.5–31–19) in order to meet the READY# setup timing requirement.



A = PHI1 + PH12 = 62.5 ns
B = Maximum READYO# Valid Delay = 35 ns
C = READY# Setup Time = 20 ns
D = Maximum Ready Control Logic Delay = A - B - C = 7.5 ns

Figure 6-8. 'READY' Timing Consideration

7.0 DRAM REFRESH CONTROLLER

7.1 Functional Description

The 82370 DRAM Refresh Controller consists of a 24-bit Refresh Address Counter and Refresh Request logic for DRAM refresh operations (see Figure 7-1). TIMER 1 can be used as a trigger signal to the DRAM Refresh Request logic. The Refresh Bus Size can be programmed to be 8- or 16-bit wide. Depending on the Refresh Bus Size, the Refresh Address Counter will be incremented with the appropriate value after every refresh cycle. The internal logic of the 82370 will give the Refresh operation the highest priority in the bus control arbitration process. Bus control is not released and re-requested if the 82370 is already a bus master.

7.2 Interface Signals

7.2.1 TOUT1/REF#

The dual function output pin of TIMER 1 (TOUT1/REF#) can be programmed to generate DRAM Refresh signal. If this feature is enabled, the rising edge of TIMER 1 output (TOUT1#) will trigger the DRAM Refresh Request logic. After some delay for gaining access of the bus, the 82370 DRAM Controller will generate a DRAM Refresh signal by driving REF# output LOW. This signal is cleared after the refresh cycle has taken place, or by a hardware reset.

If the DRAM Refresh feature is disabled, the TOUT1/REF# output pin is simply the TIMER 1 output. Detailed information of how TIMER 1 operates is discussed in section 6—Programmable Interval Timer, and will not be repeated here.

7.3 Bus Function

7.3.1 ARBITRATION

In order to ensure data integrity of the DRAMs, the 82370 gives the DRAM Refresh signal the highest priority in the arbitration logic. It allows DRAM Refresh to interrupt DMA in progress in order to perform the DRAM Refresh cycle. The DMA service will be resumed after the refresh is done.

In case of a DRAM Refresh during a DMA process, the cascaded device will be requested to get off the bus. This is done by de-asserting the EDACK signal. Once DREQn goes inactive, the 82370 will perform the refresh operation. Note that the DMA controller does not completely relinquish the system bus during refresh. The Refresh Generator simply "steals" a bus cycle between DMA accesses.

Figure 7-2 shows the timing diagram of a Refresh Cycle. Upon expiration of TIMER 1, the 82370 will try to take control of the system bus by asserting HOLD. As soon as the 82370 see HLDA go active, the DRAM Refresh Cycle will be carried out by activating the REF# signal as well as the address and control signals on the system bus (Note that REF# will not be active until two CLK periods HLDA is asserted). The address bus will contain the 24-bit ad-

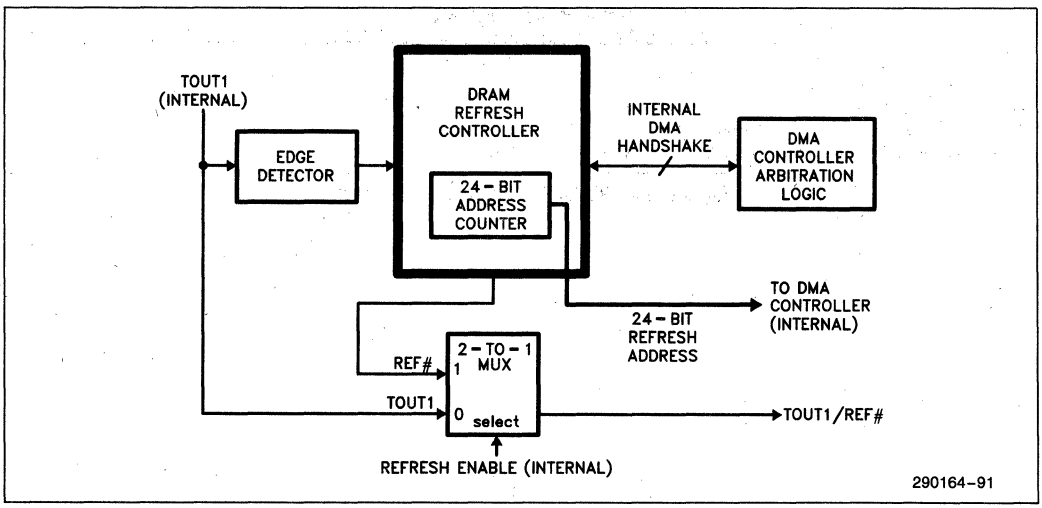


Figure 7-1. DRAM Refresh Controller

dress currently in the Refresh Address Counter. The control signals are driven the same way as in a Memory Read cycle. This "read" operation is complete when the READY# signal is driven LOW. Then, the 82370 will relinquish the bus by de-asserting HOLD. Typically, a Refresh Cycle without wait states will take five bus states to execute. If "n" wait states are added, the Refresh Cycle will last for five plus "n" bus states.

How often the Refresh Generator will initiate a refresh cycle depends on the frequency of CLKIN as will as TIMER 1's programmed mode of operation. For this specific application, TIMER 1 should be programmed to operate in Mode 2 to generate a constant clock rate. See section 6—Programmable Interval Timer for more information on programming the timer. One DRAM Refresh Cycle will be generated each time TIMER 1 expires (when TOUT1 changes from LOW to HIGH).

The Wait State Generator can be used to insert wait states during a refresh cycle. The 82370 will automatically insert the desired number of wait states as programmed in the Refresh Wait State Register (see Wait State Generator).

7.4 Modes of Operation

7.4.1 WORD SIZE AND REFRESH ADDRESS COUNTER

The 82370 supports 8- and 16-bit refresh cycle. The bus width during a refresh cycle is programmable (see Programming). The bus size can be programmed via the Refresh Control Register (see Register Overview). If the DRAM bus size is 8- or 16-bits, the Refresh Address Counter will be incremented by 1 or 2, respectively.

The Refresh Address Counter is cleared by a hardware reset.

7.5 Register Set Overview

The Refresh Generator has two internal registers to control its operation. They are the Refresh Control Register and the Refresh Wait State Register. Their port address map is shown in Table 7-1 below.

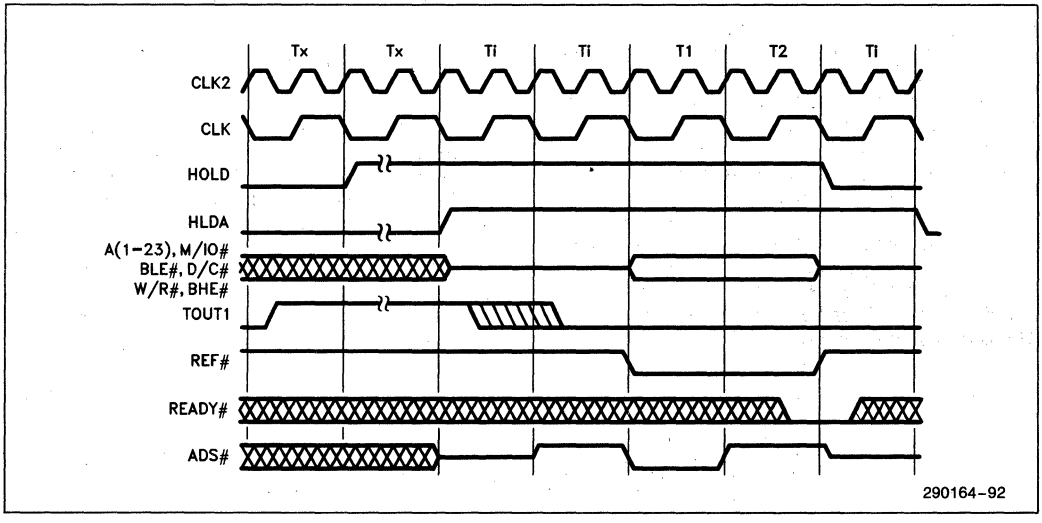


Figure 7-2. 82370 Refresh Cycle

Table 7-1. Register Address Map

Port Address	Description
1CH	Refresh Control Reg. (read/write)
75H	Ref. Wait State Reg. (read/write)

The Refresh Wait State Register is not part of the Refresh Generator. It is only used to program the number of wait states to be inserted during a refresh cycle. This register is discussed in detailed in section 7 (Wait State Generator) and will not be repeated here.

REFRESH CONTROL REGISTER

This 2-bit register serves two functions. First, it is used to enable/disable the DRAM Refresh function output. If disabled, the output of TIMER 1 is simply used as a general purpose timer. The second function of this register is to program the DRAM bus size for the refresh operation. The programmed bus size also determines how the Refresh Address Counter will be incremented after each refresh operation.

7.6 Programming

Upon hardware reset, the DRAM Refresh function is disabled (the Refresh Control Register is cleared). The following programming steps are needed before the Refresh Generator can be used. Since the rate of refresh cycles depends on how TIMER 1 is programmed, this timer must be initialized with the desired mode of operation as well as the correct refresh interval (see Programming Interval Timer). Whether or not wait states are to be generated during a refresh cycle, the Refresh Wait State Register must also be programmed with the appropriate value. Then, the DRAM Refresh feature must be enabled and the DRAM bus width should be defined. These can be done in one step by writing the appropriate control word into the Refresh Control Register (see Register Bit Definition). After these steps are done, the refresh operation will automatically be invoked by the Refresh Generator upon expiration of Timer 1.

In addition to the above programming steps, it should be noted that after reset, although the TOUT1/REF# becomes the Time 1 output, the state of this pin in undefined. This is because the Timer module has not been initialized yet. Therefore, if this output is used as a DRAM Refresh signal, this pin should be disqualified by external logic until the Refresh function is enabled. One simple solution is to logically AND this output with HLDA, since HLDA should not be active after reset.

7.7 Register Bit Definition

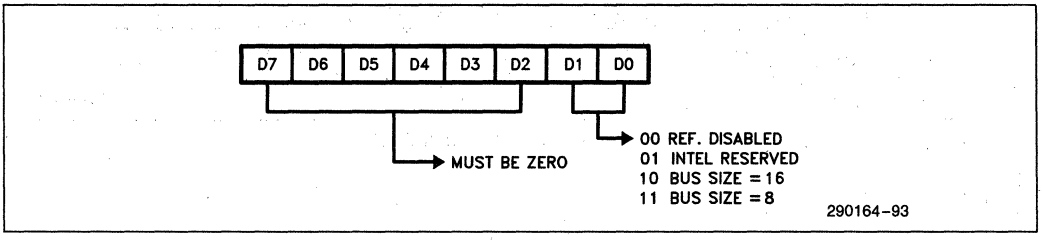
REFRESH CONTROL REGISTER

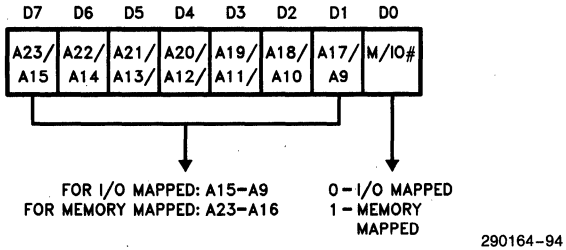
Port Address: 1CH (Read/Write)

8.0 RELOCATION REGISTER AND ADDRESS DECODE

8.1 Relocation Register

All the integrated peripheral devices in the 82370 are controlled by a set of internal registers. These registers span a total of 256 consecutive address locations (although not all the 256 locations are used). The 82370 provides a Relocation Register which allows the user to map this set of internal registers into either the memory or I/O address space. The function of the Relocation Register is to define the base address of the internal register set of the 82370 as well as if the registers are to be memory- or I/O-mapped. The format of the Relocation Register is depicted in Figure 9-1.





Port Address: 7FH (Read/Write)

Figure 8-1. Relocation Register

Note that the Relocation Register is part of the internal register set of the 82370. It has a port address of 7FH. Therefore, any time the content of the Relocation Register is changed, the physical location of this register will also be moved. Upon reset of the 82370, the content of the Relocation Register will be cleared. This implies that the 82370 will respond to its I/O addresses in the range of 0000H to 00FFH.

8.1.1 I/O-MAPPED 82370

As shown in the figure, Bit 0 of the Relocation Register determines whether the 82370 registers are to be memory-mapped or I/O mapped. When Bit 0 is set to '0', the 82370 will respond to I/O Addresses. Address signals BHE#, BLE#, A1–A7 will be used to select one of the internal registers to be accessed. Bit 1 to Bit 7 of the Relocation Register will correspond to A9 to A15 of the Address bus, respectively. Together with A8 implied to be '0', A15 to A8 will be fully decoded by the 82370. The following shows how the 82370 is mapped into the I/O address space.

Example

Relocation Register = 11001110 (0CEH)

82370 will respond to I/O address range from 0CE00H to 0CEFFH.

Therefore, this I/O mapping mechanism allows the 82370 internal registers to be located on any even, contiguous, 256 byte boundary of the system I/O space.

8.1.2 MEMORY-MAPPED 82370

When Bit 0 of the Relocation Register is set to '1', the 82370 will respond to memory addresses. Again, Address signals BHE#, BLE#, A1–A7 will be used to select one of the internal registers to be accessed. Bit 1 to Bit 7 of the Relocation Register will correspond to A17–A23, respectively. A16 is assumed to be '0', and A8–A15 are ignored. Consider the following example.

Example

Relocation Register = 10100111 (0A7H)

The 82370 will respond to memory addresses in the range of A6XX00H to A60XXFFH (where 'X' is don't care).

This scheme implies that the internal registers can be located in any even, contiguous, 2**16 byte page of the memory space.

8.2 Address Decoding

As mentioned previously, the 82370 internal registers do not occupy the entire contiguous 256 address locations. Some of the locations are 'unoccupied'. The 82370 always decodes the lower 8 address signals (BHE#, BLE#, A1–A7) to determine if any one of its registers is being accessed. If the address does not correspond to any of its registers, the 82370 will not respond. This allows external devices to be located within the 'holes' in the 82370 address space. Note that there are several unused addresses reserved for future Intel peripheral devices.

8.3 Chip-Select (CHPSEL#)

The Chip-Select signal (CHPSEL#) will go active when the 82370 is addressed in a Slave bus

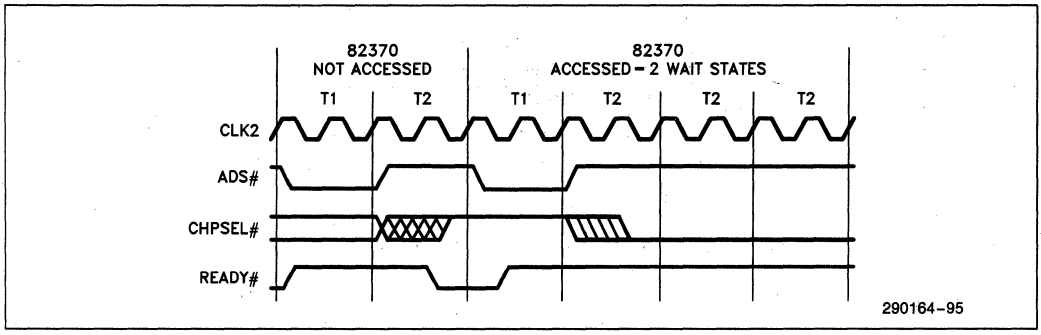


Figure 8-2. CHPSEL# Timing

cycle (either read or write), or in an interrupt acknowledge cycle in which the 82370 will drive the Data Bus. For a given bus cycle, CHPSEL# becomes active and valid in the first T2 (in a non-pipelined cycle) or in T1P (in a pipelined cycle). It will stay valid until the cycle is terminated by READY# driven active. As CHPSEL# becomes valid well before the 82370 drives the Data Bus, it can be used to control the transceivers that connect the local CPU bus to the system bus. The timing diagram of CHPSEL# is shown in Figure 8-2.

9.0 CPU RESET AND SHUTDOWN DETECT

The 82370 will activate the CPURST signal to reset the host processor when one of the following conditions occurs:

— 82370 RESET is active;

— 82370 detects a 80376 Shutdown cycle (this feature can be disabled);

— CPURST software command is issued to 80376.

Whenever the CPURST signal is activated, the 82370 will reset its own internal Slave-Bus state machine.

9.1 Hardware Reset

Following a hardware reset, the 82370 will assert its CPURST output to reset the host processor. This output will stay active for as long as the RESET input is active. During a hardware reset, the 82370 internal registers will be initialized as defined in the corresponding functional descriptions.

9.2 Software Reset

CPURST can be generated by writing the following bit pattern into 82370 register location 64H.

D7	. . .	D0
1	1 1 1 X X X	0

The Write operation into this port is considered as an 82370 access and the internal Wait State Generator will automatically determine the required number of wait states. The CPURST will be active following the completion of the Write cycle to this port. This signal will last for 62 CLK2 periods. The 82370 should not be accessed until the CPURST is deactivated.

This internal port is Write-Only and the 82370 will not respond to a Read operation to this location. Also, during a software reset command, the 82370 will reset its Slave-Bus state machine. However, its internal registers remain unchanged. This allows the operating system to distinguish a 'warm' reset by reading any 82370 internal register previously programmed for a non-default value. The Diagnostic registers can be used for this purpose (see Internal Control and Diagnostic Ports).

9.3 Shutdown Detect

The 82370 is constantly monitoring the Bus Cycle Definition signals (M/IO#, D/C#, W/R#) and is able to detect when the 80376 is in a Shutdown bus cycle. Upon detection of a processor shutdown, the 82370 will activate the CPURST output for 62 CLK2 periods to reset the host processor. This signal is generated after the Shutdown cycle is terminated by the READY# signal.

Although the 82370 Wait State Generator will not automatically respond to a Shutdown (or Halt) cycle, the Wait State Control inputs (WSC0, WSC1) can be used to determine the number of wait states in the same manner as other non-82370 bus cycles.

This Shutdown Detect feature can be enabled or disabled by writing a control bit in the Internal Control Port at address 61H (see Internal Control and Diagnostic Ports). This feature is disabled upon a hardware reset of the 82370. As in the case of Software Reset, the 82370 will reset its Slave-Bus state machine but will not change any of its internal register contents.

10.0 INTERNAL CONTROL AND DIAGNOSTIC PORTS

10.1 Internal Control Port

The format of the Internal Control Port of the 82370 is shown in Figure 10-1. This Control Port is used to enable/disable the Processor Shutdown Detect mechanism as well as controlling the Gate inputs of the Timer 2 and 3. Note that this is a Write-Only port. Therefore, the 82370 will not respond to a read operation to this port. Upon hardware reset, this port will be cleared; i.e., the Shutdown Detect feature and the Gate inputs of Timer 2 and 3 are disabled.

Port Address: 61H (Write only)

10.2 Diagnostic Ports

Two 8-bit read/write Diagnostic Ports are provided in the 82370. These are two storage registers and have no effect on the operation of the 82370. They can be used to store checkpoint data or error codes in the power-on sequence and in the diagnostic service routines. As mentioned in the CPU RESET AND SHUTDOWN DETECT section, these Diagnostic Ports can be used to distinguish between 'cold' and 'warm' reset. Upon hardware reset, both Diagnostic Ports are cleared. The address map of these Diagnostic Ports is shown in Figure 10-2.

Port		Address
Diagnostic Port 1	(Read/Write)	80H
Diagnostic Port 2	(Read/Write)	88H

Figure 10-2. Address Map of Diagnostic Ports

11.0 INTEL RESERVED I/O PORTS

There are nineteen I/O ports in the 82370 address space which are reserved for Intel future peripheral device use only. Their address locations are: 10H, 12H, 14H, 16H, 2AH, 3DH, 3EH, 45H, 46H, 76H, 77H, 7DH, 7EH, CCH, CDH, D0H, D2H, D4H, and D6H. These addresses should not be used in the system since the 82370 will respond to read/write operations to these locations and bus contention may occur if any peripheral is assigned to the same address location.

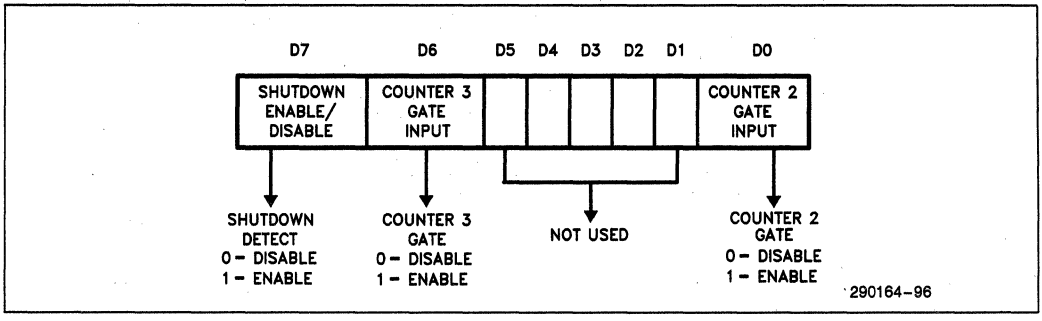


Figure 10-1. Internal Control Port

12.0 PACKAGE THERMAL SPECIFICATIONS

The intel 82370 Integrated System Peripheral is specified for operation when case temperature is within the range of 0°C to 78°C for the ceramic 132-pin PGA package, and 68°C for the 100-pin plastic package. The case temperature may be measured in any environment, to determine whether the 82370 is within specified operating range. The case temperature should be measured at the center of the top surface opposite the pins.

The ambient temperature is guaranteed as long as T_C is not violated. The ambient temperature can be calculated from the θ_{jc} and θ_{ja} from the following equations:

$T_J = T_C + P^*\theta_{jc}$

$T_A = T_j - P^*\theta_{ja}$

$T_C = T_a + P^*[\theta_{ja} - \theta_{jc}]$

Values for θ_{ja} and θ_{jc} are given in Table 12.1 for the 100-lead fine pitch. θ_{ja} is given at various airflows. Table 12.2 shows the maximum T_a allowable (without exceeding T_C) at various airflows. Note that T_a can be improved further by attaching "fins" or a "heat sink" to the package. P is calculated using the maximum ***hot*** I_{CC}.

Table 12.1 82370 Package Thermal Characteristics
Thermal Resistances (°C/Watt) θ_{jc} and θ_{ja}

Package	θ_{jc}	θ_{ja} Versus Airflow-ft³/min (m³/sec)					
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
100L Fine Pitch	7	33	27	24	21	18	17
132L PGA	2	21	17	14	12	11	10

Table 12.2 82370 Maximum Allowable Ambient Temperature at Various Airflows

Package	θ_{jc}	T_a(c) Versus Airflow-ft³/min (m³/sec)					
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
100L Fine Pitch	7	63	74	79	85	91	92
132L PGA	2	74	83	88	93	97	99

100L PQFP Pkg:
$T_c = T_a + P^*(\theta_{ja} - \theta_{jc})$
$T_c = 63 + 220$ mA$(33 - 7)$
$T_c = 63 + 220$ mA(26)
$T_c = 63 + 5.72$
$T_c = 68.7$

132L PGA Pkg:
$T_c = T_a + P^*(\theta_{ja} - \theta_{jc})$
$T_c = 74 + 220$ mA$(21 - 2)$
$T_c = 74 + 220$ mA(19)
$T_c = 74 + 4.18$
$T_c = 78.2$

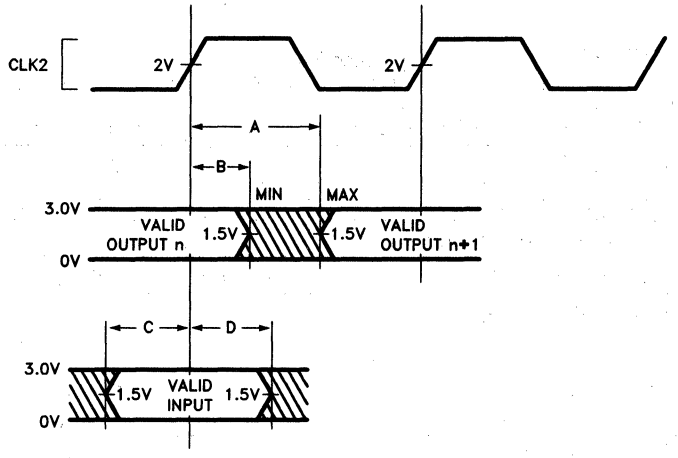
13.0 ELECTRICAL SPECIFICATIONS

82370 D.C. Specifications Functional Operating Range:

V_{CC} = 5.0V ±10%; T_{CASE} = 0°C to 78°C for 132-pin PGA, 0°C to 68°C for 100-pin plastic

Symbol	Parameter Description	Min	Max	Units	Notes
V_{IL}	Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{ILC}	CLK2 Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IHC}	CLK2 Input High Voltage	V_{CC} − 0.8	V_{CC} + 0.3	V	
V_{OL}	Output Low Voltage I_{OL} = 4 mA: A_{1-23}, D_{0-15}, BHE#, BLE# I_{OL} = 5 mA: All Others		 0.45 0.45	 V V	
V_{OH}	Output High Voltage				
I_{OH} = −1 mA	A_{23}–A_1, D_{15}–D_0, BHE#, BLE#	2.4		V	(Note 5)
I_{OH} = −0.2 mA	A_{23}–A_1, D_{15}–D_0, BHE#, BLE#	V_{CC} − 0.5		V	(Note 5)
I_{OH} = −0.9 mA	All Others	2.4		V	(Note 5)
I_{OH} = −0.18 mA	All Others	V_{CC} − 0.5		V	(Note 5)
I_{LI}	Input Leakage Current All Inputs Except: IRQ11#–IRQ23# EOP#, TOUT2/IRQ3# DREQ4/IRQ9#		±15	μA	
I_{LI1}	Input Leakage Current Inputs: IRQ11#–IRQ23# EOP#, TOUT2/IRQ3 DREQ4/IRQ9	10	−300	μA	$0 < V_{IN} < V_{CC}$ (Note 3)
I_{LO}	Output Leakage Current		±15	μA	$0 < V_{IN} < V_{CC}$
I_{CC}	Supply Current (CLK2 = 32 MHz)		220	mA	(Note 4)
C_I	Input Capacitance		12	pF	(Note 2)
C_{CLK}	CLK2 Input Capacitance		20	pF	(Note 2)

NOTES:
1. Minimum value is not 100% tested.
2. f_C = 1 MHz; sampled only.
3. These pins have weak internal pullups. They sould not be left floating.
4. I_{CC} is specified with inputs driven to CMOS levels, and outputs driving CMOS loads. I_{CC} may be higher if inputs are driven to TTL levels, or if outputs are driving TTL loads.
5. Tested at the minimum operating frequency of the part.



LEGEND:
A—Maximum output delay specification
B—Minimum output delay specification
C—Minimum input setup specification
D—Minimum input hold specification

Figure 13-1. Drive Levels and Measurement Points for A.C. Specification

82370 A.C. Specifications These A.C. timings are tested at 1.5V thresholds, except as noted. Functional Operating Range: V_{CC} = 5.0V ±10%; T_{CASE} = 0°C to 78°C for 132-pin PGA, 0°C to 68°C for 100-pin plastic

Symbol	Parameter Description	Min	Max	Units	Notes
	Operating Frequency 1/(t1a × 2)	4	16	MHz	
t1	CLK2 Period	31	125	ns	
t2a	CLK2 High Time	9		ns	At 2.0V
t2b	CLK2 High Time	5		ns	At V_{CC} − 0.8V
t3a	CLK2 Low Time	9		ns	At 2.0V
t3b	CLK2 Low Time	7		ns	At 0.8V
t4	CLK2 Fall Time		7	ns	V_{CC} − 0.8V to 0.8V
t5	CLK2 Rise Time		7	ns	0.8V to V_{CC} − 0.8V
t6	A1–A23, BHE#, BLE# EDACK0–EDACK2 Valid Delay	4	36	ns	C_L = 120 pF
t7	A1–A23, BHE#, BLE# EDACK0–EDACK3 Float Delay	4	40	ns	(Note 1)
t8	A1–A23, BHE#, BLE# Setup Time	6		ns	
t9	A1–A23, BHE#, BLE# Hold Time	4		ns	
t10	W/R#, M/IO#, D/C# Valid Delay	4	33	ns	C_L = 75 pF
t11	W/R#, M/IO#, D/C# Float Delay	4	35	ns	(Note 1)

82370 A.C. Specifications These A.C. timings are tested at 1.5V thresholds, except as noted. Functional Operating Range: V_{CC} = 5.0V ±10%; T_{CASE} = 0°C to 78°C for 132-pin PGA, 0°C to 68°C for 100-pin plastic (Continued)

Symbol	Parameter Description	Min	Max	Units	Notes
t12	W/R#, M/IO#, D/C# Setup Time	6		ns	
t13	W/R#, M/IO#, D/C# Hold Time	4		ns	
t14	ADS# Valid Delay	6	33	ns	CL = 50 pF
t15	ADS# Float Delay	4	35	ns	(Note 1)
t16	ADS# Setup Time	21		ns	
t17	ADS# Hold Time	4		ns	
t18	Slave Mode D0–D15 Read Valid	3	46	ns	C_L = 120 pF
t19	Slave Mode D0–D15 Read Float	6	35	ns	(Note 1)
t20	Slave Mode D0–D15 Write Setup	31		ns	
t21	Slave Mode D0–D15 Write Hold	26		ns	
t22	Master Mode D0–D15 Write Valid	4	40	ns	C_L = 120 pF
t23	Master Mode D0–D15 Write Float	4	35	ns	(Note 1)
t24	Master Mode D0–D15 Read Setup	8		ns	
t25	Master Mode D0–D15 Read Hold	6		ns	
t26	READY# Setup Time	19		ns	
t27	READY# Hold Time	4		ns	
t28	WSC0–WSC1 Setup Time	6		ns	
t29	WSC0–WSC1 Hold Time	21		ns	
t30	RESET Setup Time	13		ns	
t31	RESET Hold Time	4		ns	
t32	READYO# Valid Delay	4	31	ns	C_L = 25 pF
t33	CPURST Valid Delay (Falling Edge Only)	2	18	ns	C_L = 50 pF
t34	HOLD Valid Delay	5	33	ns	C_L = 100 pF
t35	HLDA Setup Time	21		ns	
t36	HLDA Hold Time	6		ns	
t37a	EOP# Setup (Synchronous)	21		ns	
t38a	EOP# Hold (Synchronous)	6		ns	
t37b	EOP# Setup (Asynchronous)	11		ns	
t38b	EOP# Hold (Asynchronous)	11		ns	
t39	EOP# Valid Delay (Falling Edge Only)	5	38	ns	C_L = 100 pF
t40	EOP# Float Delay	5	40	ns	(Note 1)
t41a	DREQ Setup (Synchronous)	21		ns	
t42a	DREQ Hold (Synchronous)	4		ns	
t41b	DREQ Setup (Asynchronous)	11		ns	
t42b	DREQ Hold (Asynchronous)	11		ns	
t43	INT Valid Delay from IRQn		500	ns	
t44	NA# Setup Time	5		ns	
t45	NA# Hold Time	15		ns	

82370 A.C. Specifications These A.C. timings are tested at 1.5V thresholds, except as noted. Functional Operating Range: V_{CC} = 5.0V ±10%; T_{CASE} = 0°C to 78°C for 132-pin PGA, 0°C to 68°C for 100-pin plastic (Continued)

Symbol	Parameter Description	Min	Max	Units	Notes
t46	CLKIN Frequency	DC	10	MHz	
t47	CLKIN High Time	30		ns	2.0V
t48	CLKIN Low Time	50		ns	0.8V
t49	CLKIN Rise Time		10	ns	0.8V to 3.7V
t50	CLKIN Fall Time		10	ns	3.7V to 0.8V
	TOUT1#/REF# Valid Delay				
t51	from CLK2 (Refresh)	4	36	ns	C_L = 120 pF
t52	from CLKIN (Timer)	3	93	ns	C_L = 120 pF
t53	TOUT2# Valid Delay (from CLKIN, Falling Edge Only)	3	93	ns	C_L = 120 pf
t54	TOUT2# Float Delay	3	36	ns	(Note 1)
t55	TOUT3# Valid Delay (from CLKIN)	3	93	ns	C_L = 120 pF
t56	CHPSEL# Valid Delay	1	35	ns	C_L = 25 pF

NOTE:
1. Float condition occurs when the maximum output current becomes less than I_{LO} in magnitude. Float delay is not tested. For testing purposes, the float condition occurs when the dynamic output driven voltage changes with current loads.

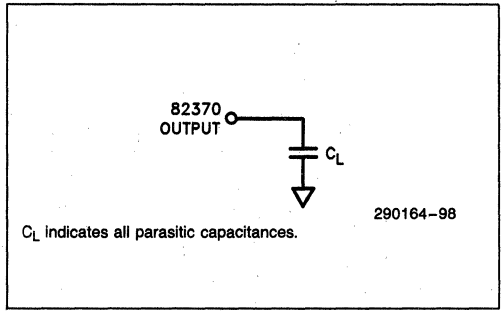


Figure 13-2. A.C. Test Load

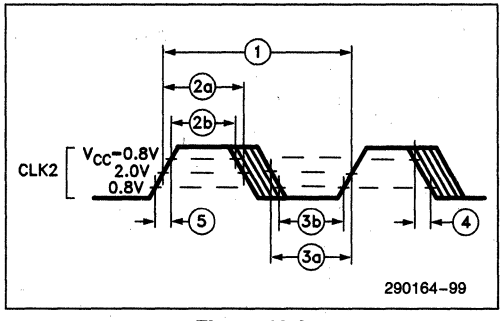


Figure 13-3

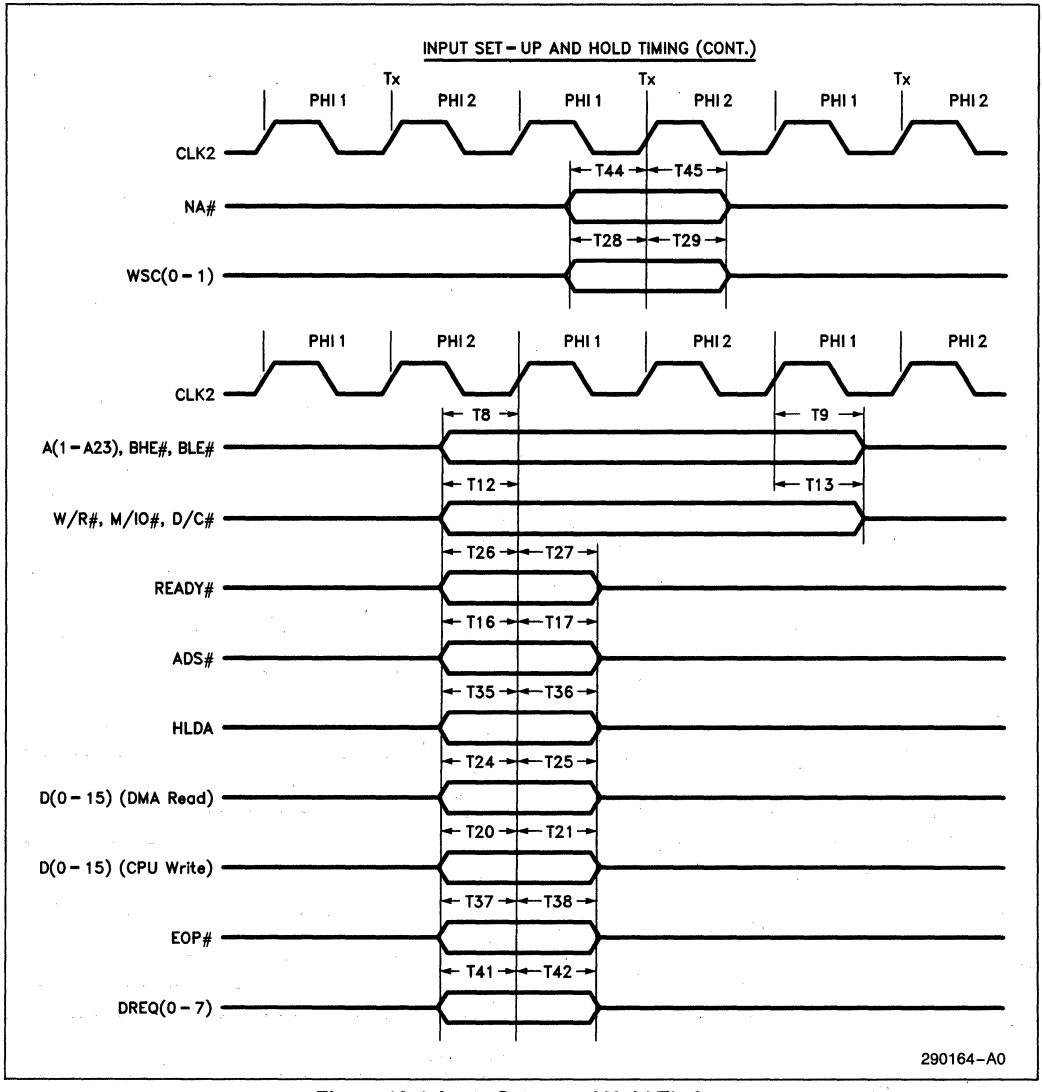


Figure 13-4. Input Setup and Hold Timing

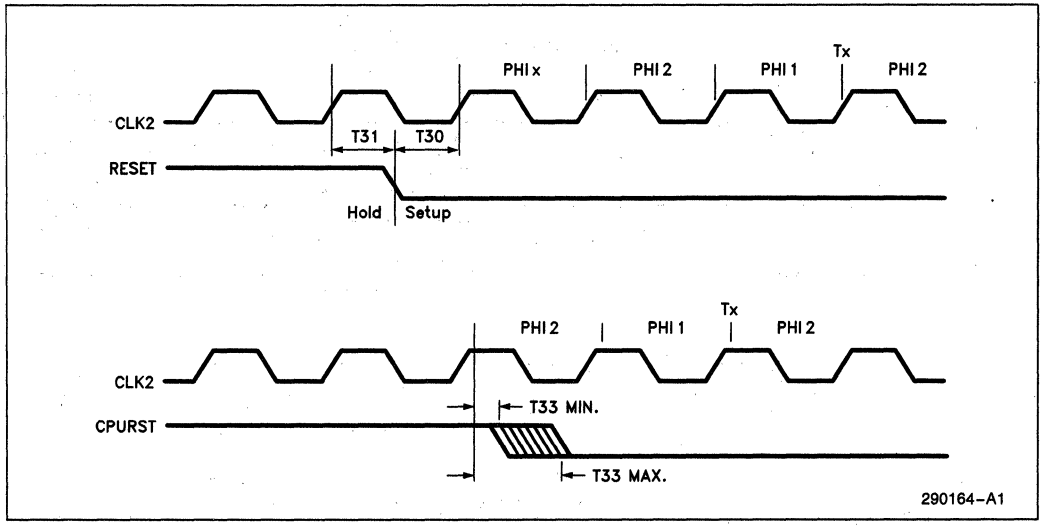


Figure 13-5. Reset Timing

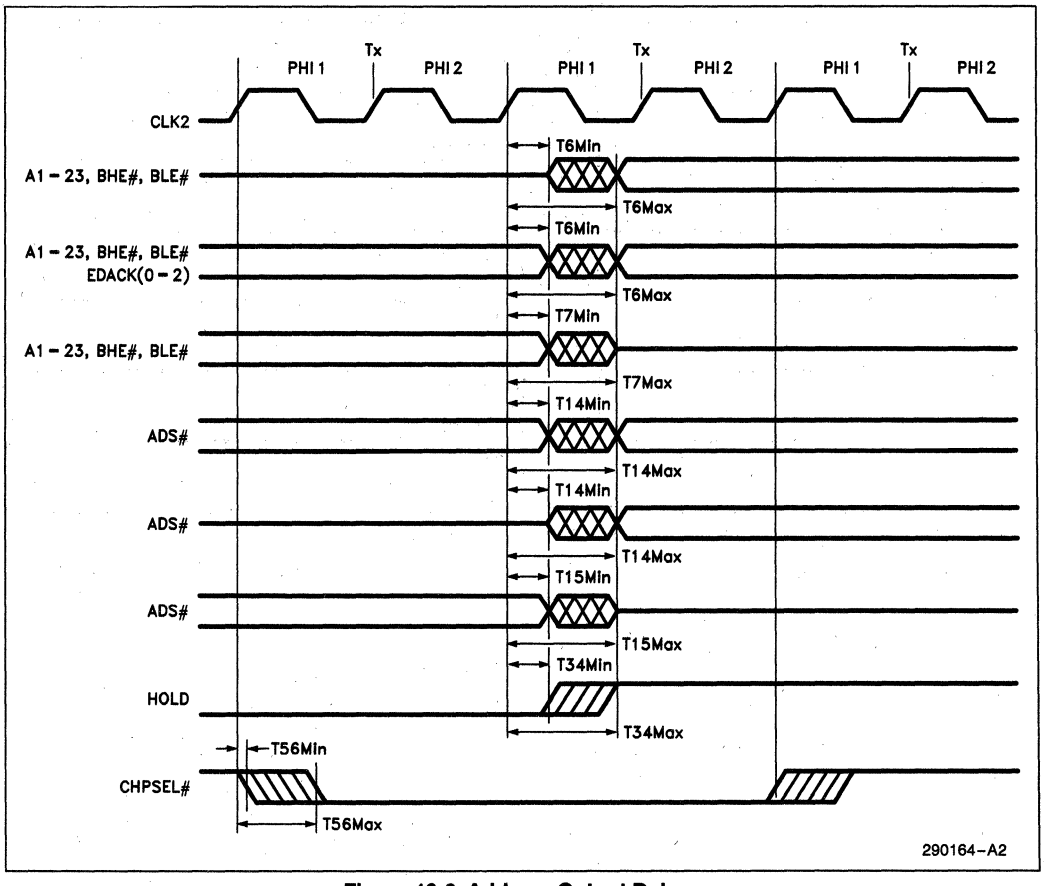


Figure 13-6. Address Output Delays

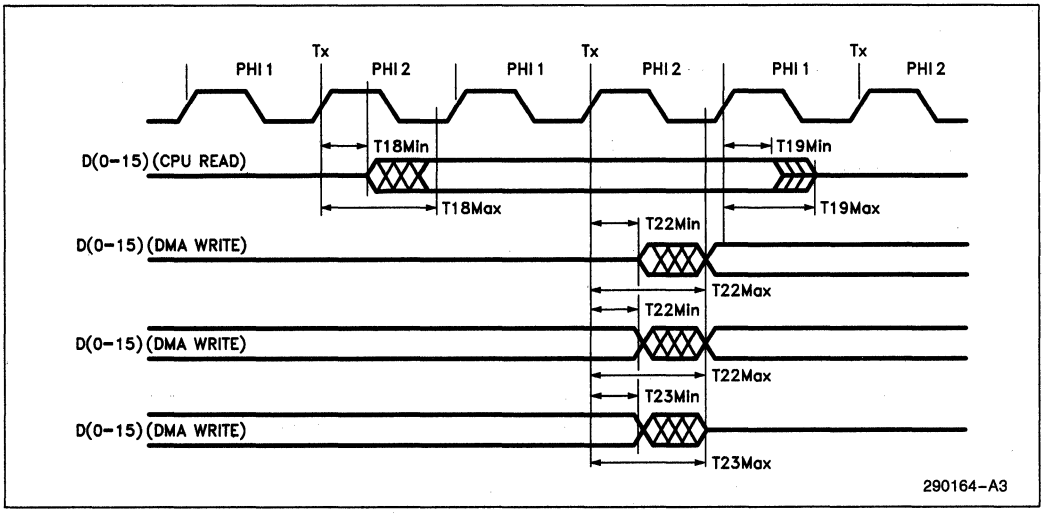


Figure 13-7. Data Bus Output Delays

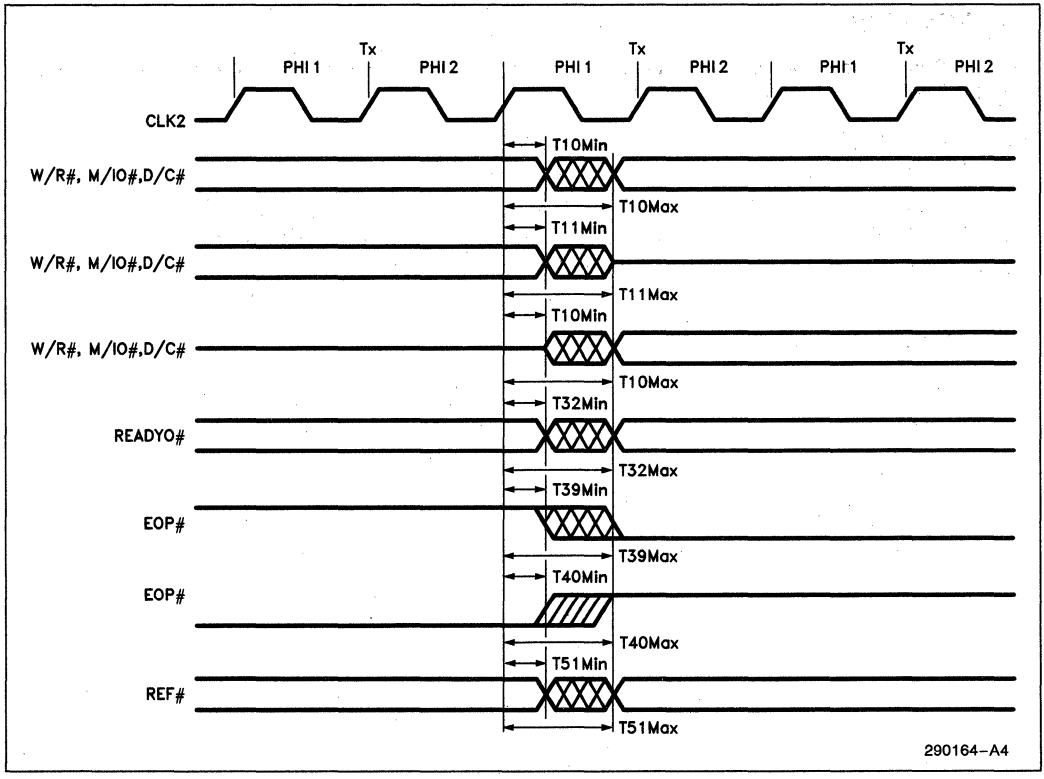


Figure 13-8. Control Output Delays

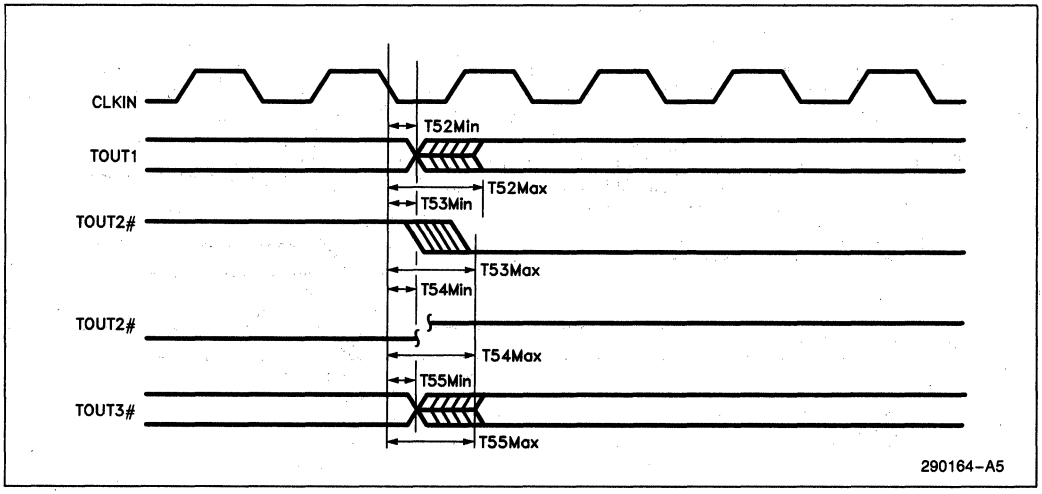


Figure 13-9. Timer Output Delays

14.0 REVISION HISTORY

This 82370 data sheet, version -002, contains updates and improvements to previous versions. A revision summary is listed here for your convenience.

The sections significantly revised since version -001 are:

— Section 12.0 Electrical Characteristics renumbered Section 13.0.
— Section 12.0 Package Thermal Specifications added.
— Section 13.0 Electrical Specifications updated T_{CASE}, V_{OH}, I_{CC}, T_{33}, T_{39}, Figure 13.6.
— Appendix C, Programming the 82370 Interrupt Controllers, added.
— Appendix D, System Notes, added.
— Section 14.0 Revision History added.

APPENDIX A
PORTS LISTED BY ADDRESS

Port Address (HEX)	Description
00	Read/Write DMA Channel 0 Target Address, A0–A15
01	Read/Write DMA Channel 0 Byte Count, B0–B15
02	Read/Write DMA Channel 1 Target Address, A0–A15
03	Read/Write DMA Channel 1 Byte Count, B0–B15
04	Read/Write DMA Channel 2 Target Address, A0–A15
05	Read/Write DMA Channel 2 Byte Count, B0–B15
06	Read/Write DMA Channel 3 Target Address, A0–A15
07	Read/Write DMA Channel 3 Byte Count, B0–B15
08	Read/Write DMA Channel 0–3 Status/Command I Register
09	Read/Write DMA Channel 0–3 Software Request Register
0A	Write DMA Channel 0–3 Set-Reset Mask Register
0B	Write DMA Channel 0–3 Mode Register I
0C	Write Clear Byte-Pointer FF
0D	Write DMA Master-Clear
0E	Write DMA Channel 0–3 Clear Mask Register
0F	Read/Write DMA Channel 0–3 Mask Register
10	Intel Reserved
11	Read/Write DMA Channel 0 Byte Count, B16–B23
12	Intel Reserved
13	Read/Write DMA Channel 1 Byte Count, B16–B23
14	Intel Reserved
15	Read/Write DMA Channel 2 Byte Count, B16–B23
16	Intel Reserved
17	Read/Write DMA Channel 3 Byte Count, B16–B23
18	Write DMA Channel 0–3 Bus Size Register
19	Read/Write DMA Channel 0–3 Chaining Register
1A	Write DMA Channel 0–3 Command Register II
1B	Write DMA Channel 0–3 Mode Register II
1C	Read/Write Refresh Control Register
1E	Reset Software Request Interrupt
20	Write Bank B ICW1, OCW2 or OCW3 Read Bank B Poll, Interrupt Request or In-Service Status Register
21	Write Bank B ICW2, ICW3, ICW4 or OCW1 Read Bank B Interrupt Mask Register
22	Read Bank B ICW2
28	Read/Write IRQ8 Vector Register
29	Read/Write IRQ9 Vector Register
2A	Reserved

Port Address (HEX)	Description
2B	Read/Write IRQ11 Vector Register
2C	Read/Write IRQ12 Vector Register
2D	Read/Write IRQ13 Vector Register
2E	Read/Write IRQ14 Vector Register
2F	Read/Write IRQ15 Vector Register
30	Write Bank A ICW1, OCW2 or OCW3 Read Bank A Poll, Interrupt Request or In-Service Status Register
31	Write Bank A ICW2, ICW3, ICW4 or OCW1 Read Bank A Interrupt Mask Register
32	Read Bank A ICW2
38	Read/Write IRQ0 Vector Register
39	Read/Write IRQ1 Vector Register
3A	Read/Write IRQ1.5 Vector Register
3B	Read/Write IRQ3 Vector Register
3C	Read/Write IRQ4 Vector Register
3D	Reserved
3E	Reserved
3F	Read/Write IRQ7 Vector Register
40	Read/Write Counter 0 Register
41	Read/Write Counter 1 Register
42	Read/Write Counter 2 Register
43	Write Control Word Register I—Counter 0, 1, 2
44	Read/Write Counter 3 Register
45	Reserved
46	Reserved
47	Write Word Register II—Counter 3
61	Write Internal Control Port
64	Write CPU Reset Register (Data—1111XXX0H)
72	Read/Write Wait State Register 0
73	Read/Write Wait State Register 1
74	Read/Write Wait State Register 2
75	Read/Write Refresh Wait State Register
76	Reserved
77	Reserved
7D	Reserved
7E	Reserved
7F	Read/Write Relocation Register
80	Read/Write Internal Diagnostic Port 0
81	Read/Write DMA Channel 2 Target Address, A16–A23
82	Read/Write DMA Channel 3 Target Address, A16–A23
83	Read/Write DMA Channel 1 Target Address, A16–A23
87	Read/Write DMA Channel 0 Target Address, A16–A23
88	Read/Write Internal Diagnostic Port 1
89	Read/Write DMA Channel 6 Target Address, A16–A23
8A	Read/Write DMA Channel 7 Target Address, A16–A23
8B	Read/Write DMA Channel 5 Target Address, A16–A23
8F	Read/Write DMA Channel 4 Target Address, A16–A23

Port Address (HEX)	Description
90	Read/Write DMA Channel 0 Requester Address, A0–A15
91	Read/Write DMA Channel 0 Requester Address, A16–A23
92	Read/Write DMA Channel 1 Requester Address, A0–A15
93	Read/Write DMA Channel 1 Requester Address, A16–A23
94	Read/Write DMA Channel 2 Requester Address, A0–A15
95	Read/Write DMA Channel 2 Requester Address, A16–A23
96	Read/Write DMA Channel 3 Requester Address, A0–A15
97	Read/Write DMA Channel 3 Requester Address, A16–A23
98	Read/Write DMA Channel 4 Requester Address, A0–A15
99	Read/Write DMA Channel 4 Requester Address, A16–A23
9A	Read/Write DMA Channel 5 Requester Address, A0–A15
9B	Read/Write DMA Channel 5 Requester Address, A16–A23
9C	Read/Write DMA Channel 6 Requester Address, A0–A15
9D	Read/Write DMA Channel 6 Requester Address, A16–A23
9E	Read/Write DMA Channel 7 Requester Address, A0–A15
9F	Read/Write DMA Channel 7 Requester Address, A16–A23
A0	Write Bank C ICW1, OCW2 or OCW3 Read Bank C Poll, Interrupt Request or In-Service Status Register
A1	Write Bank C ICW2, ICW3, ICW4 or OCW1 Read Bank C Interrupt Mask Register
A2	Read Bank C ICW2
A8	Read/Write IRQ16 Vector Register
A9	Read/Write IRQ17 Vector Register
AA	Read/Write IRQ18 Vector Register
AB	Read/Write IRQ19 Vector Register
AC	Read/Write IRQ20 Vector Register
AD	Read/Write IRQ21 Vector Register
AE	Read/Write IRQ22 Vector Register
AF	Read/Write IRQ23 Vector Register
C0	Read/Write DMA Channel 4 Target Address, A0–A15
C1	Read/Write DMA Channel 4 Byte Count, B0–B15
C2	Read/Write DMA Channel 5 Target Address, A0–A15
C3	Read/Write DMA Channel 5 Byte Count, B0–B15
C4	Read/Write DMA Channel 6 Target Address, A0–A15
C5	Read/Write DMA Channel 6 Byte Count, B0–B15
C6	Read/Write DMA Channel 7 Target Address, A0–A15
C7	Read/Write DMA Channel 7 Byte Count, B0–B15
C8	Read DMA Channel 4–7 Status/Command I Register
C9	Read/Write DMA Channel 4–7 Software Request Register
CA	Write DMA Channel 4–7 Set-Reset Mask Register
CB	Write DMA Channel 4–7 Mode Register I
CC	Reserved
CD	Reserved
CE	Write DMA Channel 4–7 Clear Mask Register
CF	Read/Write DMA Channel 4–7 Mask Register
D0	Intel Reserved
D1	Read/Write DMA Channel 4 Byte Count, B16–B23
D2	Intel Reserved
D3	Read/Write DMA Channel 5 Byte Count, B16–B23

Port Address (HEX)	Description
D4	Intel Reserved
D5	Read/Write DMA Channel 6 Byte Count, B16–B23
D6	Intel Reserved
D7	Read/Write DMA Channel 7 Byte Count, B16–B23
D8	Write DMA Channel 4–7 Bus Size Register
D9	Read/Write DMA Channel 4–7 Chaining Register
DA	Write DMA Channel 4–7 Command Register II
DB	Write DMA Channel 4–7 Mode Register II

APPENDIX B
PORTS LISTED BY FUNCTION

Port Address (HEX)	Description
DMA CONTROLLER	
0D	Write DMA Master-Clear
0C	Write DMA Clear Byte-Pointer FF
08	Read/Write DMA Channel 0–3 Status/Command I Register
C8	Read/Write DMA Channel 4–7 Status/Command I Register
1A	Write DMA Channel 0–3 Command Register II
DA	Write DMA Channel 4–7 Command Register II
0B	Write DMA Channel 0–3 Mode Register I
CB	Write DMA Channel 4–7 Mode Register I
1B	Write DMA Channel 0–3 Mode Register II
DB	Write DMA Channel 4–7 Mode Register II
09	Read/Write DMA Channel 0–3 Software Request Register
C9	Read/Write DMA Channel 4–7 Software Request Register
1E	Reset Software Request Interrupt
0E	Write DMA Channel 0–3 Clear Mask Register
CE	Write DMA Channel 4–7 Clear Mask Register
0F	Read/Write DMA Channel 0–3 Mask Register
CF	Read/Write DMA Channel 4–7 Mask Register
0A	Write DMA Channel 0–3 Set-Reset Mask Register
CA	Write DMA Channel 4–7 Set-Reset Mask Register
18	Write DMA Channel 0–3 Bus Size Register
D8	Write DMA Channel 4–7 Bus Size Register
19	Read/Write DMA Channel 0–3 Chaining Register
D9	Read/Write DMA Channel 4–7 Chaining Register
00	Read/Write DMA Channel 0 Target Address, A0–A15
87	Read/Write DMA Channel 0 Target Address, A16–A23
01	Read/Write DMA Channel 0 Byte Count, B0–B15
11	Read/Write DMA Channel 0 Byte Count, B16–B23
90	Read/Write DMA Channel 0 Requester Address, A0–A15
91	Read/Write DMA Channel 0 Requester Address, A16–A23

Port Address (HEX)	Description
DMA CONTROLLER (Continued)	
02	Read/Write DMA Channel 1 Target Address, A0–A15
83	Read/Write DMA Channel 1 Target Address, A16–A23
03	Read/Write DMA Channel 1 Byte Count, B0–B15
13	Read/Write DMA Channel 1 Byte Count, B16–B23
92	Read/Write DMA Channel 1 Requester Address, A0–A15
93	Read/Write DMA Channel 1 Requester Address, A16–A23
04	Read/Write DMA Channel 2 Target Address, A0–A15
81	Read/Write DMA Channel 2 Target Address, A16–A23
05	Read/Write DMA Channel 2 Byte Count, B0–B15
15	Read/Write DMA Channel 2 Byte Count, B16–B23
94	Read/Write DMA Channel 2 Requester Address, A0–A15
95	Read/Write DMA Channel 2 Requester Address, A16–A23
06	Read/Write DMA Channel 3 Target Address, A0–A15
82	Read/Write DMA Channel 3 Target Address, A16–A23
07	Read/Write DMA Channel 3 Byte Count, B0–B15
17	Read/Write DMA Channel 3 Byte Count, B16–B23
96	Read/Write DMA Channel 3 Requester Address, A0–A15
97	Read/Write DMA Channel 3 Requester Address, A16–A23
C0	Read/Write DMA Channel 4 Target Address, A0–A15
8F	Read/Write DMA Channel 4 Target Address, A16–A23
C1	Read/Write DMA Channel 4 Byte Count, B0–B15
D1	Read/Write DMA Channel 4 Byte Count, B16–B23
98	Read/Write DMA Channel 4 Requester Address, A0–A15
99	Read/Write DMA Channel 4 Requester Address, A16–A23
C2	Read/Write DMA Channel 5 Target Address, A0–A15
8B	Read/Write DMA Channel 5 Target Address, A16–A23
C3	Read/Write DMA Channel 5 Byte Count, B0–B15
D3	Read/Write DMA Channel 5 Byte Count, B16–B23
9A	Read/Write DMA Channel 5 Requester Address, A0–A15
9B	Read/Write DMA Channel 5 Requester Address, A16–A23
C4	Read/Write DMA Channel 6 Target Address, A0–A15
89	Read/Write DMA Channel 6 Target Address, A16–A23
C5	Read/Write DMA Channel 6 Byte Count, B0–B15
D5	Read/Write DMA Channel 6 Byte Count, B16–B23
9C	Read/Write DMA Channel 6 Requester Address, A0–A15
9D	Read/Write DMA Channel 6 Requester Address, A16–A23
C6	Read/Write DMA Channel 7 Target Address, A0–A15
8A	Read/Write DMA Channel 7 Target Address, A16–A23
C7	Read/Write DMA Channel 7 Byte Count, B0–B15
D7	Read/Write DMA Channel 7 Byte Count, B16–B23
9E	Read/Write DMA Channel 7 Requester Address, A0–A15
9F	Read/Write DMA Channel 7 Requester Address, A16–A23

Port Address (HEX)	Description
INTERRUPT CONTROLLER	
20	Write Bank B ICW1, OCW2 or OCW3 Read Bank B Poll, Interrupt Request or In-Service Status Register
21	Write Bank B ICW2, ICW3, ICW4 or OCW1 Read Bank B Interrupt Mask Register
22	Read Bank B ICW2
28	Read/Write IRQ8 Vector Register
29	Read/Write IRQ9 Vector Register
2A	Reserved
2B	Read/Write IRQ11 Vector Register
2C	Read/Write IRQ12 Vector Register
2D	Read/Write IRQ13 Vector Register
2E	Read/Write IRQ14 Vector Register
2F	Read/Write IRQ15 Vector Register
A0	Write Bank C ICW1, OCW2 or OCW3 Read Bank C Poll, Interrupt Request or In-Service Status Register
A1	Write Bank C ICW2, ICW3, ICW4 or OCW1 Read Bank C Interrupt Mask Register
A2	Read Bank C ICW2
A8	Read/Write IRQ16 Vector Register
A9	Read/Write IRQ17 Vector Register
AA	Read/Write IRQ18 Vector Register
AB	Read/Write IRQ19 Vector Register
AC	Read/Write IRQ20 Vector Register
AD	Read/Write IRQ21 Vector Register
AE	Read/Write IRQ22 Vector Register
AF	Read/Write IRQ23 Vector Register
30	Write Bank A ICW1, OCW2 or OCW3 Read Bank A Poll, Interrupt Request or In-Service Status Register
31	Write Bank A ICW2, ICW3, ICW4 or OCW1 Read Bank A Interrupt Mask Register
32	Read Bank A ICW2
38	Read/Write IRQ0 Vector Register
39	Read/Write IRQ1 Vector Register
3A	Read/Write IRQ1.5 Vector Register
3B	Read/Write IRQ3 Vector Register
3C	Read/Write IRQ4 Vector Register
3D	Reserved
3E	Reserved
3F	Read/Write IRQ7 Vector Register

Port Address (HEX)	Description
PROGRAMMABLE INTERVAL TIMER	
40	Read/Write Counter 0 Register
41	Read/Write Counter 1 Register
42	Read/Write Counter 2 Register
43	Write Control Word Register I—Counter 0, 1, 2
44	Read/Write Counter 3 Register
47	Write Word Register II—Counter 3
CPU RESET	
64	Write CPU Reset Register (Data—1111XXX0H)
WAIT STATE GENERATOR	
72	Read/Write Wait State Register 0
73	Read/Write Wait State Register 1
74	Read/Write Wait State Register 2
75	Read/Write Refresh Wait State Register
DRAM REFRESH CONTROLLER	
1C	Read/Write Refresh Control Register
INTERNAL CONTROL AND DIAGNOSTIC PORTS	
61	Write Internal Control Port
80	Read/Write Internal Diagnostic Port 0
88	Read/Write Internal Diagnostic Port 1
RELOCATION REGISTER	
7F	Read/Write Relocation Register
INTEL RESERVED PORTS	
10	Reserved
12	Reserved
14	Reserved
16	Reserved
2A	Reserved
3D	Reserved
3E	Reserved
45	Reserved
46	Reserved
76	Reserved
77	Reserved
7D	Reserved
7E	Reserved
CC	Reserved
CD	Reserved
D0	Reserved
D2	Reserved
D4	Reserved
D6	Reserved

APPENDIX C
PROGRAMMING THE 82370 INTERRUPT CONTROLLERS

This Appendix describes two methods of programming and initializing the Interrupt Controllers of the 82370. A simple interrupt service routine is also shown which provides compatibility with the 82C59 Interrupt Controller.

The two methods of programming the 8237° Interrupt Controllers are needed to provide simple initialization procedures in different software environments. For new applications, a simple initialization and programming sequence can be used. For PC-DOS or other applications which expect 8259s, an interrupt handler for initialization traps must be provided. Once the handler is in place, all three 82370 Interrupt Controller banks can be programmed or initialized in the same manner as an 8259.

The ICW2 interrupt is generated by the 82370 when writing the ICW2 command to any of the interrupt controller banks. This interrupt is supplied to provide compatibility to existing code that expects to be programming 82C59s. The ICW2 value is stored in the ICW2 register of the associated bank, but is ignored by the controller. It is the responsibility of the ICW2 interrupt handler to read the ICW2 register and use its value to program the individual vector registers accordingly.

NEW APPLICATIONS

New applications do not generally require compatibility with previous code, or at least the code is usually easily modifiable. If the application fits this description, then the ICW2 interrupt can be ignored. This is done by initializing the interrupt controller as necessary, and before enabling CPU interrupts, removing the ICW2 interrupt request by reading the ICW2 register. Listing 1 shows the code for doing this for bank A. The same procedure can be used for the other banks.

Listing 1.
Initialization of an 82370 Interrupt Controller Bank Without ICW2 Interrupts

```
INIT_BANK_A   proc near

     cli                    ;disable all interrupts

;initialize controller logic
     mov al,ICW1            ;begin sequence
     out 30h,al
     mov al,ICW2            ;send dummy ICW2
     out 31h,al
     mov al,ICW3            ;send ICW3 if necessary
     out 31h,al
     mov al,ICW4            ;send ICW4
     out 31h,al

     mov al,BANK_A_MASK     ;write to mask register (OCW1)
     out 31h,al

;program vector registers

     mov al,ICW2            ;IRQ0
     out 38h,al
     mov al,ICW2+1          ;IRQ1
     out 39h,al
     mov al,ICW2_VECTOR     ;IRQ1.5 (probably never used in
     out 3Ah,al                          ; this system)
     mov al,ICW2+3          ;IRQ3
     out 3Bh,al
     mov al,ICW2+4          ;IRQ4
     out 3Ch,al
     mov al,ICW2+7          ;IRQ7
     out 3Fh,al

;remove ICW2 interrupt request

     in  al,31h             ;read mask register to work around
                            ; A-step errata

     in  al,32h             ;read ICW2 register to clear
                            ; interrupt request

;return to calling program

     sti                    ;re-enable interrupts
     ret

INIT_BANK_A  endp
```

OLD APPLICATIONS

In applications where 8259 compatibility is required, the ICW2 interrupt handler must be invoked whenever an interrupt controller is initialized (ICW1–ICW2–ICWn sequence). The handler's purpose is to read the ICW2 value from the ICW2 read register and write the appropriate sequence of vectors to the vector registers. Listing 2 shows the typical initialization sequence (this is not changed from the 8259), and the required initialization for operation of the ICW2 interrupt handler. Listing 2 shows the ICW2 interrupt handler.

Listing 2.
Initialization of Bank A for ICW2 Interrupts

```
    cli                     ;disable all interrupts

;initialize controller logic

    mov  al,ICW1            ;begin sequence
    out  30h,al
    mov  al,ICW2            ;send dummy ICW2
    out  31h,al

;*******
    mov  al,ICW3            ;send ICW3 if necessary
    out  31h,al             ; note that using ICW3 for
                            ; cascading bank B is not required
                            ; and will affect the way EOIs are
                            ; required for nesting. It is
                            ; advised that ICW3 not be used.
;*******

    mov  al,ICW4            ;send ICW4
    out  ;31h,al

    mov  al,Bank_A_Mask     ;write to mask register (OCW1=7Bh)
    out  31h,al             ;don't mask off IRQ1.5 or Default
                            ; interrupt (IRQ7)

;program necessary vector registers

    mov  al,ICW2_VECTOR     ;IRQ1.5
    out  3Ah,al

    mov  al,IRQ7_DEFAULT_VECTOR
    out  3Fh,al

;remove ICW2 interrupt request for bank A

    in   al,31h             ;read mask register to work around
                            ; A-step errata

    in   al.32h             ;read ICW2 register to clear
                            ; interrupt request

;at this point install interrupt call vector for ICW2, if
;not already done somewhere else in the code

    sti                     ;re-enable interrupts
```

Listing 3.
ICW2 Interrupt Service Routine

```
ICW2_INT_HANDLER        proc near

    push ax             ;save registers
    push cx
    push dx

; service bank B

    in   al,21h         ;read mask register for A-step errata

    in   al,22h         ;read ICW2
    mov  cx,8           ;count vectors
    mov  dx,28h         ;point to vectors

BANK_B_LOOP;

    out  dx,al          ;write vector
    inc  al             ;next vector
    inc  dx             ;next vector I/O address
    loop BANK_B_LOOP

;service bank C

    in   al,0A1h        ;read mask register for A-step errata

    in   al,0A2h        ;read ICW2
    mov  cx,8           ;count vectors
    mov  dx,0A8h        ;point to vectors

BANK_C_LOOP:

    out  dx,al          ;write vector
    inc  al             ;next vector
    inc  dx             ;next vector i/o address
    loop BANK_C_LOOP

    pop  dx             ;restore registers
    pop  cx
    pop  ax
    iret                ;return

ICW2_INT_HANDLER        endp
```

Table 1. Interrupt Controller Registers

Bank A:			
	30H	write	ICW1, OCW2, OCW3
		read	Poll, IRR, ISR
	31H	write	ICW2, ICW3, ICW4, OCW1
		read	IMR
	32H	read	ICW2 read register
	38H	read/write	IRQ0 vector
	39H	read/write	IRQ1 vector
	3AH	read/write	IRQ1.5 vector
	3BH	read/write	IRQ3 vector
	3CH	read/write	IRQ4 vector
	3DH	RESERVED	
	3EH	RESERVED	
	3FH	read/write	IRQ7 vector
Bank B:			
	20H	write	ICW1, OCW2, OCW3
		read	Poll, IRR, ISR
	21H	write	ICW2, ICW3, ICW4, OCW1
		read	IMR
	22H	read	ICW2 read register
	28H	read/write	IRQ8 vector
	29H	read/write	IRQ9 vector
	2AH	RESERVED	
	2BH	read/write	IRQ11 vector
	2CH	read/write	IRQ12 vector
	2DH	read/write	IRQ13 vector
	2EH	read/write	IRQ14 vector
	2FH	read/write	IRQ15 vector
Bank C:			
	A0H	write	ICW1, OCW2, OCW3
		read	Poll, IRR, ISR
	A1H	write	ICW2, ICW3, ICW4, OCW1
		read	IMR
	A2H	read	ICW2 read register
	A8H	read/write	IRQ16 vector
	A9H	read/write	IRQ17 vector
	AAH	read/write	IRQ18 vector
	ABH	read/write	IRQ19 vector
	ACH	read/write	IRQ20 vector
	ADH	read/write	IRQ21 vector
	AEH	read/write	IRQ22 vector
	AFH	read/write	IRQ23 vector

APPENDIX D
SYSTEM NOTES

1. BHE# IN MASTER MODE.

In Master Mode, BHE# will be activated during DMA to/from 8-bit devices residing at even locations when the remaining byte count is greater than 1.

For example, if an 8-bit device is located at 00000000 Hex and the number of bytes to be transferred is > 1, the first address/BHE# combination will be 00000000/0. In some systems this will cause the bus controller to perform two 8-bit accesses, the first to 0000000 Hex and the second to 00000001 Hex. However, the 82370's DMA will only read/write one byte. This may or may not cause a problem in the system depending on what is located at 00000001 Hex.

Solution:

There are two solutions if BH# active is unacceptable. Of the two, number 2 is the cleanest and most recommended.

1. If there is an 8-bit device that uses DMA located at an even address, do not use that address + 1. The limitation of this solution is that the user must have complete control over what addresses will be used in the end system.

2. Do not allow the Bus Controller to split cycles for the DMA.

2. RESET OUTPUT OF 82370:

The 80376 requires its RESET line to be active for 80 clock cycles. The 82370 generates holds the RESET line active for 62 clock cycles.

The following design example shows how the user can extend the active high of the RESET line to 80 clock cycles.

Extending the RESET Output of the 82370

This section describes a hardware solution for using the 82370's CPURST output and the software reset command to cause the 80376 to enter into a self-test.

The 80376 requires two simultaneous events in order to initiate the self-test sequence. The RESET input of the processor must be held active for at least 80 CLK2 periods and the BUSY# input must be low 8 CLK2 periods prior to and 8 CLK2 periods subsequent to RESET going inactive.

A system which does not have an 80387SX will simply have the BUSY# input to the 80376 tied low. A system which contains the 80387SX will require extra logic between the BUSY# output of the 80387SX and the BUSY# input of the 80376 in order to force self-test on reset. The extra BUSY# logic required will not be described here.

The 82370 CPURST output is intended to be retimed with faster TTL components in order to meet the RESET input setup time requirements of the 80376 and 80387SX. This requires a 74F379 (quad flip-flop with enable) or equivalent. The flip-flops required are described in TECHBIT (Ed Grochowski, April 10, 1987).

The 82370 does not meet the RESET pulse duration requirements for causing self-test of the 80376 when a software reset command is issued to the 82370. The 82370 provides a RESET pulse width of 62 CLK2 periods, the 80376 requires 80 CLK2 periods as mentioned earlier.

In order to cause the 80376 to do a self-test after a software reset, the CPURST output pulse of the 82370 must be lengthened. Figure 1 shows a circuit which will do this.

intel

UNITED STATES
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

JAPAN
Intel Japan K.K.
5-6 Tokodai, Tsukuba-shi
Ibaraki, 300-26

FRANCE
Intel Corporation S.A.R.L.
1, Rue Edison, BP 303
78054 Saint-Quentin-en-Yvelines Cedex

UNITED KINGDOM
Intel Corporation (U.K.) Ltd.
Pipers Way
Swindon
Wiltshire, England SN3 1RJ

WEST GERMANY
Intel Semiconductor GmbH
Dornacher Strasse 1
8016 Feldkirchen bei Muenchen

HONG KONG
Intel Semiconductor Ltd.
10/F East Tower
Bond Center
Queensway, Central

CANADA
Intel Semiconductor of Canada, Ltd.
190 Attwell Drive, Suite 500
Rexdale, Ontario M9W 6H8

Printed in U.S.A./0189/75K/RRD RJ
Microprocessor/Peripheral

ISBN 1-55512-041-5