intel

Microprocessor and Peripheral Handbook

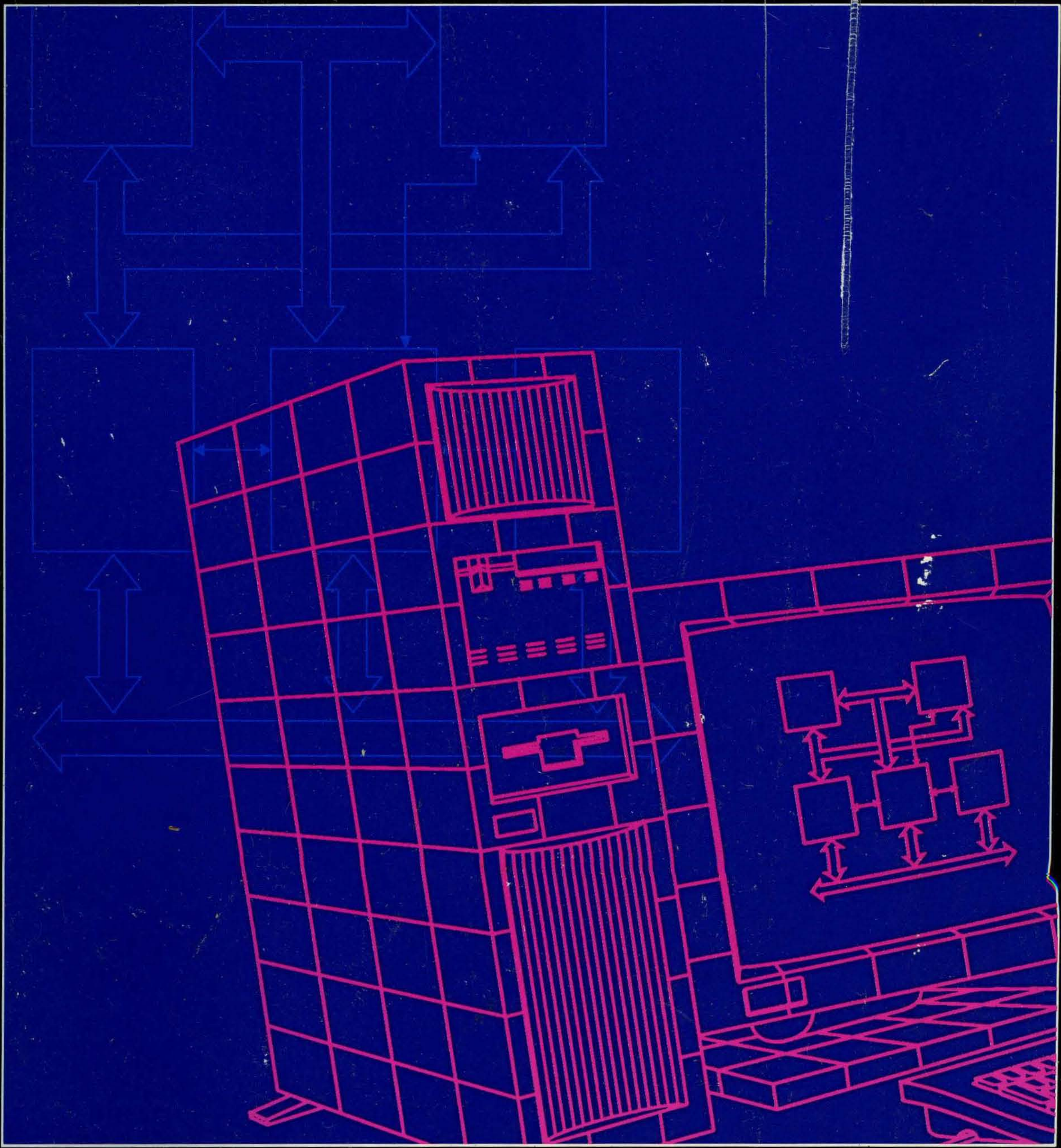
Volume II—Peripherals

1989

intel

Microprocessor and Peripheral Handbook

Volume II—Peripheral



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

U.S. and CANADA LITERATURE ORDER FORM

NAME: ____________________

COMPANY: ____________________

ADDRESS: ____________________

CITY: ____________________ STATE: __________ ZIP: __________

COUNTRY: ____________________

PHONE NO.: (   ) ____________________

ORDER NO.	TITLE	QTY.		PRICE		TOTAL
			×		=	
			×		=	
			×		=	
			×		=	
			×		=	
			×		=	
			×		=	
			×		=	
			×		=	
			×		=	

Subtotal __________

Must Add Your Local Sales Tax __________

Postage: add 10% of subtotal → Postage __________

Total __________

Pay by check, money order, or include company purchase order with this form ($100 minimum).We also accept VISA, MasterCard or American Express. Make payment to Intel Literature Sales. Allow 2-4 weeks for delivery.

□ VISA □ MasterCard □ American Express Expiration Date ____________________

Account No. ____________________

Signature ____________________

Mail To: Intel Literature Sales
P.O. Box 58130
Santa Clara, CA 95052-8130

International Customers outside the U.S. and Canada should use the International order form or contact their local Sales Office or Distributor.

For phone orders in the U.S. and Canada
Call Toll Free: (800) 548-4725

Prices good until 12/31/89.

Source HB

intel

INTERNATIONAL LITERATURE ORDER FORM

NAME: ______________________________

COMPANY: ______________________________

ADDRESS: ______________________________

CITY: ____________________ STATE: __________ ZIP: __________

COUNTRY: ______________________________

PHONE NO.: (______) ______________________________

ORDER NO.	TITLE	QTY.		PRICE		TOTAL
□□□□□□	____________	_____	×	_____	=	_____
□□□□□□	____________	_____	×	_____	=	_____
□□□□□□	____________	_____	×	_____	=	_____
□□□□□□	____________	_____	×	_____	=	_____
□□□□□□	____________	_____	×	_____	=	_____
□□□□□□	____________	_____	×	_____	=	_____
□□□□□□	____________	_____	×	_____	=	_____
□□□□□□	____________	_____	×	_____	=	_____
□□□□□□	____________	_____	×	_____	=	_____
□□□□□□	____________	_____	×	_____	=	_____

Subtotal __________

Must Add Your
Local Sales Tax __________

Total __________

PAYMENT

Cheques should be made payable to your local Intel Sales Office (see inside back cover.)

Other forms of payment may be available in your country. Please contact the Literature Coordinator at your local Intel Sales Office for details.

The completed form should be marked to the attention of the LITERATURE COORDINATOR and returned to your local Intel Sales Office.

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 II PERIPHERAL

1989

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, î, ICE, iCEL, iCS, iDBP, iDIS, I^{2}ICE, 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	
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	
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	

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

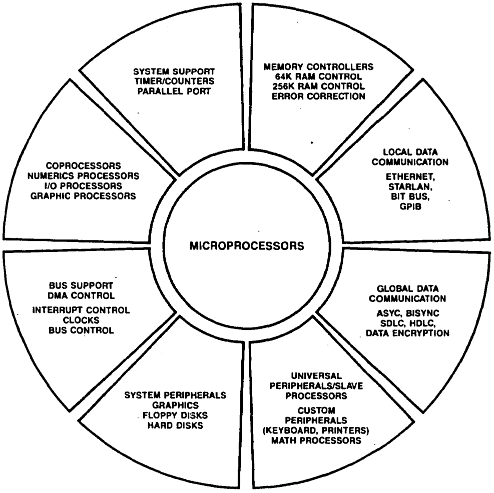
Intel microprocessors and peripherals provide the highest system performance for the demands of today's (and tomorrow's) microprocessor-based applications. For example, the 80386 32 bit offers the highest performance for multitasking, multiuser systems. Intel's peripheral products have been designed 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.

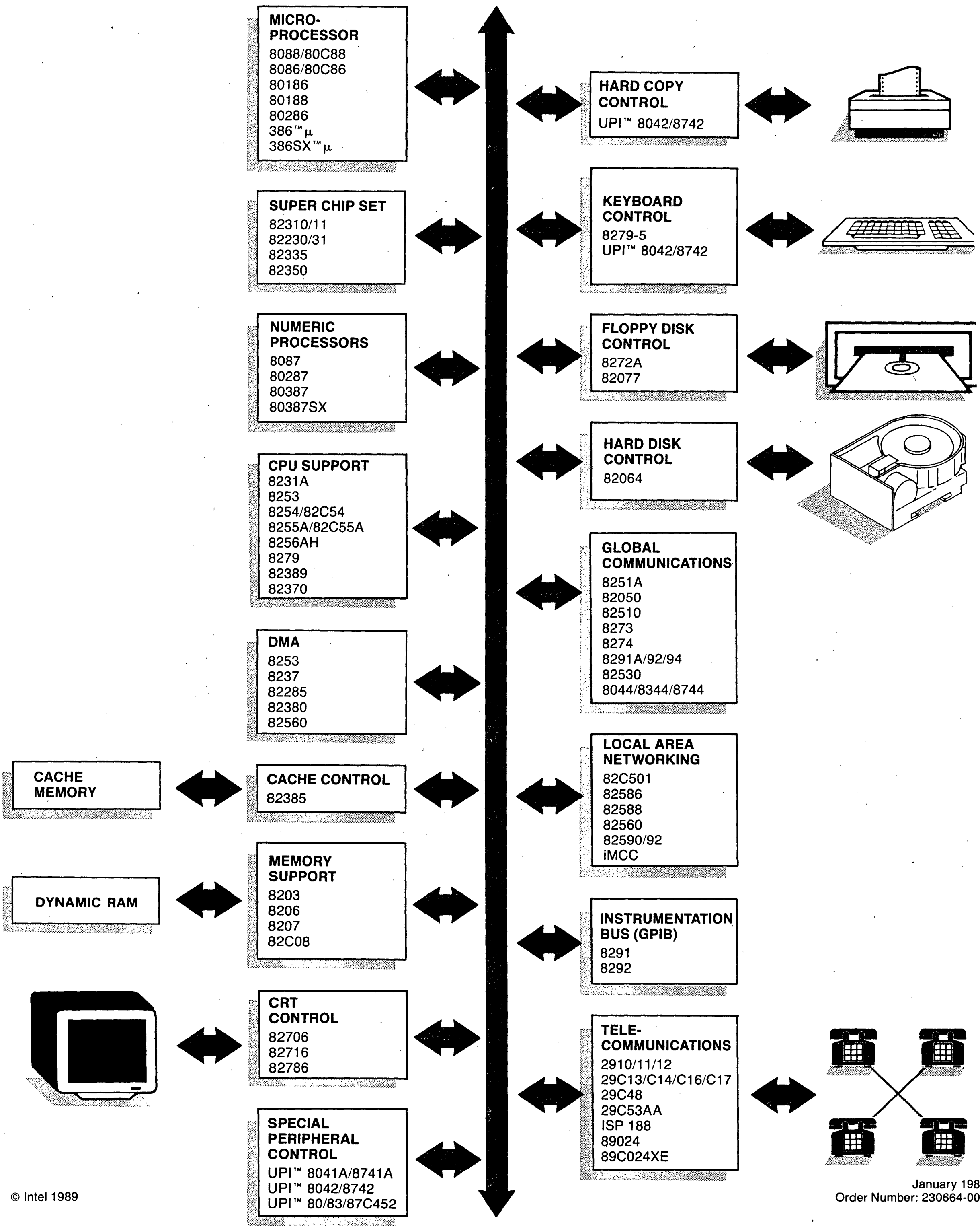
An "X" in a horizontal application row indicates a potential peripheral or CPU, depending upon the features desired. For example, a conversational terminal could use either of the three display controllers, depending upon features like the number of characters per row or font capability. A "Y" indicates 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

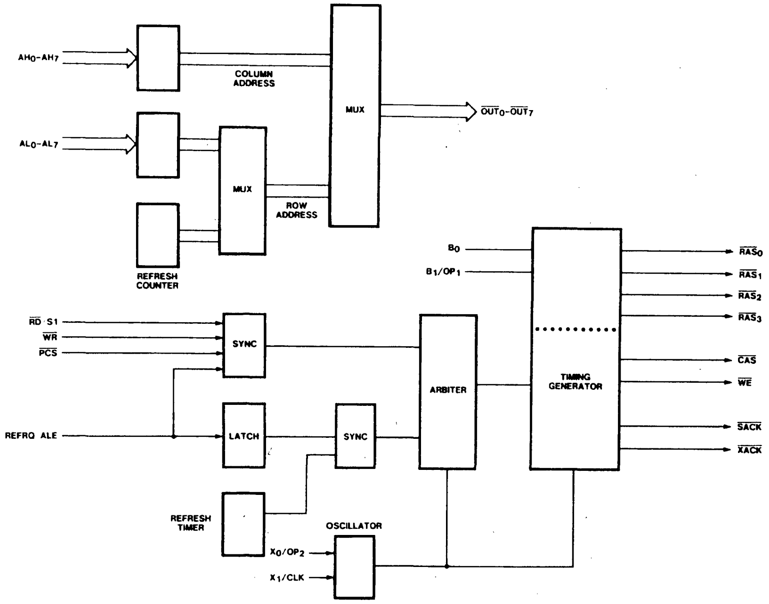
January 198
Order Number: 230664-00

Memory Controllers 5

8203
64K DYNAMIC RAM CONTROLLER

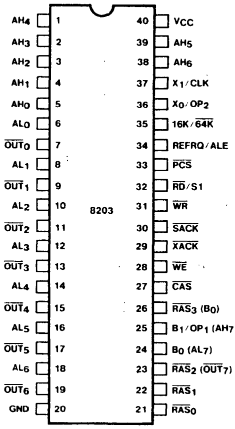
- **Provides All Signals Necessary to Control 64K and 16K Dynamic Memories**
- **Directly Addresses and Drives Up to 64 Devices Without External Drivers**
- **Provides Address Multiplexing and Strobes**
- **Provides a Refresh Timer and a Refresh Counter**
- **Provides Refresh/Access Arbitration**
- **Internal Clock Capability with the 8203-1 and the 8203-3**
- **Fully Compatible with Intel® 8080A, 8085A, iAPX88, and iAPX 86 Family Microprocessors**
- **Decodes CPU Status for Advanced Read Capability in 16K Mode with the 8203-1 and the 8203-3.**
- **Provides System Acknowledge and Transfer Acknowledge Signals**
- **Refresh Cycles May be Internally or Externally Requested (For Transparent Refresh)**
- **Internal Series Damping Resistors on All RAM Outputs**

The Intel® 8203 is a Dynamic RAM System Controller designed to provide all signals necessary to use 64K or 16K Dynamic RAMs in microcomputer systems. The 8203 provides multiplexed addresses and address strobes, refresh logic, refresh/access arbitration. Refresh cycles can be started internally or externally. The 8203-1 and the 8203-3 support an internal crystal oscillator and Advanced Read Capability. The 8203-3 is a ±5% V_{CC} part.



210444-1

Figure 1. 8203 Block Diagram



210444-2

Figure 2.
Pin Configuration

Table 1. Pin Descriptions

Symbol	Pin No.	Type	Name and Function
AL_0 AL_1 AL_2 AL_3 AL_4 AL_5 AL_6	6 8 10 12 14 16 18	I I I I I I I	**ADDRESS LOW:** CPU address inputs used to generate memory row address.
AH_0 AH_1 AH_2 AH_3 AH_4 AH_5 AH_6	5 4 3 2 1 39 38	I I I I I I I	**ADDRESS HIGH:** CPU address inputs used to generate memory column address.
B_0/AL_7 $B_1/OP_1/$ AH_7	24 25	I I	**BANK SELECT INPUTS:** Used to gate the appropriate $\overline{RAS}$ output for a memory cycle. B_1/OP_1 option used to select the Advanced Read Mode. (Not available in 64K mode.) See Figure 5. When in 64K RAM Mode, pins 24 and 25 operate as the AL_7 and AH_7 address inputs.
$\overline{PCS}$	33	I	**PROTECTED CHIP SELECT:** Used to enable the memory read and write inputs. Once a cycle is started, it will not abort even if $\overline{PCS}$ goes inactive before cycle completion.
$\overline{WR}$	31	I	**MEMORY WRITE REQUEST.**
$\overline{RD}/S1$	32	I	**MEMORY READ REQUEST:** S1 function used in Advanced Read mode selected by OP_1 (pin 25).
REFRQ/ ALE	34	I	**EXTERNAL REFRESH REQUEST:** ALE function used in Advanced Read mode, selected by OP_1 (pin 25).
$\overline{OUT}_0$ $\overline{OUT}_1$ $\overline{OUT}_2$ $\overline{OUT}_3$ $\overline{OUT}_4$ $\overline{OUT}_5$ $\overline{OUT}_6$	7 9 11 13 15 17 19	O O O O O O O	**OUTPUT OF THE MULTIPLEXER:** These outputs are designed to drive the addresses of the Dynamic RAM array. (Note that the $\overline{OUT}_{0-7}$ pins do not require inverters or drivers for proper operation.)
$\overline{WE}$	28	O	**WRITE ENABLE:** Drives the Write Enable inputs of the Dynamic RAM array.
$\overline{CAS}$	27	O	**COLUMN ADDRESS STROBE:** This output is used to latch the Column Address into the Dynamic RAM array.
$\overline{RAS}_0$ $\overline{RAS}_1$ $\overline{RAS}_2/$ $\overline{OUT}_7$ $\overline{RAS}_3/B_0$	21 22 23 26	O O O I/O	**ROW ADDRESS STROBE:** Used to latch the Row Address into the bank of dynamic RAMs, selected by the 8203 Bank Select pins (B_0, B_1/OP_1). In 64K mode, only $\overline{RAS}_0$ and $\overline{RAS}_1$ are available; pin 23 operates as $\overline{OUT}_7$ and pin 26 operates as the B_0 bank select input.
$\overline{XACK}$	29	O	**TRANSFER ACKNOWLEDGE:** This output is a strobe indicating valid data during a read cycle or data written during a write cycle. $\overline{XACK}$ can be used to latch valid data from the RAM array.

Table 1. Pin Descriptions (Continued)

Symbol	Pin No.	Type	Name and Function
$\overline{SACK}$	30	O	**SYSTEM ACKNOWLEDGE:** This output indicates the beginning of a memory access cycle. It can be used as an advanced transfer acknowledge to eliminate wait states. (Note: If a memory access request is made during a refresh cycle, $\overline{SACK}$ is delayed until $\overline{XACK}$ in the memory access cycle).
X_0/OP_2 X_1/CLK	36 37	I/O I/O	**OSCILLATOR INPUTS:** These inputs are designed for a quartz crystal to control the frequency of the oscillator. If X_0/OP_2 is shorted to pin 40 (V_{CC}) or if X_0/OP_2 is connected to +12V through a 1 KΩ resistor then X_1/CLK becomes a TTL input for an external clock. (Note: Crystal mode for the 8203-1 and the 8203-3 only).
16K/$\overline{64K}$	35	I	**MODE SELECT:** This input selects 16K mode or 64K mode. Pins 23–26 change function based on the mode of operation.
V_{CC}	40		**POWER SUPPLY:** +5V.
GND	20		**GROUND.**

FUNCTIONAL DESCRIPTION

The 8203 provides a complete dynamic RAM controller for microprocessor systems as well as expansion memory boards.

The 8203 has two modes, one for 16K dynamic RAMs and one for 64Ks, controlled by pin 35.

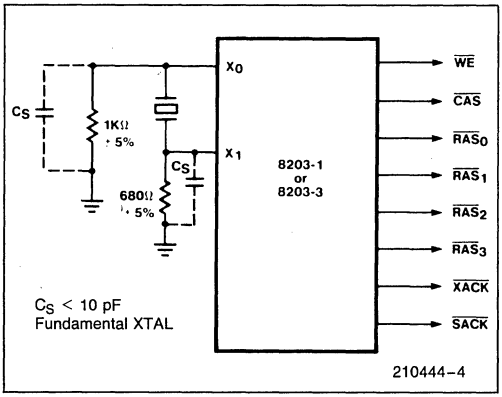


Figure 3. Crystal Operation for the 8203-1 and 8203-3

All 8203 timing is generated from a single reference clock. This clock is provided via an external oscillator or an on-chip crystal oscillator. All output signal transitions are synchronous with respect to this clock reference, except for the trailing edges of the CPU handshake signals $\overline{SACK}$ and $\overline{XACK}$.

CPU memory requests normally use the $\overline{RD}$ and $\overline{WR}$ inputs. The Advanced-Read mode allows ALE and S1 to be used in place of the $\overline{RD}$ input.

Failsafe refresh is provided via an internal timer which generates refresh requests. Refresh requests can also be generated via the REFRQ input.

An on-chip synchronizer/arbiter prevents memory and refresh requests from affecting a cycle in progress. The READ, WRITE, and external REFRESH requests may be asynchronous to the 8203 clock; on-chip logic will synchronize the requests, and the arbiter will decide if the requests should be delayed, pending completion of a cycle in progress.

16K/64 Option Selection

Pin 35 is a strap input that controls the two 8203 modes. Figure 4 shows the four pins that are multiplexed. In 16K mode (pin 35 tied to V_{CC} or left open), the 8203 has two Bank Select inputs to select one of four $\overline{RAS}$ outputs. In this mode, the 8203 is exactly compatible with the Intel 8202A Dynamic RAM Controller. In 64K mode (pin 35 tied to GND), there is only one Bank Select input (pin 26) to select the two $\overline{RAS}$ outputs. More than two banks of 64K dynamic RAMs can be used with external logic.

Other Option Selections

The 8203 has two strapping options. When OP_1 is selected (16K mode only), pin 32 changes from a $\overline{RD}$ input to an S1 input, and pin 34 changes from a REFRQ input to an ALE input. See "Refresh Cycles" and "Read Cycles" for more detail. OP_1 is selected by tying pin 25 to +12V through a 5.1 KΩ resistor on the 8203-1 or 8203-3 only.

When OP_2 is selected, the internal oscillator is disabled and pin 37 changes from a crystal input (X_1) to a CLK input for an external TTL clock. OP_2 is selected by shorting pin 36 (X_0/OP_2) directly to pin 40 (V_{CC}). No current limiting resistor should be used. OP_2 may also be selected by tying pin 36 to +12V through a 1 KΩ resistor.

Refresh Timer

The refresh timer is used to monitor the time since the last refresh cycle occurred. When the appropriate amount of time has elapsed, the refresh timer will request a refresh cycle. External refresh requests will reset the refresh timer.

Refresh Counter

The refresh counter is used to sequentially refresh all of the memory's rows. The 8-bit counter is incremented after every refresh cycle.

Pin #	16K Function	64K Function
23	$\overline{RAS_2}$	Address Output ($\overline{OUT_7}$)
24	Bank Select (B_0)	Address Input (AL_7)
25	Bank Select (B_1)	Address Input (AH_7)
26	$\overline{RAS_3}$	Bank Select (B_0)

Figure 4. 16K/64K Mode Selection

Inputs			Outputs			
	B_1	B_0	$\overline{RAS_0}$	$\overline{RAS_1}$	$\overline{RAS_2}$	$\overline{RAS_3}$
16K Mode	0	0	0	1	1	1
	0	1	1	0	1	1
	1	0	1	1	0	1
	1	1	1	1	1	0
64K Mode	—	0	0	1	—	—
	—	1	1	0	—	—

Figure 5. Bank Selection

Address Multiplexer

The address multiplexer takes the address inputs and the refresh counter outputs, and gates them onto the address outputs at the appropriate time. The address outputs, in conjunction with the $\overline{RAS}$ and $\overline{CAS}$ outputs, determine the address used by the dynamic RAMs for read, write, and refresh cycles. During the first part of a read or write cycle, AL_0–AL_7 are gated to $\overline{OUT_0}$–$\overline{OUT_7}$, then AH_0–AH_7 are gated to the address outputs.

During a refresh cycle, the refresh counter is gated onto the address outputs. All refresh cycles are $\overline{RAS}$-only refresh ($\overline{CAS}$ inactive, $\overline{RAS}$ active).

To minimize buffer delay, the information on the address outputs is inverted from that on the address inputs.

$\overline{OUT_0}$–$\overline{OUT_7}$ do not need inverters or buffers unless additional drive is required.

Synchronizer/Arbiter

The 8203 has three inputs, REFRQ/ALE (pin 34), $\overline{RD}$ (pin 32) and $\overline{WR}$ (pin 31). The $\overline{RD}$ and $\overline{WR}$ inputs allow an external CPU to request a memory read or write cycle, respectively. The REFRQ/ALE input allows refresh requests to be requested external to the 8203.

All three of these inputs may be asynchronous with respect to the 8203's clock. The arbiter will resolve conflicts between refresh and memory requests, for both pending cycles and cycles in progress. Read and write requests will be given priority over refresh requests.

System Operation

The 8203 is always in one of the following states:

a) IDLE
b) TEST Cycle
c) REFRESH Cycle
d) READ Cycle
e) WRITE Cycle

The 8203 is normally in the IDLE state. Whenever one of the other cycles is requested, the 8203 will

Description	Pin #	Normal Function	Option Function
B_1/OP_1 (16K only)/AH_7	25	Bank ($\overline{RAS}$) Select	Advanced-Read Mode (8203-1, -3)
X_0/OP_2	36	Crystal Oscillator (8203-1 and 8203-3)	External Oscillator

Figure 6. 8203 Option Selection

leave the IDLE state to perform the desired cycle. If no other cycles are pending, the 8203 will return to the IDLE state.

Test Cycle

The TEST Cycle is used to check operation of several 8203 internal functions. TEST cycles are requested by activating the $\overline{\text{PCS}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$ inputs. The TEST Cycle will reset the refresh address counter and perform a WRITE Cycle. The TEST Cycle should not be used in normal system operation, since it would affect the dynamic RAM refresh.

Refresh Cycles

The 8203 has two ways of providing dynamic RAM refresh:

1) Internal (failsafe) refresh
2) External (hidden) refresh

Both types of 8203 refresh cycles activate all of the $\overline{\text{RAS}}$ outputs, while $\overline{\text{CAS}}$, $\overline{\text{WE}}$, $\overline{\text{SACK}}$, and $\overline{\text{XACK}}$ remain inactive.

Internal refresh is generated by the on-chip refresh timer. The timer uses the 8203 clock to ensure that refresh of all rows of the dynamic RAM occurs every 2 milliseconds (128 cycles) or every 4 milliseconds (256 cycles). If REFRQ is inactive, the refresh timer will request a refresh cycle every 10–16 microseconds.

External refresh is requested via the REFRQ input (pin 34). External refresh control is not available when the Advanced-Read mode is selected. External refresh requests are latched, then synchronized to the 8203 clock.

The arbiter will allow the refresh request to start a refresh cycle only if the 8203 is not in the middle of a cycle.

When the 8203 is in the idle state a simultaneous memory request and external refresh request will result in the memory request being honored first. This 8203 characteristic can be used to "hide" refresh cycles during system operation. A circuit similar to Figure 7 can be used to decode the CPU's instruction fetch status to generate an external refresh request. The refresh request is latched while the 8203 performs the instruction fetch; the refresh cycle will start immediately after the memory cycle is completed, even if the $\overline{\text{RD}}$ input has not gone inactive. If the CPU's instruction decode time is long enough, the 8203 can complete the refresh cycle before the next memory request is generated.

If the 8203 is not in the idle state then a simultaneous memory request and an external refresh request may result in the refresh request being honored first.

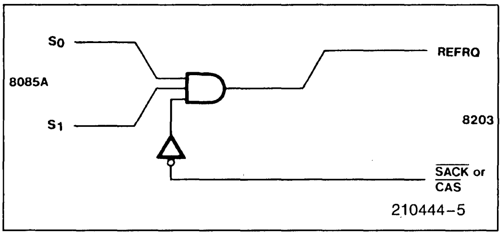


Figure 7. Hidden Refresh

Certain system configurations require complete external refresh requests. If external refresh is requested faster than the minimum internal refresh timer (t_{REF}), then, in effect, all refresh cycles will be caused by the external refresh request, and the internal refresh timer will never generate a refresh request.

Read Cycles

The 8203 can accept two different types of memory Read requests:

1) Normal Read, via the $\overline{\text{RD}}$ input
2) Advanced Read, using the S1 and ALE inputs (16K mode only)

The user can select the desired Read request configuration via the B_1/OP_1 hardware strapping option on pin 25.

	Normal Read	Advanced Read
Pin 25	B_1 Input	OP_1 (+12V)
Pin 32	RD Input	S1 Input
Pin 34	REFRQ Input	ALE Input
# RAM Banks	4 ($\overline{\text{RAS}}_{0-3}$)	2 ($\overline{\text{RAS}}_{2-3}$)
Ext. Refresh	Yes	No

Figure 8. 8203 Read Options

Normal Reads are requested by activating the $\overline{\text{RD}}$ input, and keeping it active until the 8203 responds with an $\overline{\text{XACK}}$ pulse. The $\overline{\text{RD}}$ input can go inactive as soon as the command hold time (t_{CHS}) is met.

Advanced Read cycles are requested by pulsing ALE while S1 is active; if S1 is inactive (low) ALE is ignored. Advanced Read timing is similar to Normal Read timing, except the falling edge of ALE is used as the cycle start reference.

If a Read cycle is requested while a refresh cycle is in progress, then the 8203 will set the internal delayed-SACK latch. When the Read cycle is eventually started, the 8203 will delay the active SACK transition until XACK goes active, as shown in the A.C. timing diagrams. This delay was designed to compensate for the CPU's READY setup and hold times. The delayed-SACK latch is cleared after every READ cycle.

Based on system requirements, either SACK or XACK can be used to generate the CPU READY signal. XACK will normally be used; if the CPU can tolerate an advanced READY, then SACK can be used, but only if the CPU can tolerate the amount of advance provided by SACK. If SACK arrives too early to provide the appropriate number of WAIT states, then either XACK or a delayed form of SACK should be used.

Write Cycles

Write cycles are similar to Normal Read cycles, except for the WE output. WE is held inactive for Read cycles, but goes active for Write cycles. All 8203 Write cycles are "early-write" cycles; WE goes active before CAS goes active by an amount of time sufficient to keep the dynamic RAM output buffers turned off.

General System Considerations

All memory requests (Normal Reads, Advanced Reads, Writes) are qualified by the PCS input. PCS should be stable, either active or inactive, prior to the leading edge of RD, WR, or ALE. Systems which use battery backup should pullup PCS to prevent erroneous memory requests.

In order to minimize propagation delay, the 8203 uses an inverting address multiplexer without latches. The system must provide adequate address setup and hold times to guarantee RAS and CAS setup and hold times for the RAM. The t_{AD} A.C. parameter should be used for this system calculation.

The B_0–B_1 inputs are similar to the address inputs in that they are not latched. B_0 and B_1 should not be changed during a memory cycle, since they directly control which RAS output is activated.

The 8203 uses a two-stage synchronizer for the memory request inputs (RD, WR, ALE), and a separate two stage synchronizer for the external refresh input (REFRQ). As with any synchronizer, there is always a finite probability of metastable states inducing system errors. The 8203 synchronizer was designed to have a system error rate less than 1 memory cycle every three years based on the full operating range of the 8203.

A microprocessor system is concerned when the data is valid after RD goes low. See Figure 9. In order to calculate memory read access times, the dynamic RAM's A.C. specifications must be examined, especially the RAS-access time (t_{RAC}) and the CAS-access time (t_{CAC}). Most configurations will be CAS-access limited; i.e., the data from the RAM will be stable $t_{cc,max}$ (8203) + t_{CAC} (RAM) after a memory read cycle is started. Be sure to add any delays (due to buffers, data latches, etc.) to calculate the overall read access time.

Since the 8203 normally performs "early-write" cycles, the data must be stable at the RAM data inputs by the time CAS goes active, including the RAM's data setup time. If the system does not normally guarantee sufficient write data setup, you must either delay the WR input signal or delay the 8203 WE output.

Delaying the WR input will delay all 8203 timing, including the READY handshake signals, SACK and XACK, which may increase the number of WAIT states generated by the CPU.

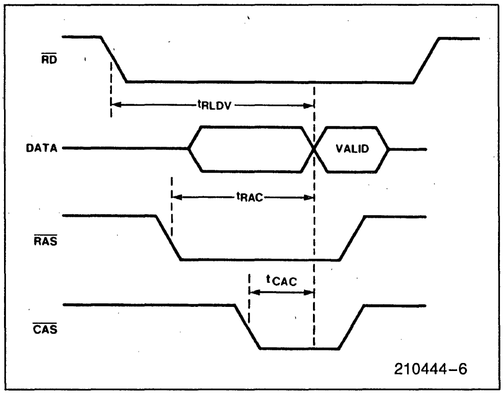


Figure 9. Read Access Time

If the WE output is externally delayed beyond the CAS active transition, then the RAM will use the falling edge of WE to strobe the write data into the RAM. This WE transition should not occur too late during the CAS active transition, or else the WE to CAS requirements of the RAM will not be met.

The RAS$_{0-3}$, CAS, OUT$_{0-7}$, and WE outputs contain on-chip series damping resistors (typically 20Ω) to minimize overshoot.

Some dynamic RAMs require more than 2.4V V_{IH}. Noise immunity may be improved for these RAMs by adding pull-up resistors to the 8203's outputs. Intel RAMs do not require pull-up resistors.

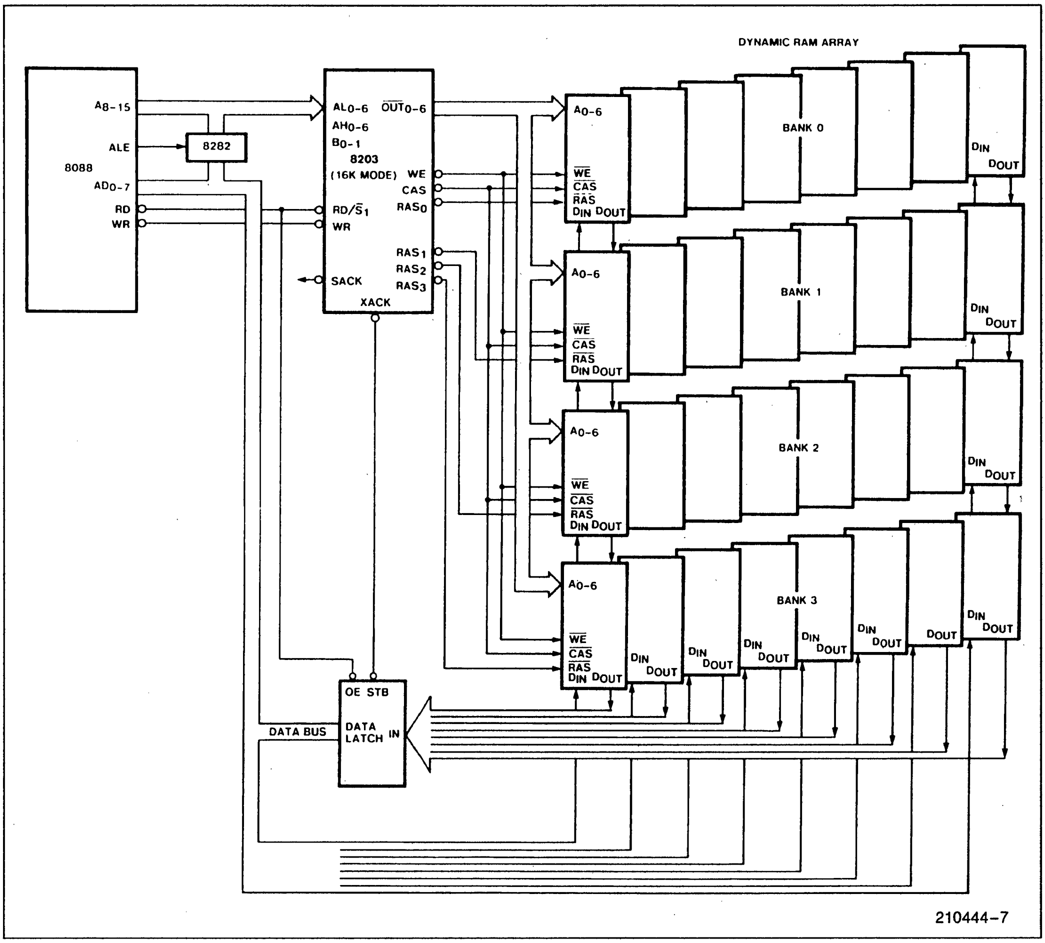


Figure 10. Typical 8088 System

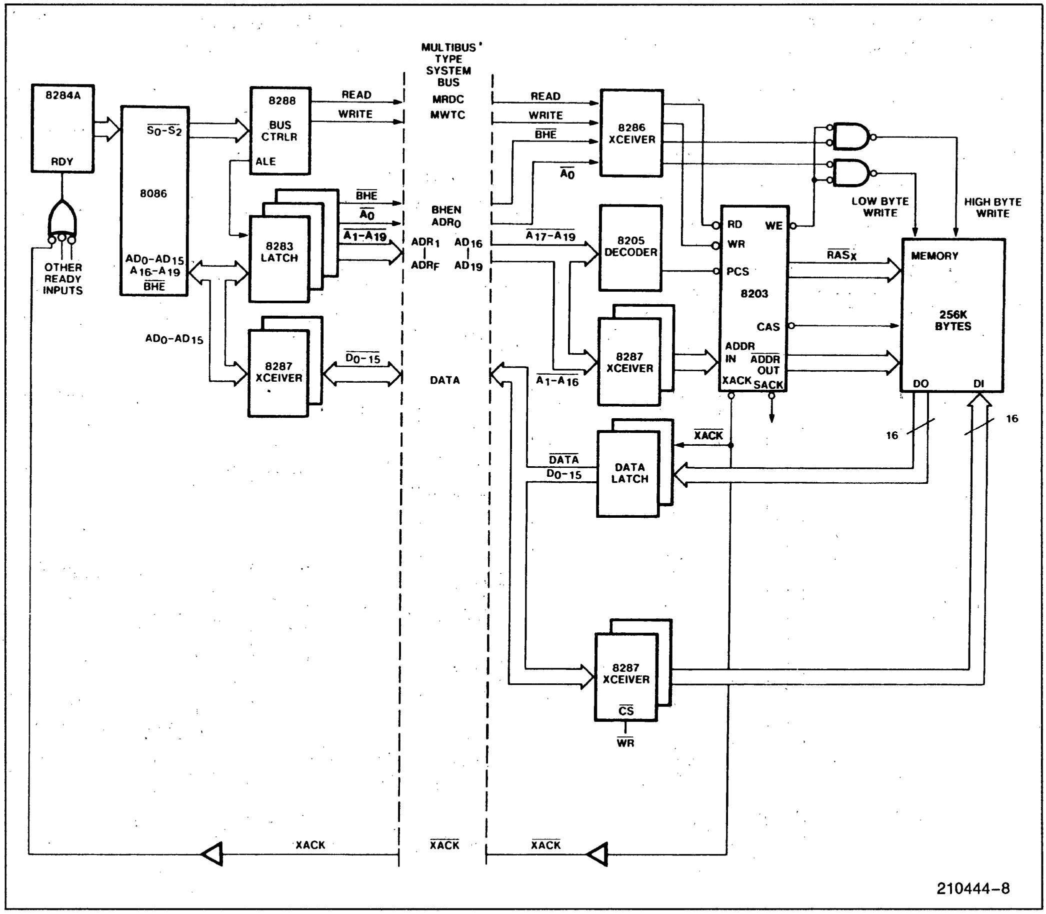


Figure 11. 8086/256K Byte System

ABSOLUTE MAXIMUM RATINGS*

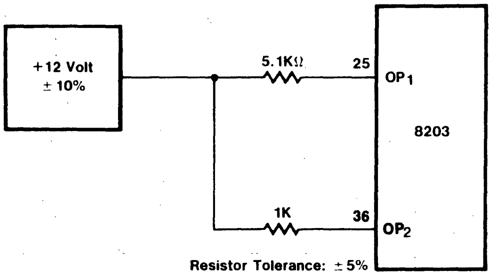
Ambient Temperature Under Bias ...... 0°C to 70°C
Storage Temperature .... −65°C to +150°C
Voltage On Any Pin
With Respect to Ground .... −0.5V to +7V
Power Dissipation .... 1.6 Watts

**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} = 5.0V ±10% (5.0V ±5% for 8203-3); GND = 0V

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{CC}	Input Clamp Voltage		−1.0	V	$I_C = -5$ mA
I_{CC}	Power Supply Current		290	mA	
I_F	Forward Input Current CLK, 64K/16K Mode Select All Other Inputs(3)		 −2.0 −320	 mA μA	 $V_F = 0.45V$ $V_F = 0.45V$
I_R	Reverse Input Current(3)		40	μA	$V_R = V_{CC}$(1, 5)
V_{OL}	Output Low Voltage $\overline{SACK}$, $\overline{XACK}$ All Other Outputs		 0.45 0.45	 V V	 $I_{OL} = 5$ mA $I_{OL} = 3$ mA
V_{OH}	Output High Voltage $\overline{SACK}$, $\overline{XACK}$ All Other Outputs	 2.4 2.6		 V V	$V_{IL} = 0.65V$ $I_{OH} = -1$ mA $I_{OH} = -1$ mA
V_{IL}	Input Low Voltage		0.8	V	$V_{CC} = 5.0V$(2)
V_{IH1}	Input High Voltage	2.0	V_{CC}	V	$V_{CC} = 5.0V$
V_{IH2}	Option Voltage		V_{CC}	V	(4)
C_{IN}	Input Capacitance		30	pF	F = 1 MHz(6) $V_{BIAS} = 2.5V$, $V_{CC} = 5V$

NOTES:
1. I_R = 200 μA for pin 37 (CLK).
2. For test mode $\overline{RD}$ & $\overline{WR}$ must be held at GND.
3. Except for pin 36 in XTAL mode.
4. 8203-1 and 8203-3 support both OP_1 and OP_2, 8203 only supports OP_2.



210444-3

5. I_R = 150 μA for pin 35 (Mode Select 16K/64K).
6. Sampled not 100% tested, T_A = 25°C.

A.C. CHARACTERISTICS

T_J = 0°C to 70°C; V_{CC} = 5V ±10% (5.0V ±5% for 8203-3); GND = 0V
Measurements made with respect to $\overline{\text{RAS}}_0$–$\overline{\text{RAS}}_3$, $\overline{\text{CAS}}$, $\overline{\text{WE}}$, $\overline{\text{OUT}}_0$–$\overline{\text{OUT}}_6$ are at 2.4V and 0.8V. All other pins are measured at 1.5V. All times are in ns.

Symbol	Parameter	Min	Max	Notes
t_P	Clock Period	40	54	
t_{PH}	External Clock High Time	20		
t_{PL}	External Clock Low Time—Above (>) 20 MHz	17		
t_{PL}	External Clock Low Time—Below (≤) 20 MHz	20		
t_{RC}	Memory Cycle Time	$10t_P - 30$	$12t_P$	4, 5
t_{REF}	Refresh Time (128 cycles)	$264t_P$	$288t_P$	
t_{RP}	$\overline{\text{RAS}}$ Precharge Time	$4t_P - 30$		
t_{RSH}	$\overline{\text{RAS}}$ Hold After $\overline{\text{CAS}}$	$5t_P - 30$		3
t_{ASR}	Address Setup to $\overline{\text{RAS}}$	$t_P - 30$		3
t_{RAH}	Address Hold From $\overline{\text{RAS}}$	$t_P - 10$		3
t_{ASC}	Address Setup to $\overline{\text{CAS}}$	$t_P - 30$		3
t_{CAH}	Address Hold from $\overline{\text{CAS}}$	$5t_P - 20$		3
t_{CAS}	$\overline{\text{CAS}}$ Pulse Width	$5t_P - 10$		
t_{WCS}	$\overline{\text{WE}}$ Setup to $\overline{\text{CAS}}$	$t_P - 40$		
t_{WCH}	$\overline{\text{WE}}$ Hold After $\overline{\text{CAS}}$	$5t_P - 35$		8
t_{RS}	$\overline{\text{RD}}$, $\overline{\text{WR}}$, ALE, REFRQ Delay From $\overline{\text{RAS}}$	$5t_P$		2, 6
t_{MRP}	$\overline{\text{RD}}$, $\overline{\text{WR}}$ Setup to $\overline{\text{RAS}}$	0		5
t_{RMS}	REFRQ Setup to $\overline{\text{RD}}$, $\overline{\text{WR}}$	$2t_P$		6
t_{RMP}	REFRQ Setup to $\overline{\text{RAS}}$	$2t_P$		5
t_{PCS}	$\overline{\text{PCS}}$ Setup to $\overline{\text{RD}}$, $\overline{\text{WR}}$, ALE	20		
t_{AL}	S1 Setup to ALE	15		
t_{LA}	S1 Hold From ALE	30		
t_{CR}	$\overline{\text{RD}}$, $\overline{\text{WR}}$, ALE to $\overline{\text{RAS}}$ Delay	$t_P + 30$	$2t_P + 70$	2
t_{CC}	$\overline{\text{RD}}$, $\overline{\text{WR}}$, ALE to $\overline{\text{CAS}}$ Delay	$t_P + 25$	$4t_P + 85$	2
t_{SC}	CMD Setup to Clock	15		1
t_{MRS}	$\overline{\text{RD}}$, $\overline{\text{WR}}$ Setup to REFRQ	5		2
t_{CA}	$\overline{\text{RD}}$, $\overline{\text{WR}}$, ALE to $\overline{\text{SACK}}$ Delay		$2t_P + 47$	2, 9
t_{CX}	$\overline{\text{CAS}}$ to $\overline{\text{XACK}}$ Delay	$5t_P - 25$	$5t_P + 20$	
t_{CS}	$\overline{\text{CAS}}$ to $\overline{\text{SACK}}$ Delay	$5t_P - 25$	$5t_P + 40$	2, 10
t_{ACK}	$\overline{\text{XACK}}$ to $\overline{\text{CAS}}$ Setup	10		
t_{XW}	$\overline{\text{XACK}}$ Pulse Width	$t_P - 25$		7
t_{CK}	$\overline{\text{SACK}}$, $\overline{\text{XACK}}$ Turn-Off Delay		35	
t_{KCH}	CMD Inactive Hold After $\overline{\text{SACK}}$, $\overline{\text{XACK}}$	10		

A.C. CHARACTERISTICS (Continued)

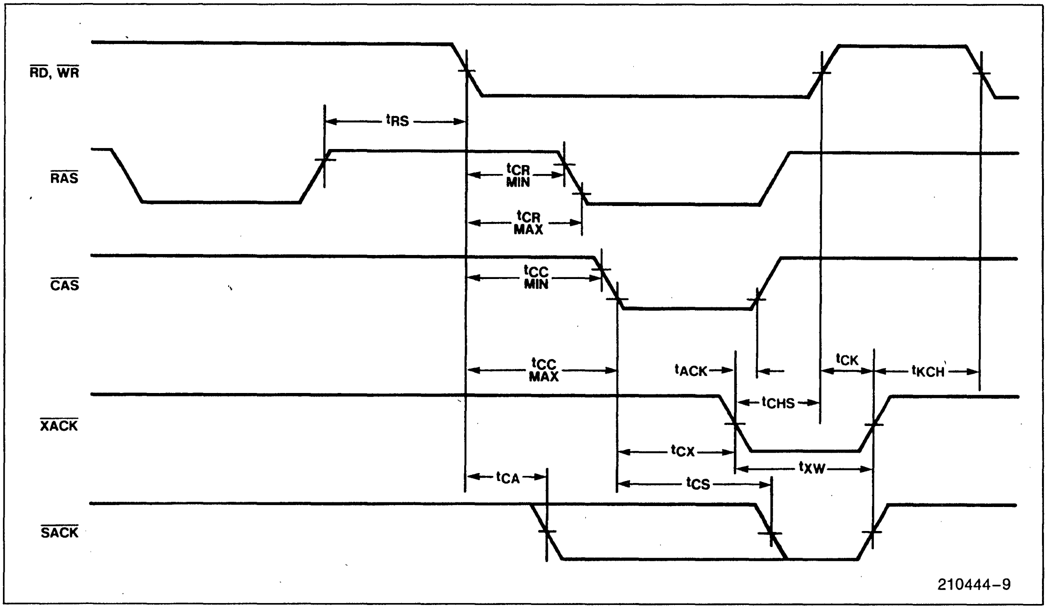
T_J = 0°C to 70°C; V_{CC} = 5V ±10% (5.0V ±5% for 8203-3); GND = 0V
Measurements made with respect to $\overline{RAS_0}-\overline{RAS_3}$, $\overline{CAS}$, $\overline{WE}$, $\overline{OUT_0}-\overline{OUT_6}$ are at 2.4V and 0.8V. All other pins are measured at 1.5V. All times are in ns.

Symbol	Parameter	Min	Max	Notes
t_{LL}	REFRQ Pulse Width	20		
t_{CHS}	CMD Hold Time	30		11
t_{RFR}	REFRQ to $\overline{RAS}$ Delay		$4t_P + 100$	6
t_{WW}	$\overline{WR}$ to $\overline{WE}$ Delay	0	50	8
t_{AD}	CPU Address Delay	0	40	3

NOTES:
1. t_{SC} is a reference point only. ALE, $\overline{RD}$, $\overline{WR}$, and REFRQ inputs do not have to be externally synchronized to 8203 clock.
2. If t_{RS} min and t_{MRS} min are met then t_{CA}, t_{CR}, and t_{CC} are valid, otherwise t_{CS} is valid.
3. t_{ASR}, t_{RAH}, t_{ASC}, t_{CAH}, and t_{RSH} depend upon B_0-B_1 and CPU address remaining stable throughout the memory cycle. The address inputs are not latched by the 8203.
4. For back-to-back refresh cycles, t_{RC} max = 13 t_P.
5. t_{RC} max is valid only if t_{RMP} min is met (READ, WRITE followed by REFRESH) or t_{MRP} min is met (REFRESH followed by READ, WRITE).
6. t_{RFR} is valid only if t_{RS} min and t_{RMS} min are met.
7. t_{XW} min applies when $\overline{RD}$, $\overline{WR}$ has already gone high. Otherwise $\overline{XACK}$ follows $\overline{RD}$, $\overline{WR}$.
8. $\overline{WE}$ goes high according to t_{WCH} or t_{WW}, whichever occurs first.
9. t_{CA} applies only when in normal $\overline{SACK}$ mode.
10. t_{CS} applies only when in delayed $\overline{SACK}$ mode.
11. t_{CHS} must be met only to ensure a $\overline{SACK}$ active pulse when in delayed $\overline{SACK}$ mode. $\overline{XACK}$ will always be activated for at least t_{XW} (t_P − 25 ns). Violating t_{CHS} min does not otherwise affect device operation.

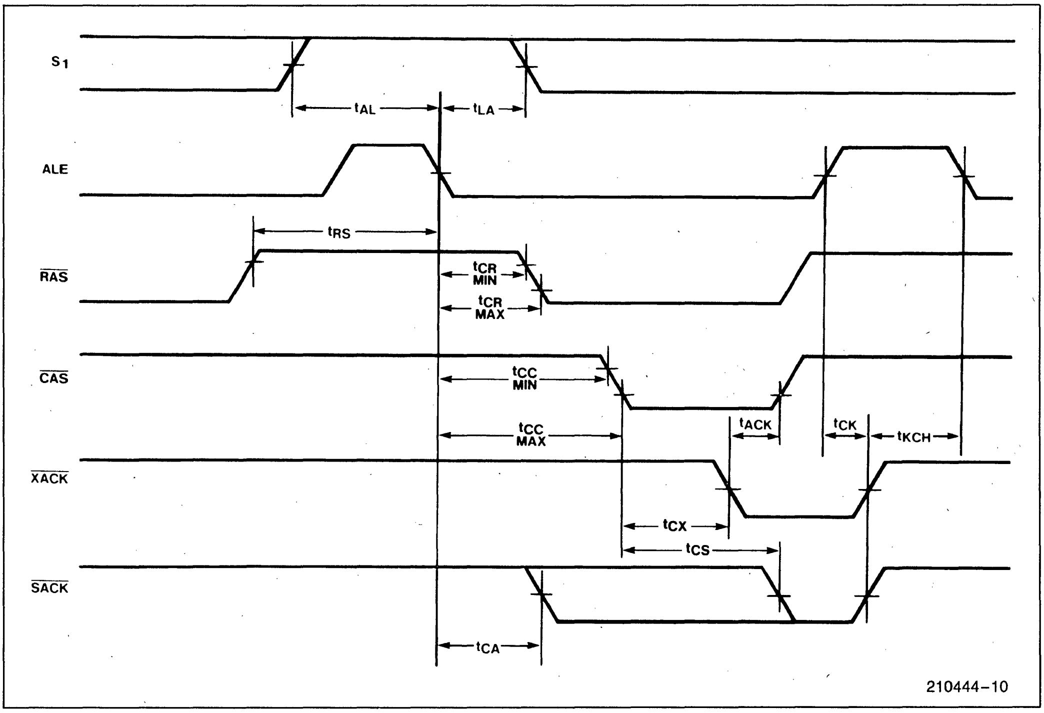
WAVEFORMS

Normal Read or Write Cycle

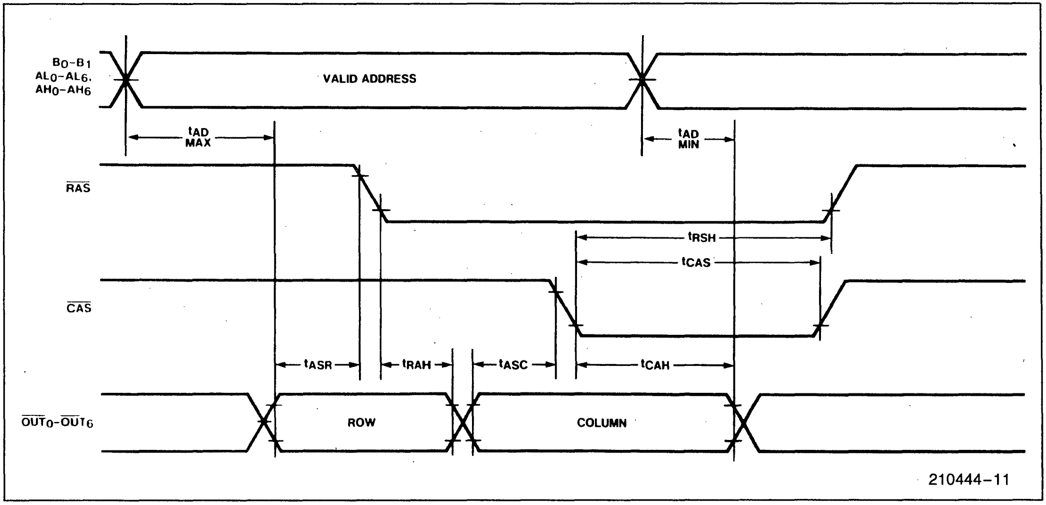


WAVEFORMS (Continued)

Advanced Read Mode

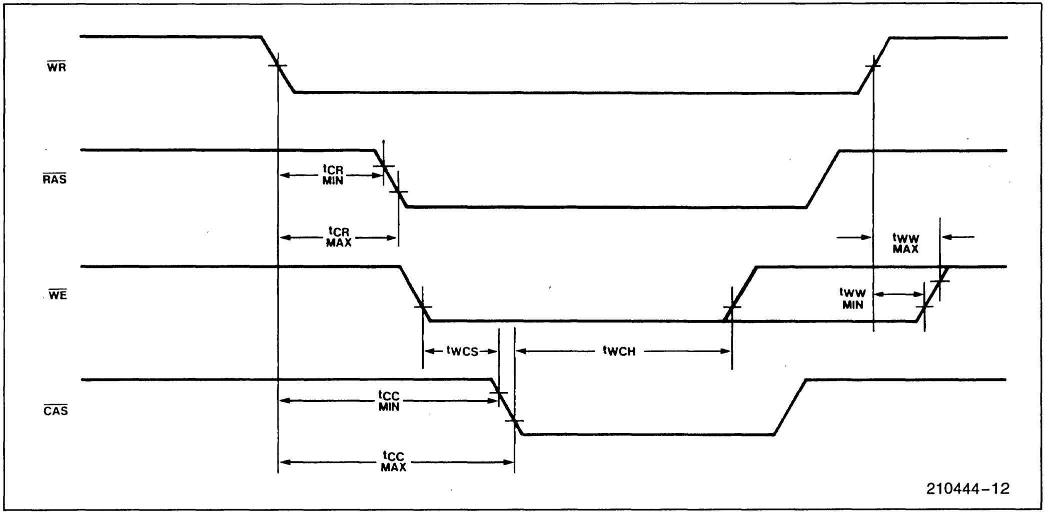


Memory Compatibility Timing



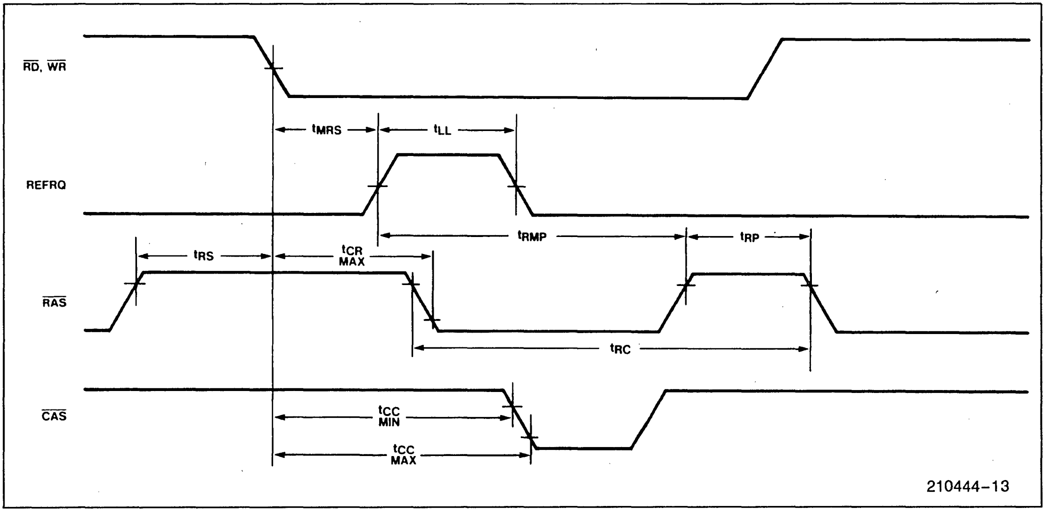
WAVEFORMS (Continued)

Write Cycle Timing



210444-12

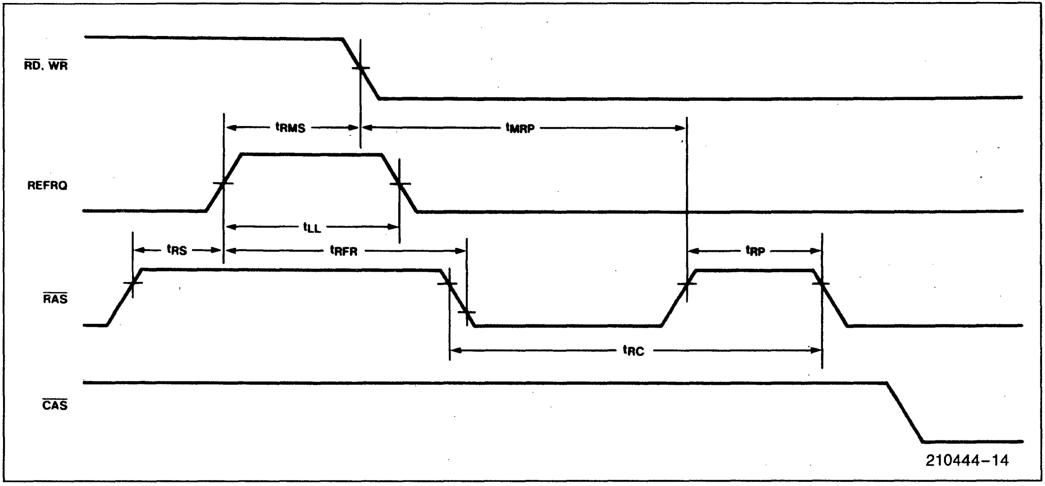
Read or Write Followed By External Refresh



210444-13

WAVEFORMS (Continued)

External Refresh Followed By Read or Write



Clock and System Timing

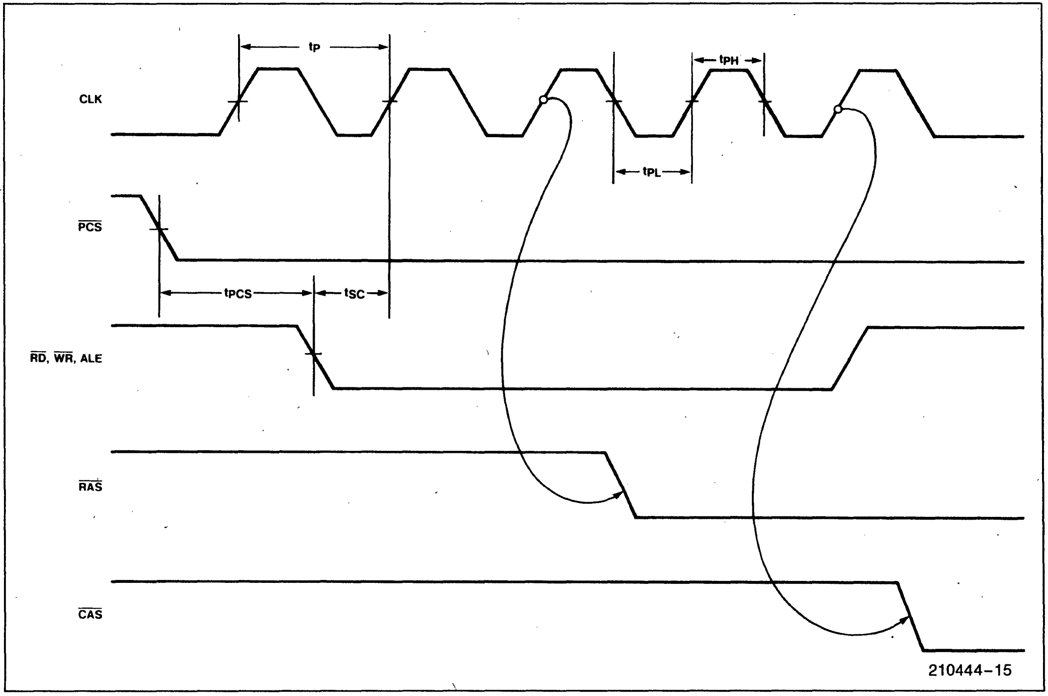
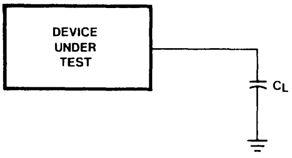


Table 2. 8203 Output Loading. All specifications are for the Test Load unless otherwise noted.

Pin	Test Load
$\overline{SACK}$, $\overline{XACK}$	$C_L = 30$ pF
$\overline{OUT_0}-\overline{OUT_6}$	$C_L = 160$ pF
$\overline{RAS_0}-\overline{RAS_3}$	$C_L = 60$ pF
$\overline{WE}$	$C_L = 224$ pF
$\overline{CAS}$	$C_L = 320$ pF

A.C. TESTING LOAD CIRCUIT



210444-16

NOTE:
C_L includes jig capacitance.

The typical rising and falling characteristic curves for the $\overline{OUT}$, $\overline{RAS}$, $\overline{CAS}$ and $\overline{WE}$ output buffers can be used to determine the effects of capacitive loading on the A.C. Timing Parameters. Using this design tool in conjunction with the timing waveforms, the designer can determine typical timing shifts based on system capacitive load.

Example: Find the effect on t_{CR} and t_{CC} using 32 64K Dynamic RAMs configured in 2 banks.

1) Determine the typical $\overline{RAS}$ and $\overline{CAS}$ capacitance:

   From the data sheet $\overline{RAS}$ = 5 pF and $\overline{CAS}$ = 5 pF.

   ∴ $\overline{RAS}$ load = 80 pF + board capacitance.

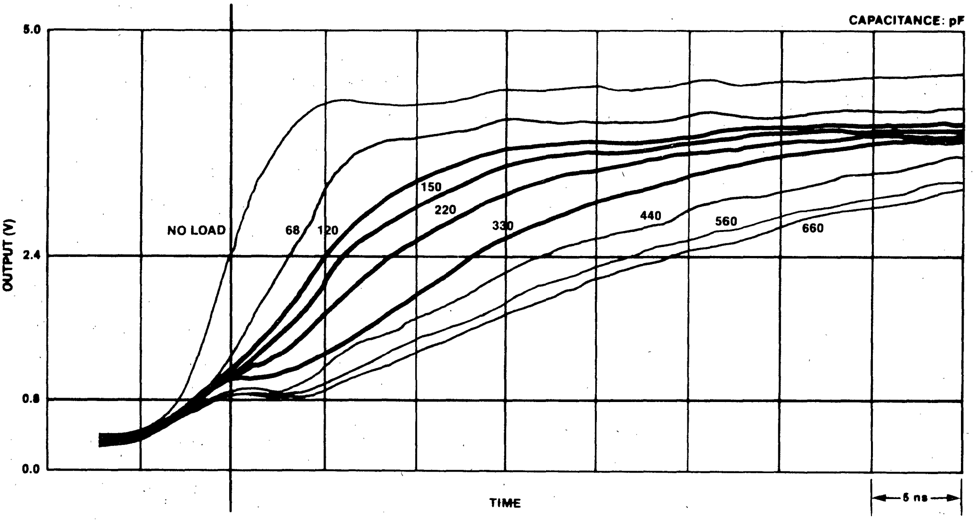
   $\overline{CAS}$ load = 160 pF + board capacitance.

   Assume 2 pF/in (trace length) for board capacitance and for this example 4 inches for $\overline{RAS}$ and 8 inches for $\overline{CAS}$.

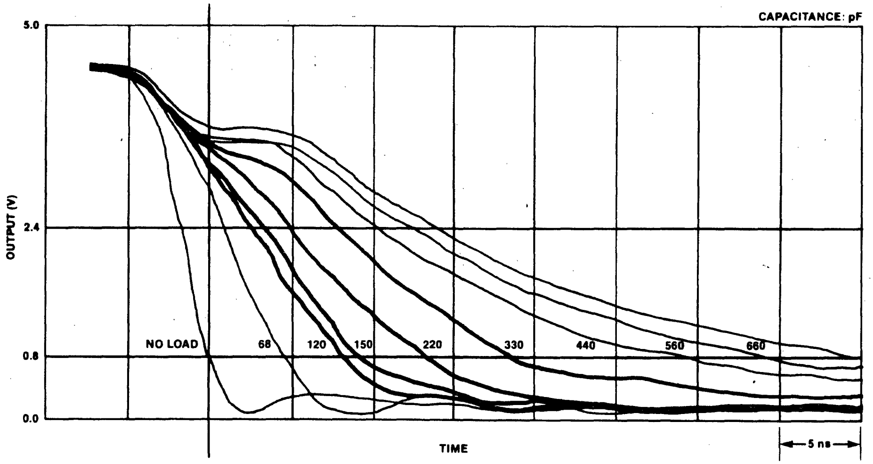
2) From the waveform diagrams, we determine that the falling edge timing is needed for t_{CR} and t_{CC}. Next find the curve that *best* approximates the test load; i.e., 68 pF for $\overline{RAS}$ and 330 pF for $\overline{CAS}$.

3) If we use 88 pF for $\overline{RAS}$ loading, then t_{CR} (min.) spec should be increased by about 1 ns, and t_{CR} (max.) spec should be increased by *about* 2 ns. Similarly if we use 176 pF for $\overline{CAS}$, then t_{CC} (min.) should decrease by 3 ns and t_{CC} (max.) should decrease by about 7 ns.

A.C. CHARACTERISTICS FOR DIFFERENT CAPACITIVE LOADS



210444-17



210444-18

NOTE:
Use the Test Load as the base capacitance for estimating timing shifts for system critical timing parameters.

MEASUREMENT CONDITIONS

Pins not measured are loaded with the Test Load capacitance

T_A = 25°C V_{CC} = +5V t_P = 50 ns

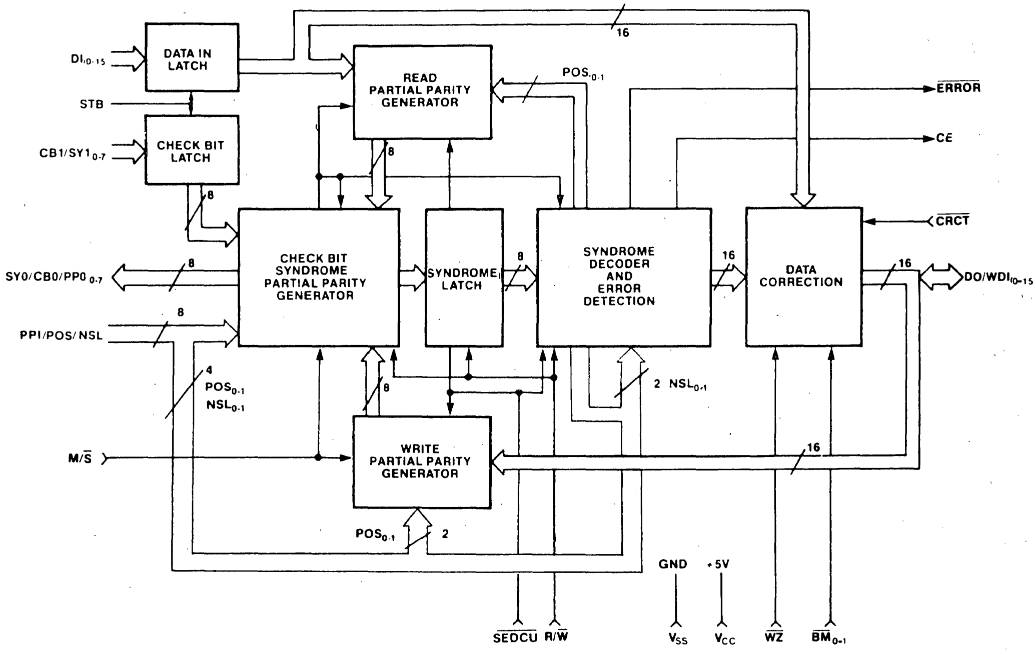
8206
ERROR DETECTION AND CORRECTION UNIT

- **Detects All Single Bit, and Double Bit and Most Multiple Bit Errors**
- **Corrects All Single Bit Errors**
- **3 Selections**

3 Selections	8206-1	8206
Detection	35 ns	42 ns
Correction	55 ns	67 ns

- **Syndrome Outputs for Error Logging**
- **Automatic Error Scrubbing with 8207**
- **Expandable to Handle 80 Bit Memories**
- **Separate Input and Output Busses—No Timing Strobes Required**
- **Supports Read With and Without Correction, Writes, Partial (Byte) Writes, and Read-Modify-Writes**
- **HMOS III Technology for Low Power**
- **68 Pin Leadless JEDEC Package**
- **68 Pin Grid Array Package**

The HMOS 8206 Error Detection and Correction Unit is a high-speed device that provides error detection and correction for memory systems (static and dynamic) requiring high reliability and performance. Each 8206 handles 8 or 16 data bits and up to 8 check bits. 8206's can be cascaded to provide correction and detection for up to 80 bits of data. Other 8206 features include the ability to handle byte writes, memory initialization, and error logging.



205220-1

Figure 1. 8206 Block Diagram

September 1987
Order Number: 205220-008

Table 1. 8206 Pin Description

Symbol	Pin No.	Type	Name and Function
DI_{0-15}	1, 68-61, 59-53	I	**DATA IN:** These inputs accept a 16 bit data word from RAM for error detection and/or correction.
CBI/SYI_0 CBI/SYI_1 CBI/SYI_2 CBI/SYI_3 CBI/SYI_4 CBI/SYI_5 CBI/SYI_6 CBI/SYI_7	5 6 7 8 9 10 11 12	I I I I I I I I	**CHECK BITS IN/SYNDROME IN:** In a single 8206 system,or in the master in a multi-8206 system, these inputs accept the check bits (5 to 8) from the RAM. In a single 8206 16 bit system, CBI_{0-5} are used. In slave 8206's these inputs accept the syndrome from the master.
DO/WDI_0 DO/WDI_1 DO/WDI_2 DO/WDI_3 DO/WDI_4 DO/WDI_5 DO/WDI_6 DO/WDI_7 DO/WDI_8 DO/WDI_9 DO/WDI_{10} DO/WDI_{11} DO/WDI_{12} DO/WDI_{13} DO/WDI_{14} DO/WDI_{15}	51 50 49 48 47 46 45 44 42 41 40 39 38 37 36 35	I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O	**DATA OUT/WRITE DATA IN:** In a read cycle, data accepted by DI_{0-15} appears at these outputs corrected if $\overline{CRCT}$ is low, or uncorrected if $\overline{CRCT}$ is high. The $\overline{BM}$ inputs must be high to enable the output buffers during the read cycle. In a write cycle, data to be written into the RAM is accepted by these inputs for computing the write check bits. In a partial-write cycle, the byte not to be modified appears at either DO_{0-7} if $\overline{BM}_0$ is high, or DO_{8-15} if $\overline{BM}_1$ is high, for writing to the RAM. When $\overline{WZ}$ is active, it causes the 8206 to output all zeros at DO_{0-15}, with the proper write check bits on CBO.
$SYO/CBO/PPO_0$ $SYO/CBO/PPO_1$ $SYO/CBO/PPO_2$ $SYO/CBO/PPO_3$ $SYO/CBO/PPO_4$ $SYO/CBO/PPO_5$ $SYO/CBO/PPO_6$ $SYO/CBO/PPO_7$	23 24 25 27 28 29 30 31	O O O O O O O O	**SYNDROME OUT/CHECK BITS OUT/PARTIAL PARITY OUT:** In a single 8206 system, or in the master in a multi-8206 system, the syndrome appears at these outputs during a read. During a write, the write check bits appear. In slave 8206's the partial parity bits used by the master appear at these outputs. The syndrome is latched (during read-modify-writes) by $R/\overline{W}$ going low.
PPI_0/POS_0 PPI_1/POS_1	13 14	I I	**PARTIAL PARITY IN/POSITION:** In the master in a multi-8206 system, these inputs accept partial parity bits 0 and 1 from the slaves. In a slave 8206 these inputs inform it of its position within the system (1 to 4). Not used in a single 8206 system.
PPI_2/NSL_0 PPI_3/NSL_1	15 16	I I	**PARTIAL PARITY IN/NUMBER OF SLAVES:** In the master in a multi-8206 system, these inputs accept partial parity bits 2 and 3 from the slaves. In a multi-8206 system these inputs are used in slave number 1 to tell it the total number of slaves in the system (1 to 4). Not used in other slaves or in a single 8206 system.
PPI_4CE	17	I/O	**PARTIAL PARITY IN/CORRECTABLE ERROR:** In the master in a multi-8206 system this pin accepts partial parity bit 4. In slave number 1 only, or in a single 8206 system, this pin outputs the correctable error flag. CE is latched by $R/\overline{W}$ going low. Not used in other slaves.

Table 1. 8206 Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
PPI_5 PPI_6 PPI_7	18 19 20	I I I	**PARTIAL PARITY IN:** In the master in a multi-8206 system these pins accept partial parity bits 5 to 7. The number of partial parity bits equals the number of check bits. Not used in single 8206 systems or in slaves.
$\overline{ERROR}$	22	O	**ERROR:** This pin outputs the error flag in a single 8206 system or in the master of a multi-8206 system. It is latched by $R/\overline{W}$ going low. Not used in slaves.
$\overline{CRCT}$	52	I	**CORRECT:** When low this pin causes data correction during a read or read-modify-write cycle. When high, it causes error correction to be disabled, although error checking is still enabled.
$\overline{STB}$	2	I	**STROBE:** STB is an input control used to strobe data at the DI inputs and check-bits at the CBI/SYI inputs. The signal is active high to admit the inputs. The signals are latched by the high-to-low transition of STB.
$\overline{BM}_0$ $\overline{BM}_1$	33 32	I I	**BYTE MARKS:** When high, the Data Out pins are enabled for a read cycle. When low, the Data Out buffers are tristated for a write cycle. $\overline{BM}_0$ controls DO_{0-7}, while $\overline{BM}_1$ controls DO_{8-15}. In partial (byte) writes, the byte mark input is low for the new byte to be written.
$R/\overline{W}$	21	I	**READ/WRITE:** When high this pin causes the 8206 to perform detection and correction (if $\overline{CRCT}$ is low). When low, it causes the 8206 to generate check-bits. On the high-to-low transition the syndrome is latched internally for read-modify-write cycles.
$\overline{WZ}$	34	I	**WRITE ZERO:** When low this input overrides the $\overline{BM}_{0-1}$ and $R/\overline{W}$ inputs to cause the 8206 to output all zeros at DO_{0-15} with the corresponding check-bits at CBO_{0-7}. Used for memory initialization.
$M/\overline{S}$	4	I	**MASTER/SLAVE:** Input tells the 8206 whether it is a master (high) or a slave (low).
$\overline{SEDCU}$	3	I	**SINGLE EDC UNIT:** Input tells the master whether it is operating as a single 8206 (low) or as the master in a multi-8206 system (high). Not used in slaves.
V_{CC}	60	I	**POWER SUPPLY:** +5V
V_{SS}	26	I	**LOGIC GROUND**
V_{SS}	43	I	**OUTPUT DRIVER GROUND**

FUNCTIONAL DESCRIPTION

The 8206 Error Detection and Correction Unit provides greater memory system reliability through its ability to detect and correct memory errors. It is a single chip device that can detect and correct all single bit errors and detect all double bit and some higher multiple bit errors. Some other odd multiple bit errors (e.g., 5 bits in error) are interpreted as single bit errors, and the CE flag is raised. While some even multiple bit errors (e.g., 4 bits in error) are interpreted as no error, most are detected as double bit errors. This error handling is a function of the number of check bits used by the 8206 (see Figure 2) and the specific Hamming code used. Errors in check bits are not distinguished from errors in a word.

For more information on error correction codes, see Intel Application Notes AP-46 and AP-73.

A single 8206 handles 8 or 16 bits of data, and up to 5 8206's can be cascaded in order to handle data paths of 80 bits. For a single 8206 8 bit system, the DI_{8-15}, DO/WDI_{8-15} and $\overline{BM}_1$ inputs are grounded. See the Multi-Chip systems section for information on 24-80 bit systems.

The 8206 has a "flow through" architecture. It supports two kinds of error correction architecture: 1) Flow-through, or correct-always; and 2) Parallel, or check-only. These are two separate 16-pin busses, one to accept data from the RAM (DI) and the other to deliver corrected data to the system bus (DO/WDI). The logic is entirely combinatorial during a read cycle. This is in contrast to an architecture with only one bus, with bidirectional bus drivers that must first read the data and then be turned around to output the corrected data. The latter architecture typically requires additional hardware (latches and/or transceivers) and may be slower in a system due to timing skews of control signals.

Data Word Bits	Check Bits
8	5
16	6
24	6
32	7
40	7
48	8
56	8
64	8
72	8
80	8

Figure 3. Number of Check Bits Used by 8206

READ CYCLE

With the $R/\overline{W}$ pin high, data is received from the RAM outputs into the DI pins where it is optionally latched by the STB signal. Check bits are generated from the data bits and compared to the check bits read from the RAM into the CBI pins. If an error is detected the $\overline{ERROR}$ flag is activated and the correctable error flag (CE) is used to inform the system whether the error was correctable or not. With the $\overline{BM}$ inputs high, the word appears corrected at the DO pins if the error was correctable, or unmodified if the error was uncorrectable.

If more than one 8206 is being used, then the check bits are read by the master. The slaves generate a partial parity output (PPO) and pass it to the master. The master 8206 then generates and returns the syndrome to the slaves (SYO) for correction of the data.

The 8206 may alternatively be used in a "check-only" mode with the $\overline{CRCT}$ pin left high. With the correction facility turned off, the propagation delay from memory outputs to 8206 outputs is significantly shortened. In this mode the 8206 issues an $\overline{ERROR}$ flag to the CPU, which can then perform one of several options: lengthen the current cycle for correction, restart the instruction, perform a diagnostic routine, etc.

A syndrome word, five to eight bits in length and containing all necessary information about the existence and location of an error, is made available to the system at the SYO_{0-7} pins. Error logging may be accomplished by latching the syndrome and the memory address of the word in error.

WRITE CYCLE

For a full write, in which an entire word is written to memory, the data is written directly to the RAM, bypassing the 8206. The same data enters the 8206 through the WDI pins where check bits are generated. The Byte Mark inputs must be low to tristate the DO drivers. The check bits, 5 to 8 in number, are then written to the RAM through the CBO pins for storage along with the data word. In a multi-chip system, the master writes the check bits using partial parity information from the slaves.

In a partial write, part of the data word is overwritten, and part is retained in memory. This is accomplished by performing a read-modify-write cycle. The complete old word is read into the 8206 and corrected, with the syndrome internally latched by $R/\overline{W}$ going low. Only that part of the word not to be modified is output onto the DO pins, as controlled by the Byte Mark inputs. That portion of the word to be overwritten is supplied by the system bus. The 8206 then calculates check bits for the new word, using the byte from the previous read and the new byte from the system bus, and writes them to the memory.

READ-MODIFY-WRITE CYCLES

Upon detection of an error the 8206 may be used to correct the bit in error in memory. This reduces the probability of getting multiple-bit errors in subsequent read cycles. This correction is handled by executing read-modify-write cycles.

The read-modify-write cycle is controlled by the R/$\overline{W}$ input. After (during) the read cycle, the system dynamic RAM controller or CPU examines the 8206 $\overline{ERROR}$ and CE outputs to determine if a correctable error occurred. If it did, the dynamic RAM controller or CPU forces R/$\overline{W}$ low, telling the 8206 to latch the generated syndrome and drive the corrected check bits onto the CBO outputs. The corrected data is available on the DO pins. The DRAM controller then writes the corrected data and corresponding check bits into memory.

The 8206 may be used to perform read-modify-writes in one or two RAM cycles. If it is done in two cycles, the 8206 latches are used to hold the data and check bits from the read cycle to be used in the following write cycle. The Intel 8207 Dual Port Dynamic RAM controller allows read-modify-write cycles in one memory cycle. See the System Environment section.

INITIALIZATION

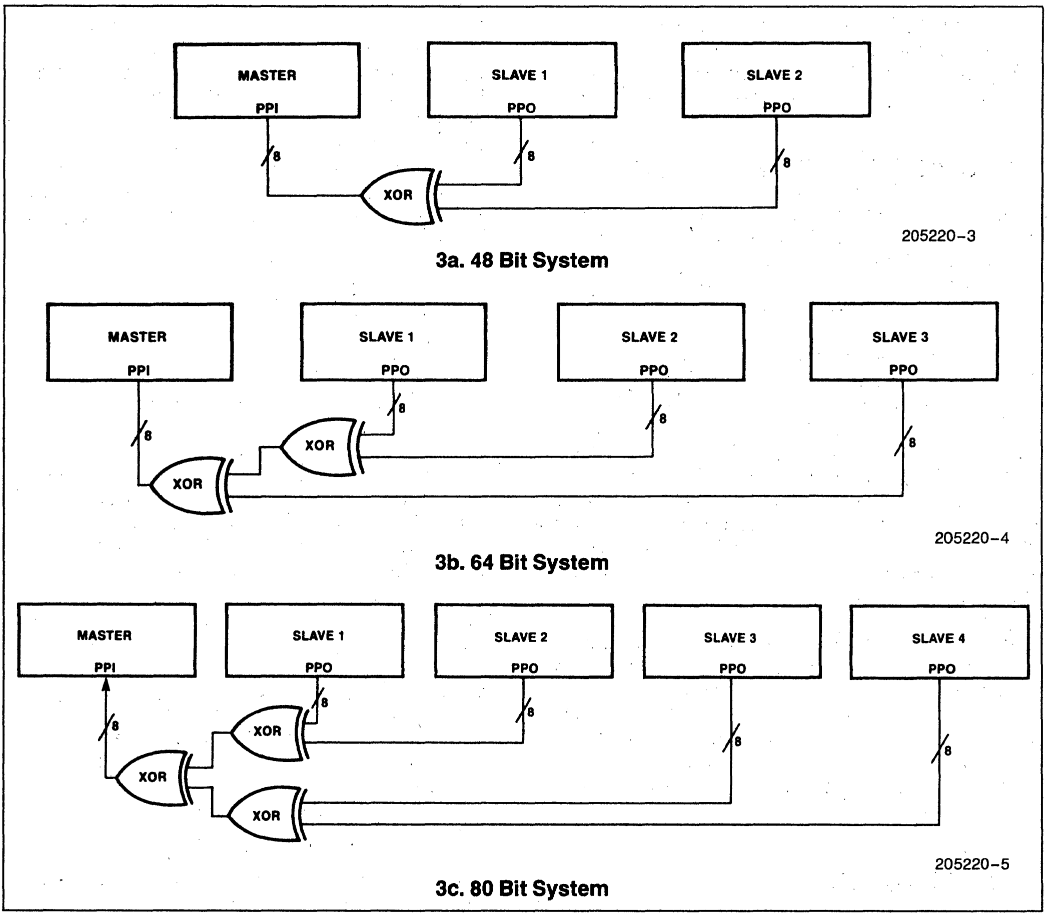
A memory system operating with ECC requires some form of initialization at system power-up in order to set valid data and check bit information in memory. The 8206 supports memory initialization by the write zero function. By activating the $\overline{WZ}$ pin, the 8206 will write a data pattern of zeros and the associated check bits in the current write cycle. By thus writing to all memory at power-up, a controller can set memory to valid data and check bits. Massive memory failure, as signified by both data and check bits all ones or zeros, will be detected as an uncorrectable error.

MULTI-CHIP SYSTEMS

A single 8206 handles 8 or 16 bits of data and 5 or 6 check bits, respectively. Up to 5 8206's can be cascaded for 80 bit memories with 8 check bits.

When cascaded, one 8206 operates as a master, and all others as slaves. As an example, during a read cycle in a 32 bit system with one master and one slave, the slave calculates parity on its portion of the word—"partial parity"—and presents it to the master through the PPO pins. The master combines the partial parity from the slave with the parity it calculated from its own portion of the word to generate the syndrome. The syndrome is then returned by the master to the slave for error correction. In systems with more than one slave the above description continues to apply, except that the partial parity outputs of the slaves must be XOR'd externally. Figure 4 shows the necessary external logic for multi-chip systems. Write and read-modify-write cycles are carried out analogously. See the System Operation section for multi-chip wiring diagrams.

There are several pins used to define whether the 8206 will operate as a master or a slave. Tables 3 and 4 illustrate how these pins are tied.



3a. 48 Bit System

3b. 64 Bit System

3c. 80 Bit System

Figure 4. External Logic for Multi-Chip Sytems

Table 3. Master/Slave Pin Assignments

Pin No.	Pin Name	Master	Slave 1	Slave 2	Slave 3	Slave 4
4	$M/\overline{S}$	+5V	gnd	gnd	gnd	gnd
3	$\overline{SEDCU}$	+5V	+5V	+5V	+5V	+5V
13	PPI_0/POS_0	PPI	gnd	+5V	gnd	+5V
14	PPI_1/POS_1	PPI	gnd	gnd	+5V	+5V
15	PPI_2/NSL_0	PPI	*	+5V	+5V	+5V
16	PPI_3/NSL_1	PPI	*	+5V	+5V	+5V

NOTE:
Pins 13, 14, 15, 16 have internal pull-up resistors and may be left as N.C. where specified as connecting to +5V.

Table 4. NSL Pin Assignments for Slave 1

	Number of Slaves			
Pin	1	2	3	4
PPI_2/NSL_0	GND	+5V	GND	+5V
PPI_3/NSL_1	GND	GND	+5V	+5V

The timing specifications for multi-chip systems must be calculated to take account of the external XOR gating in 3, 4 and 5-chip systems. Let tXOR be the delay for a single external TTL XOR gate. Then the following equations show how to calculate the relevant timing parameters for 2-chip (n =0), 3-chip (n = 1), 4-chip (n = 2), and 5-chip (n = 2) systems:

Data-in to corrected data-out (read cycle) =

TDVSV + TPVSV +TSVQV + ntXOR

Data-in to error flag (read cycle) =

TDVSV + TPVEV + ntXOR

Data-in to correctable error flag (read cycle) =

TDVSV + TPVSV + TSVCV + ntXOR

Write data to check-bits valid (full write cycle) =

TQVQV + TPVSV + ntXOR

Data-in to check-bits valid (read-mod-write cycle) =

TDVSV + TPVSV + TSVQV + TQVQV + TPVSV + 2ntXOR

Data-in to check-bits valid (non-correcting read-modify-write cycle) =

TDVQU + TQVQV + TPVSV + ntXOR

HAMMING CODE

The 8206 uses a modified Hamming code which was optimized for multi-chip EDCU systems. The code is such that partial parity is computed by all 8206's in parallel. No 8206 requires more time for propagation through logic levels than any other one, and hence no one device becomes a bottleneck in the parity operation. However, one or two levels of external TTL XOR gates are required in systems with three to five chips. The code appears in Table 5. The check bits are derived from the table by XORing or XNORing together the bits indicated by 'X's in each row corresponding to a check bit. For example, check bit 0 in the MASTER for data word 1000110101101011 will be "0". It should be noted that the 8206 will detect the gross-error condition of all lows or all highs.

Error correction is accomplished by identifying the bad bit and inverting it. Table 5 can also be used as an error syndrome table by replacing the 'X's with '1's. Each column then represents a different syndrome word, and by locating the column corresponding to a particular syndrome the bit to be corrected may be identified. If the syndrome cannot be located then the error cannot be corrected. For example, if the syndrome word is 00110111, the bit to be corrected is bit 5 in the slave one data word (bit 21).

The syndrome decoding is also summarized in Tables 6 and 7 which can be used for error logging. By finding the appropriate syndrome word (starting with bit zero, the least significant bit), the result is either: 1) no error; 2) an identified (correctable) single bit error; 3) a double bit error; or 4) a multi-bit uncorrectable error.

Table 5. Modified Hamming Code Check Bit Generation

Check bits are generated by XOR'ing (except for the CB0 and CB1 data bits, which are XNOR'ed in the Master) the data bits in the rows corresponding to the check bits. Note there are 6 check bits in a 16-bit system, 7 in a 32-bit system, and 8 in 48-or-more-bit systems.

BYTE NUMBER		0								1								OPERATION
BIT NUMBER		0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
CHECK BITS	CB0 =	x	x	-	x	-	x	x	-	x	-	-	x	-	x	-	-	XNOR
	CB1 =	x	-	x	-	-	x	-	x	-	x	-	x	x	-	x	-	XNOR
	CB2 =	-	x	x	-	x	-	x	x	-	-	x	-	x	-	-	x	XOR
	CB3 =	x	x	x	x	x	-	-	-	x	x	x	-	-	-	-	-	XOR
	CB4 =	-	-	-	x	x	x	x	x	-	-	-	-	-	x	x	x	XOR
	CB5 =	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	XOR
	CB6 =	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	XOR
	CB7 =	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	XOR
DATA BITS		0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	
		0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	
		16 BIT OR MASTER																

2								3								OPERATION
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
-	x	x	x	-	x	x	-	-	x	x	-	-	x	-	-	XOR
x	x	x	-	-	x	-	x	x	x	-	-	-	-	-	x	XOR
-	x	x	x	-	x	x	x	-	-	x	x	-	-	-	-	XOR
x	x	-	-	x	-	x	x	x	-	-	x	x	-	-	-	XOR
x	x	-	-	x	x	x	x	-	-	-	-	x	-	x	-	XOR
-	-	-	x	x	x	x	x	-	-	-	-	-	x	x	x	XOR
-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	XOR
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	XOR
1	1	1	1	2	2	2	2	2	2	2	2	2	2	3	3	
6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	
SLAVE #1																

BYTE NUMBER		4								5								6								7								8								9								OPERATION
BIT NUMBER		0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
CHECK BITS	CB0 =	x	x	-	x	-	x	x	-	x	-	-	x	-	x	-	-	x	-	x	-	x	x	-	-	x	-	x	x	-	-	x	-	-	x	x	x	-	x	x	-	-	x	x	-	-	x	-	-	XOR
	CB1 =	x	-	x	-	-	x	-	x	-	x	-	x	x	-	x	-	-	x	x	-	-	-	x	x	x	x	x	-	-	-	x	-	-	x	x	x	-	x	x	x	-	-	x	x	-	-	-	-	XOR
	CB2 =	-	x	x	-	x	-	x	x	-	-	x	-	x	-	-	x	-	x	x	x	-	x	x	-	-	x	x	-	-	x	-	-	x	-	-	x	-	x	x	-	-	x	x	-	-	x	-	x	XOR
	CB3 =	x	x	x	x	x	-	-	-	x	x	x	-	-	-	-	-	x	-	x	-	-	x	x	-	x	x	-	-	x	x	-	-	-	x	x	x	x	-	-	x	x	x	-	-	x	-	-	-	XOR
	CB4 =	-	-	-	x	x	x	x	x	-	-	-	-	-	x	x	x	-	-	-	x	x	x	x	x	-	-	-	-	-	x	x	x	-	x	x	-	-	-	x	x	x	x	x	-	-	-	x	-	XOR
	CB5 =	x	x	x	x	x	x	x	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	x	-	x	x	x	x	-	x	-	-	-	x	-	-	-	x	XOR
	CB6 =	x	x	x	x	x	x	x	x	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	-	-	-	-	-	-	-	-	x	x	-	-	x	x	x	x	-	-	-	-	x	-	x	-	XOR
	CB7 =	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	XOR
DATA BITS		3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	6	6	6	6	6	6	6	6	6	6	7	7	7	7	7	7	7	7	7	7	
		2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	
		SLAVE #2																SLAVE #3																SLAVE #4																

Table 6. 8206 Syndrome Decoding

Syndrome Bits	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
7 6 5 4	3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 0 0 0		N	CB0	CB1	D	CB2	D	D	18	CB3	D	D	0	D	1	2	D
0 0 0 1		CB4	D	D	5	D	6	7	D	D	D	16	D	4	D	D	17
0 0 1 0		CB5	D	D	11	D	19	12	D	D	8	9	D	10	D	D	67
0 0 1 1		D	13	14	D	15	D	D	21	20	D	D	66	D	22	23	D
0 1 0 0		CB6	D	D	25	D	26	49	D	D	48	24	D	27	D	D	50
0 1 0 1		D	52	55	D	51	D	D	70	28	D	D	65	D	53	54	D
0 1 1 0		D	29	31	D	64	D	D	69	68	D	D	32	D	33	34	D
0 1 1 1		30	D	D	37	D	38	39	D	D	35	71	D	36	D	D	U
1 0 0 0		CB7	D	D	43	D	77	44	D	D	40	41	D	42	D	D	U
1 0 0 1		D	45	46	D	47	D	D	74	72	D	D	U	D	73	U	D
1 0 1 0		D	59	75	D	79	D	D	58	60	D	D	56	D	U	57	D
1 0 1 1		63	D	D	62	D	U	U	D	D	U	U	D	61	D	D	U
1 1 0 0		D	U	U	D	U	D	D	U	76	D	D	U	D	U	U	D
1 1 0 1		78	D	D	U	D	U	U	D	D	U	U	D	U	D	D	U
1 1 1 0		U	D	D	U	D	U	U	D	D	U	U	D	U	D	D	U
1 1 1 1		D	U	U	D	U	D	D	U	U	D	D	U	D	U	U	D

N = No Error
CBX = Error in Check Bit X
X = Error in Data Bit X
D = Double Bit Error
U = Uncorrectable Multi-Bit Error

SYSTEM ENVIRONMENT

The 8206 interface to a typical 32 bit memory system is illustrated in Figure 5. For larger systems, the partial parity bits from slaves two to four must be XOR'ed externally, which calls for one level of XOR gating for three 8206's and two levels for four or five 8206's.

The 8206 is designed for direct connection to the Intel 8207 Dynamic RAM Controller. The 8207 has the ability to perform dual port memory control, and Figure 6 illustrates a highly integrated dual port RAM implementation using the 8206 and 8207. The 8206/8207 combination permits such features as automatic scrubbing (correcting errors in memory during refresh), extending RAS and CAS timings for Read-Modify-Writes in single memory cycles, and automatic memory initialization upon reset. Together these two chips provide a complete dual-port, error-corrected dynamic RAM subsystem.

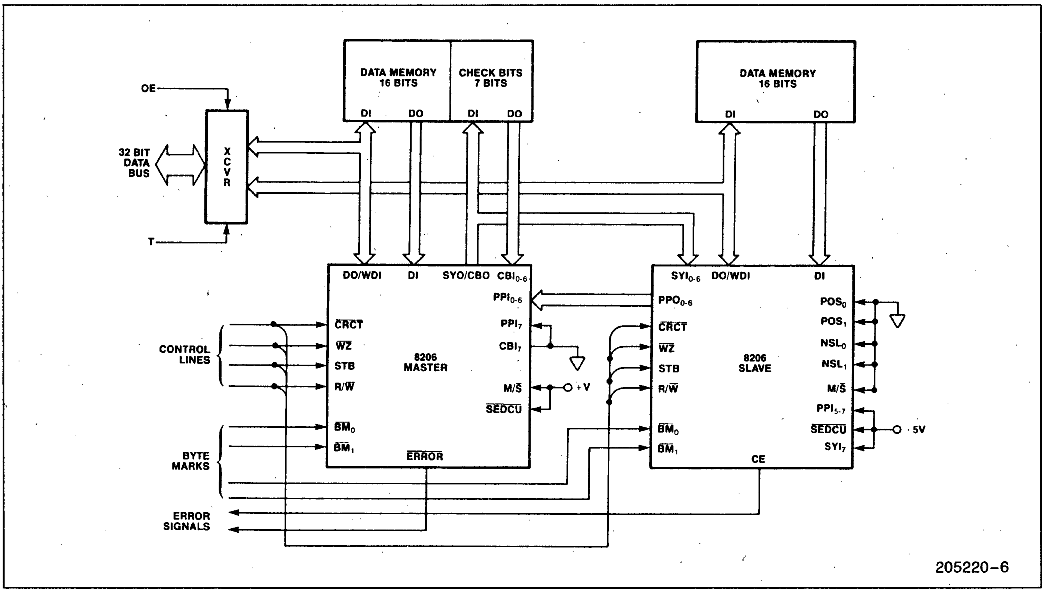


Figure 5. 32-Bit 8206 System Interface

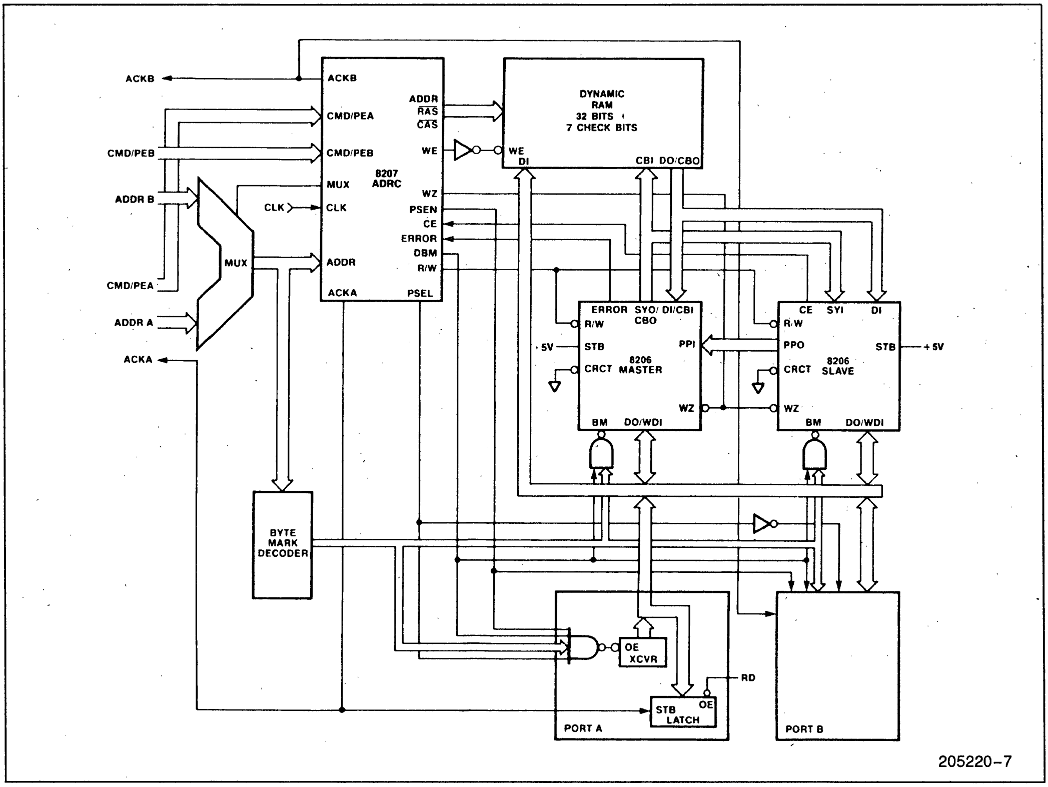


Figure 6. Dual Port RAM Subsystem with 8206/8207 (32-bit bus)

MEMORY BOARD TESTING

The 8206 lends itself to straightforward memory board testing with a minimum of hardware overhead. The following is a description of four common test modes and their implementation.

Mode 0— Read and write with error correction.
Implementation: This mode is the normal 8206 operating mode.

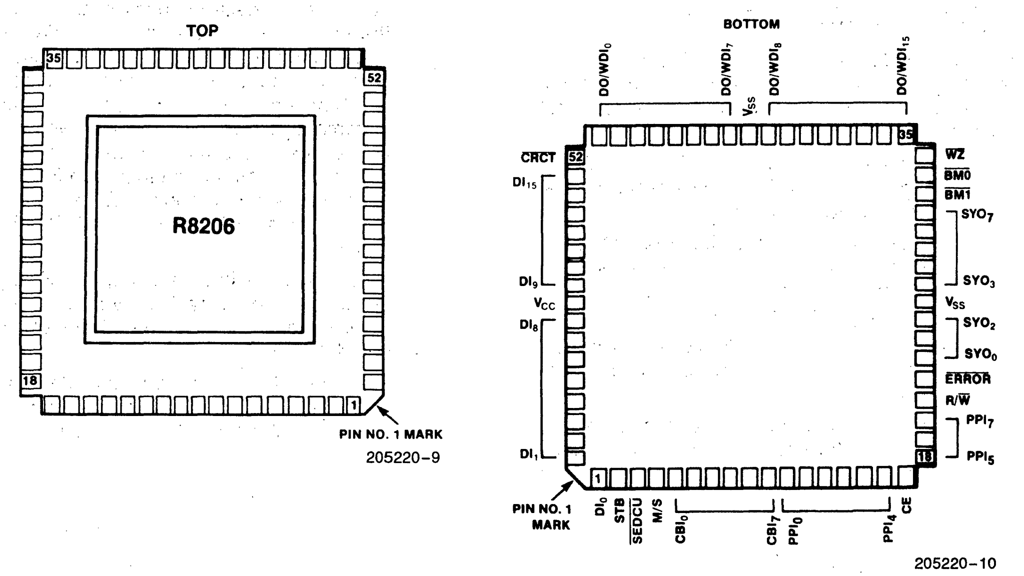
Mode 1— Read and write data with error correction disabled to allow test of data memory.
Implementation: This mode is performed with $\overline{\text{CRCT}}$ deactivated.

Mode 2— Read and write check bits with error correction disabled to allow test of check bits memory.

Implementation: Any pattern may be written into the check bits memory by judiciously choosing the proper data word to generate the desired check bits, through the use of the 8206 Hamming code. To read out the check bits it is first necessary to fill the data memory with all zeros, which may be done by activating $\overline{\text{WZ}}$ and incrementing memory addresses with $\overline{\text{WE}}$ to the check bits memory held inactive, and then performing ordinary reads. The check bits will then appear directly at the SYO outputs, with bits CB0 and CB1 inverted.

Mode 3— Write data, without altering or writing check bits, to allow the storage of bit combinations to cause error correction and detection.

Implementation: This mode is implemented by writing the desired word to memory with $\overline{\text{WE}}$ to the check bits array held inactive.



NOTE:
68 pin JEDEC TYPE A hermetic chip carrier

Figure 8a. 8206 Leadless Chip Carrier (LCC) Pinout Diagram

Top View

• 68 • 66 • 64 • 62 • 60 • 58 • 56 • 54 • 52

• 1 • 2 • 67 • 65 • 63 • 61 • 59 • 57 • 55 • 53 • 51

• 3 • 4 • 50 • 49

• 5 • 6 • 48 • 47

• 7 • 8 • 46 • 45

• 9 • 10 • 44 • 43

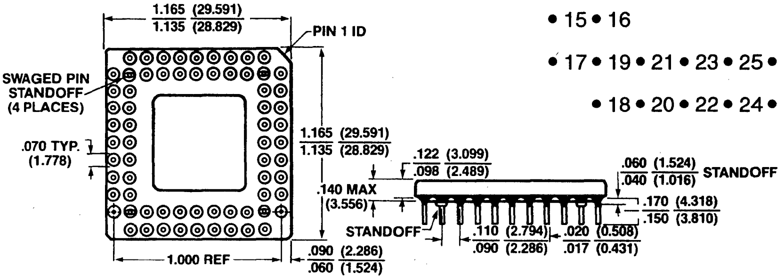
• 11 • 12 • 42 • 41

• 13 • 14 • 40 • 39

• 15 • 16 • 38 • 37

• 17 • 19 • 21 • 23 • 25 • 27 • 29 • 31 • 33 • 36 • 35

• 18 • 20 • 22 • 24 • 26 • 28 • 30 • 32 • 34



205220-13

NOTE:
68 lead ceramic pin grid array package, type A

Figure 8b. 8206 Pin Grid Array (PGA) Package and Pinout Diagram

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. 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.*

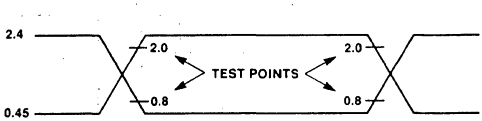
D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5.0V ± 10%, V_{SS} = GND

Symbol	Parameter	Min	Max	Units	Test Conditions
I_{CC}	Power Supply Current —Single 8206 or Slave #1 —Master in Multi-Chip or Slaves #2, 3, 4		 270 230	 mA mA	
V_{IL}(1)	Input Low Voltage	−0.5	0.8	V	
V_{IH}(1)	Input High Voltage	2.0	V_{CC} + 0.5V	V	
V_{OL}	Output Low Voltage —DO —All Others		 0.45 0.45	 V V	 I_{OL} = 8 mA I_{OL} = 2.0 mA
V_{OH}	Output High Voltage —DO, CBO —All Other Outputs	 2.6 2.4		 V V	 I_{OH} = −2 mA I_{OH} = −0.4 mA
I_{LO}	I/O Leakage Current —PPI_4/CE —DO/WDI_{0-15}		 ±20 ±10	 μA μA	0.45V ≤ $V_{I/O}$ ≤ V_{CC}
I_{LI}	Input Leakage Current —$PPI_{0-3,\ 5-7}$, CBI_{6-7}, $\overline{SEDCU}$(2) —All Other Input Only Pins		 ±20 ±10	 μA μA	0V ≤ V_{IN} ≤ V_{CC}

NOTES:

1. $\overline{SEDCU}$ (pin 3) and M/$\overline{S}$ (pin 4) are device strapping options and should be tied to V_{CC} or GND. V_{IH} min = V_{CC} −0.5V and V_{IL} max = 0.5V.
2. PPI_{0-7} (pins 13–20) and CBI_{6-7} (pins 11, 12) have internal pull-up resistors and if left unconnected will be pulled to V_{CC}.

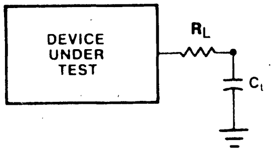
A.C. TESTING INPUT, OUTPUT WAVEFORM



205220–14

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



205220–15

C_L Includes Jig Capacitance

A.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ± 10%, V_{SS} = 0V, R_L = 22Ω, C_L = 50 pF; all times are in ns

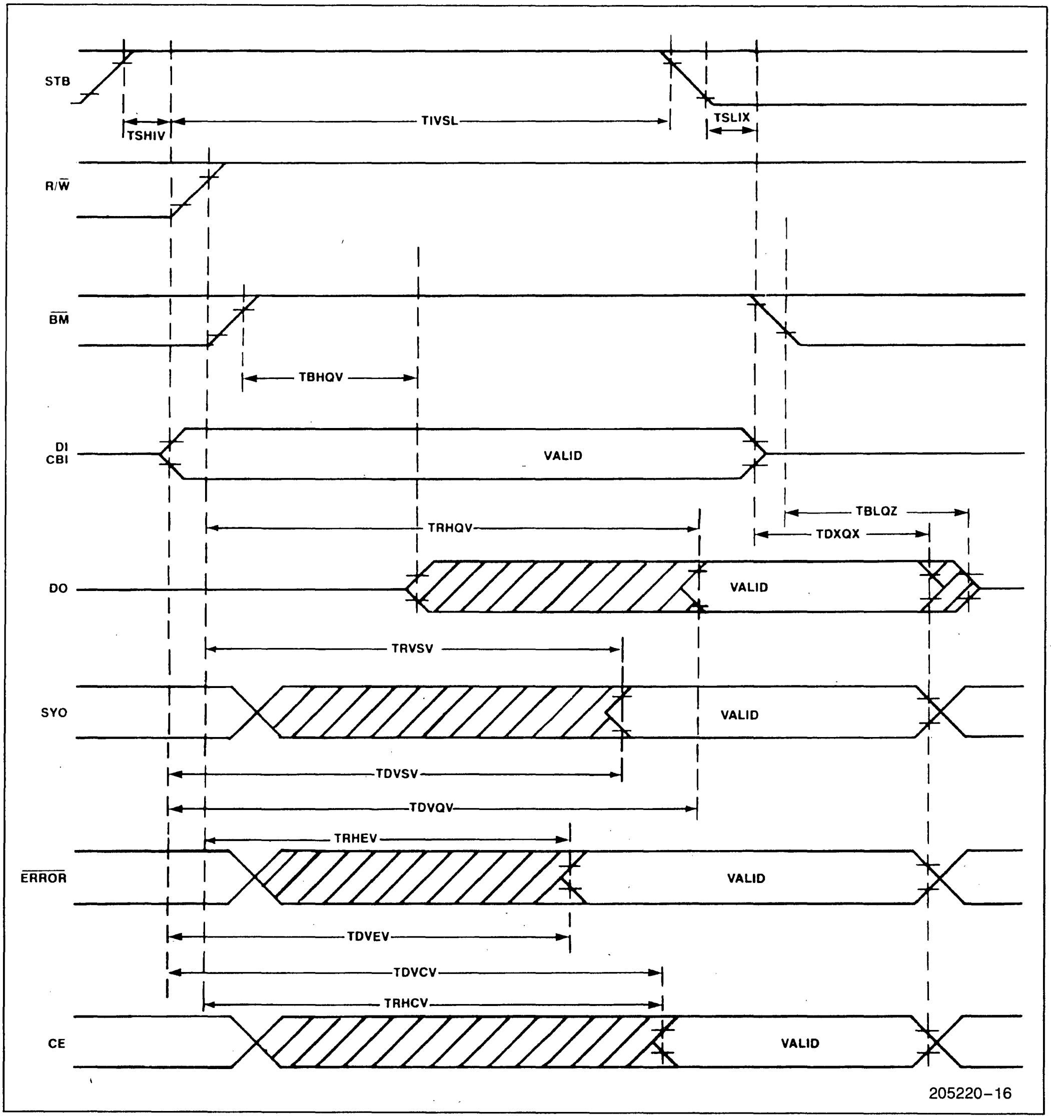
Symbol	Parameter	8206-1		8206		Notes
		Min	Max	Min	Max	
T_{RHEV}	$\overline{\text{ERROR}}$ Valid from R/$\overline{\text{W}}$ ↑		20		25	
T_{RHCV}	CE Valid from R/$\overline{\text{W}}$ ↑ (Single 8206)		34		44	
T_{RHQV}	Corrected Data Valid from R$\overline{\text{W}}$ ↑		44		54	1
T_{RVSV}	SYO/CBO/PPO Valid from R/$\overline{\text{W}}$		32		42	1
T_{DVEV}	$\overline{\text{ERROR}}$ Valid from Data/Check Bits In		35		42	
T_{DVCV}	CE Valid from Data/Check Bits In		50		70	
T_{DVQV}	Corrected Data Valid from Data/Check Bits In		55		67	
T_{DVSV}	SYO/PPO Valid from Data/Check Bits In		40		55	
T_{BHQV}	Corrected Data Access Time		35		37	
T_{BXQX}	Hold Time from Data/Check Bits In	0		0		1
T_{BLQZ}	Corrected Data Float Delay	0	25	0	28	1
T_{SHIV}	STB High to Data/Check Bits In Valid	30		30		2
T_{IVSL}	Data/Check Bits In to STB ↓ Set-Up	5		5		
T_{SLIX}	Data/Check Bits In from STB ↓ Hold	15		25		
T_{PVEV}	$\overline{\text{ERROR}}$ Valid from Partial Parity In		21		30	3
T_{PVQV}	Corrected Data (Master) from Partial Parity In		46		61	1, 3
T_{PVSV}	Syndrome/Check Bits Out from Partial Parity In		32		43	1, 3
T_{SVQV}	Corrected Data (Slave) Valid from Syndrome		41		51	3
T_{SVCV}	CE Valid from Syndrome (Slave Number 1)		43		48	3
T_{QVQV}	Check Bits/Partial Parity Out from Write Data In		44		64	1
T_{RHSX}	Check Bits/Partial Parity Out from R/W, $\overline{\text{WZ}}$ Hold	0		0		1
T_{RLSX}	Syndrome Out from R/$\overline{\text{W}}$ Hold	0		0		
T_{QXQX}	Hold Time from Write Data In	0		0		1
T_{SVRL}	Syndrome Out to R/$\overline{\text{W}}$ ↓ Set-Up	5		17		3
T_{DVRL}	Data/Check Bits to R/$\overline{\text{W}}$ Set-Up	24		39		1
T_{DVQU}	Uncorrected Data Out from Data In		29		32	
T_{TVQV}	Corrected Data Out from $\overline{\text{CRCT}}$ ↓		25		30	
T_{WLQL}	W$\overline{\text{Z}}$ ↓ to Zero Out		25		30	
T_{WHQX}	Zero Out from $\overline{\text{WZ}}$ ↑ Hold	0		0		0

NOTES:

1. A.C. Test Levels for CBO and DO are 2.4V and 0.8V.
2. T_{SHIV} is required to guarantee output delay timings: T_{DVEV}, T_{DVCV}, T_{DVQV}, T_{DVSV}. T_{SHIV} + T_{IVSL} guarantees a min STB pulse width of 35 ns.
3. Not required for 8/16 bit systems.

WAVEFORMS

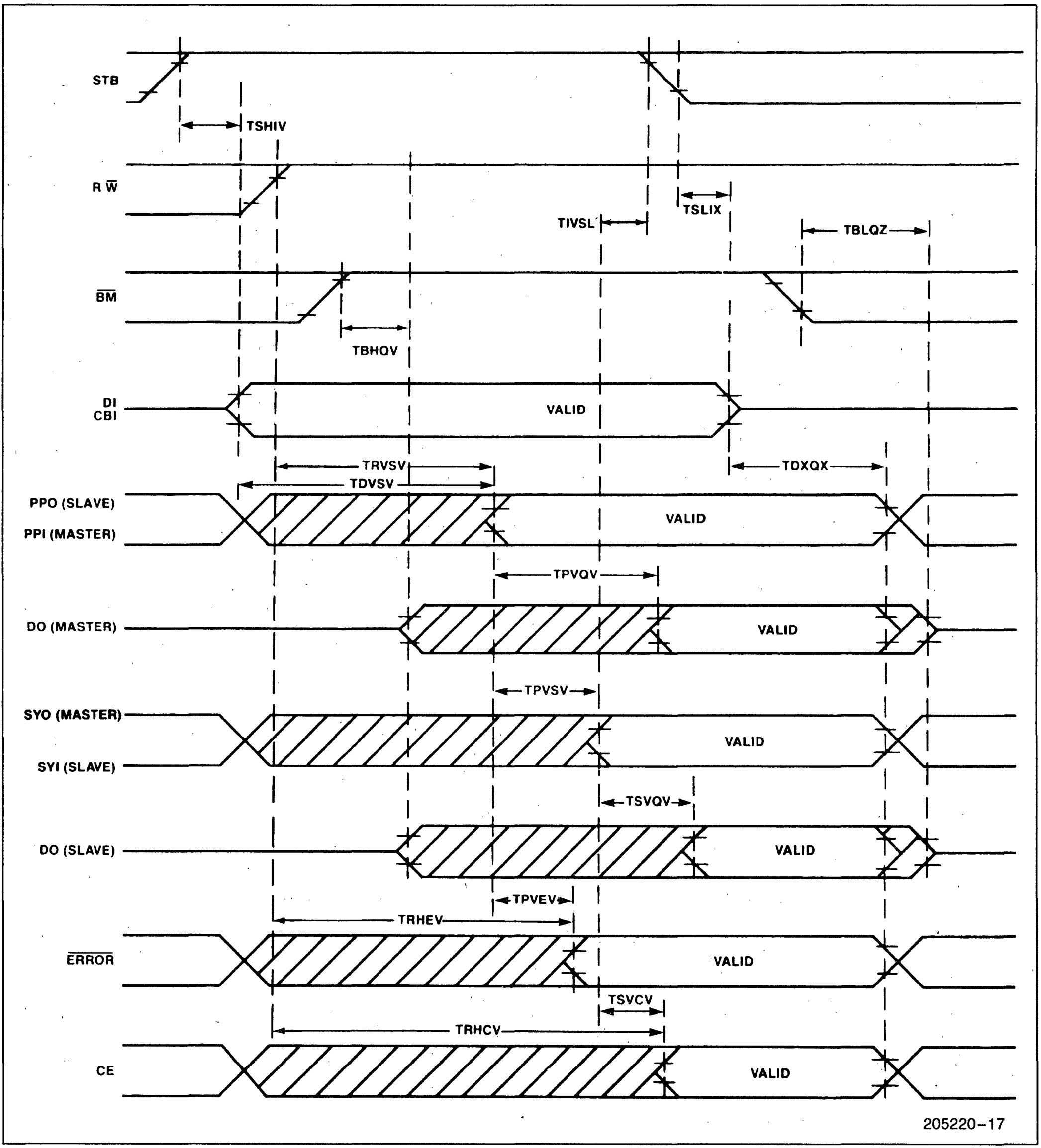
READ



205220-16

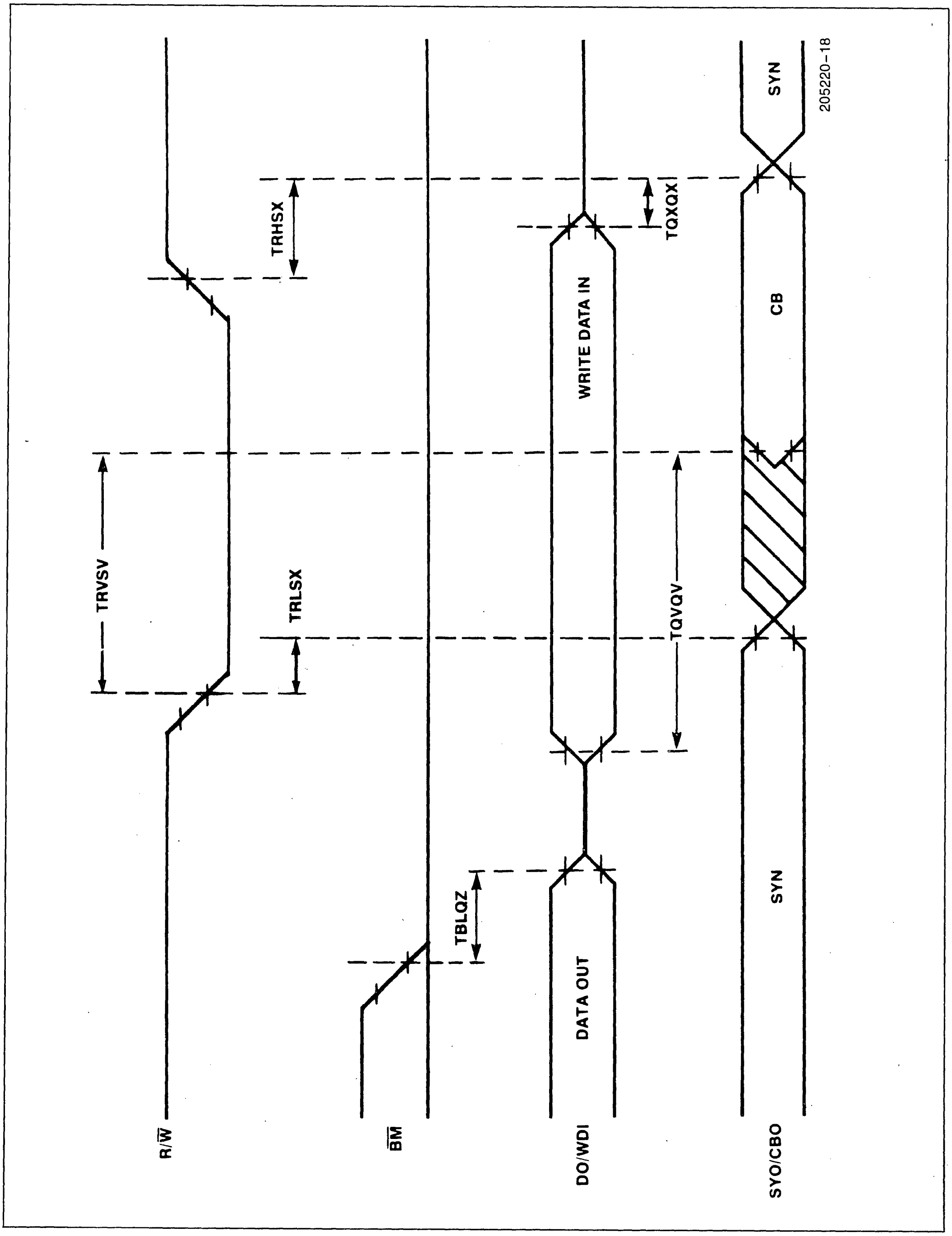
WAVEFORMS (Continued)

READ—MASTER/SLAVE



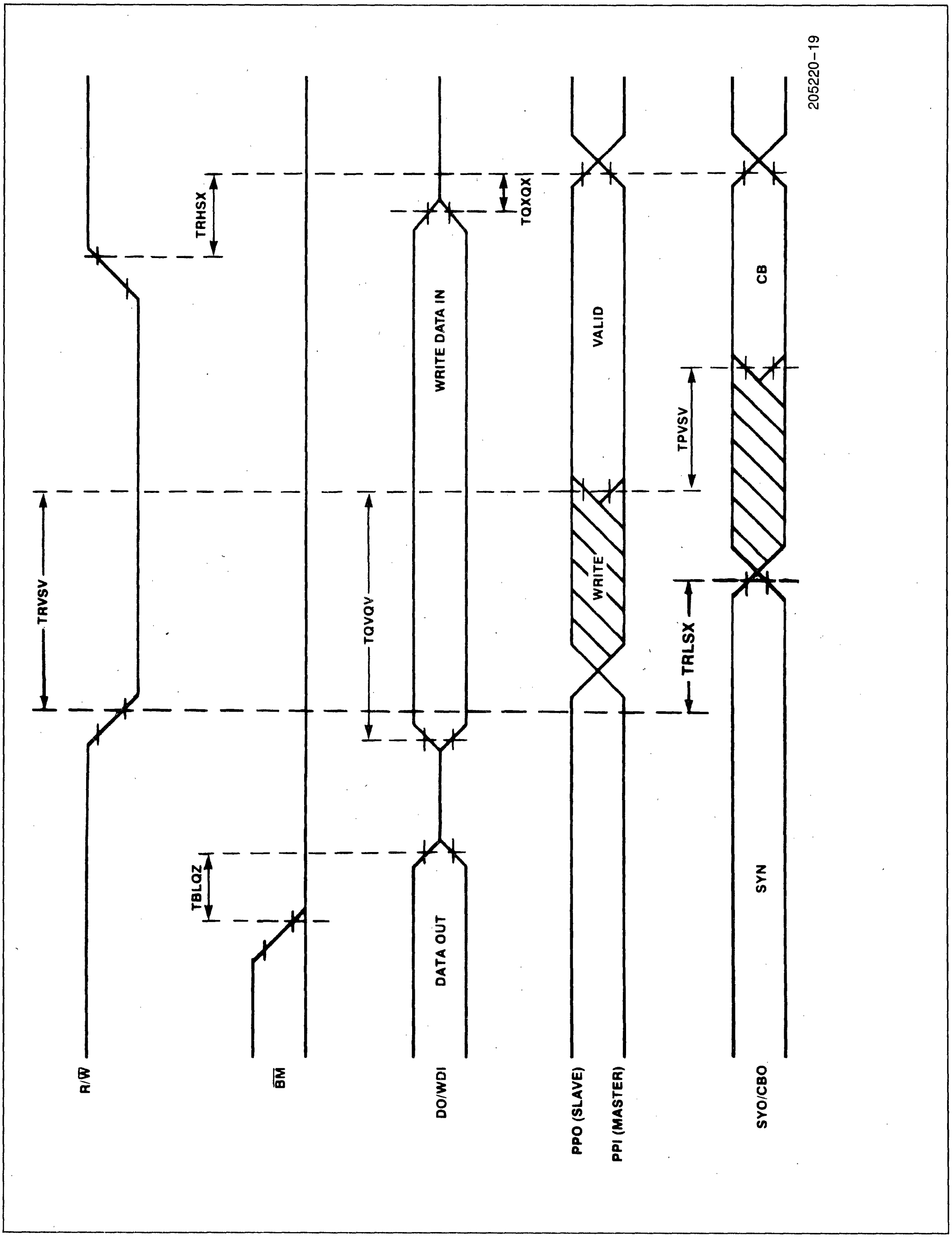
WAVEFORMS (Continued)

FULL WRITE



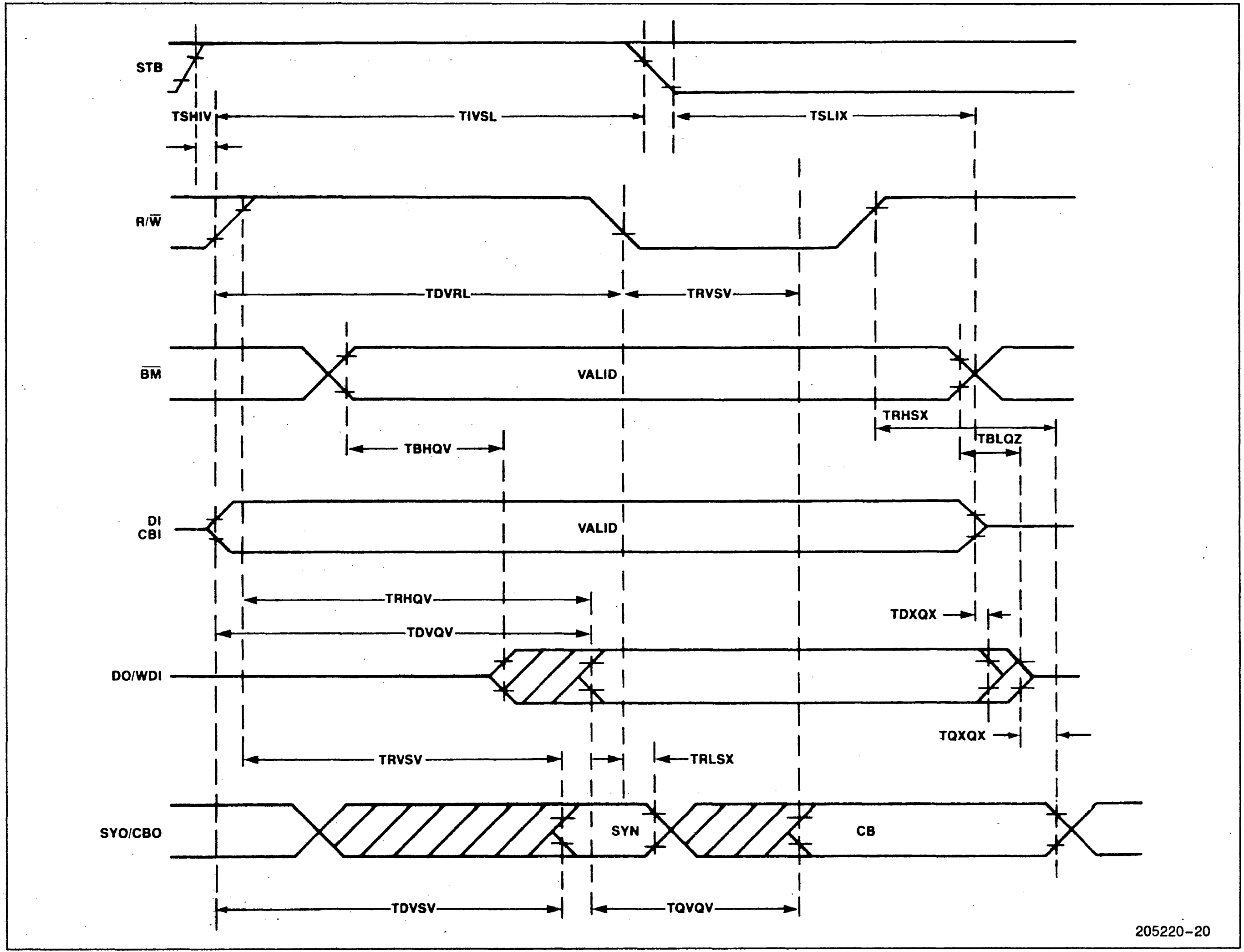
WAVEFORMS (Continued)

FULL WRITE—MASTER/SLAVE



WAVEFORMS (Continued)

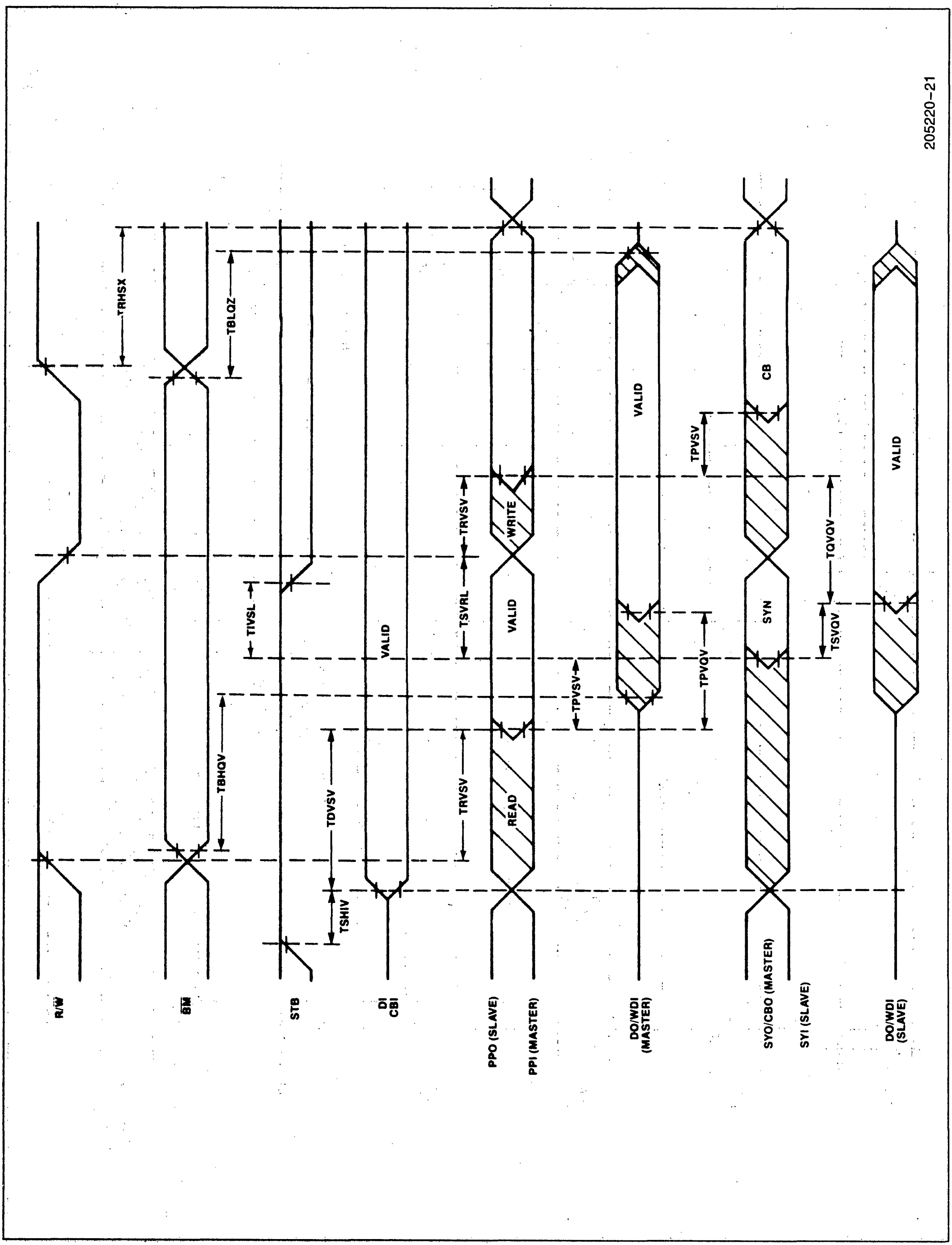
READ MODIFY WRITE



205220-20

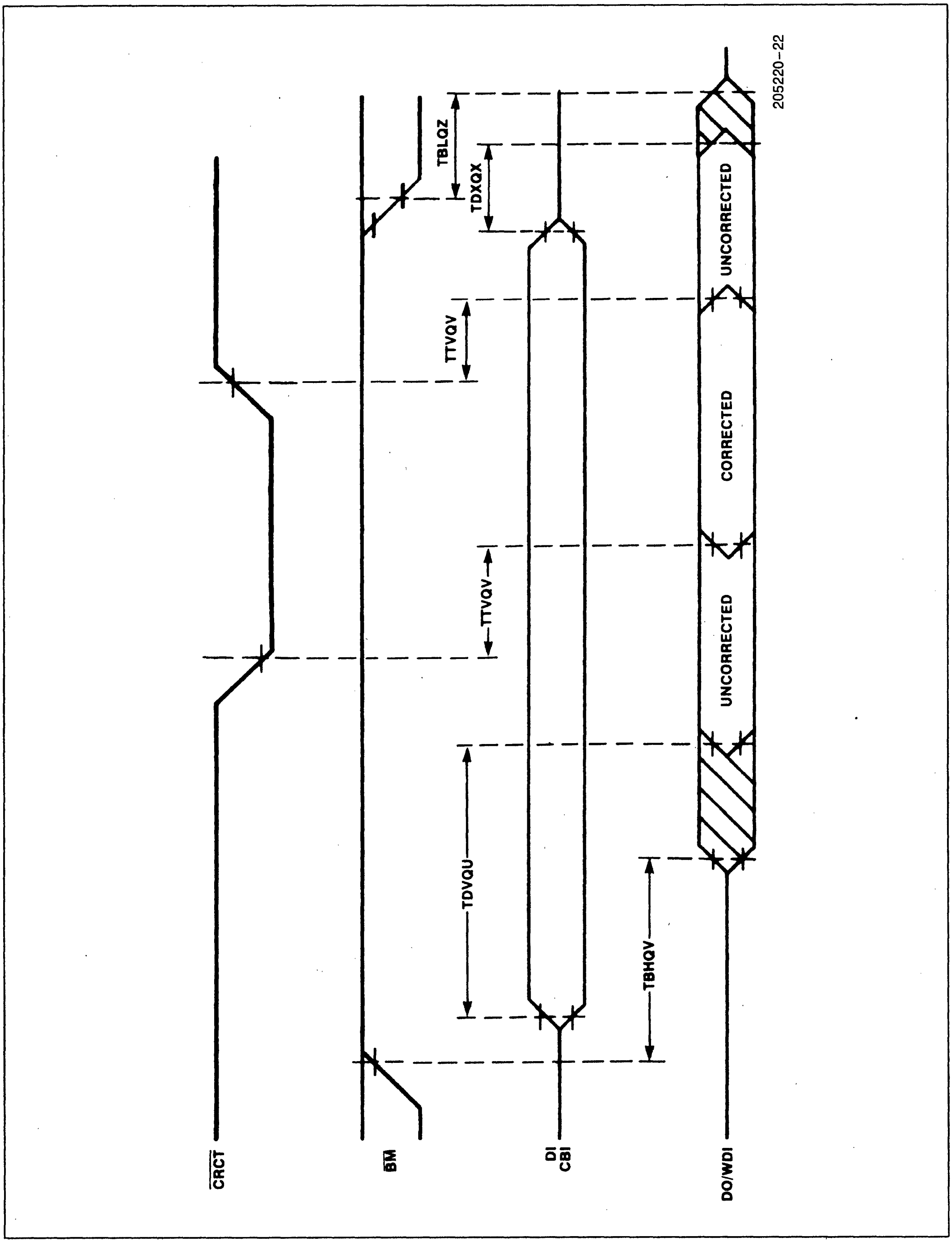
WAVEFORMS (Continued)

READ MODIFY WRITE—MASTER/SLAVE



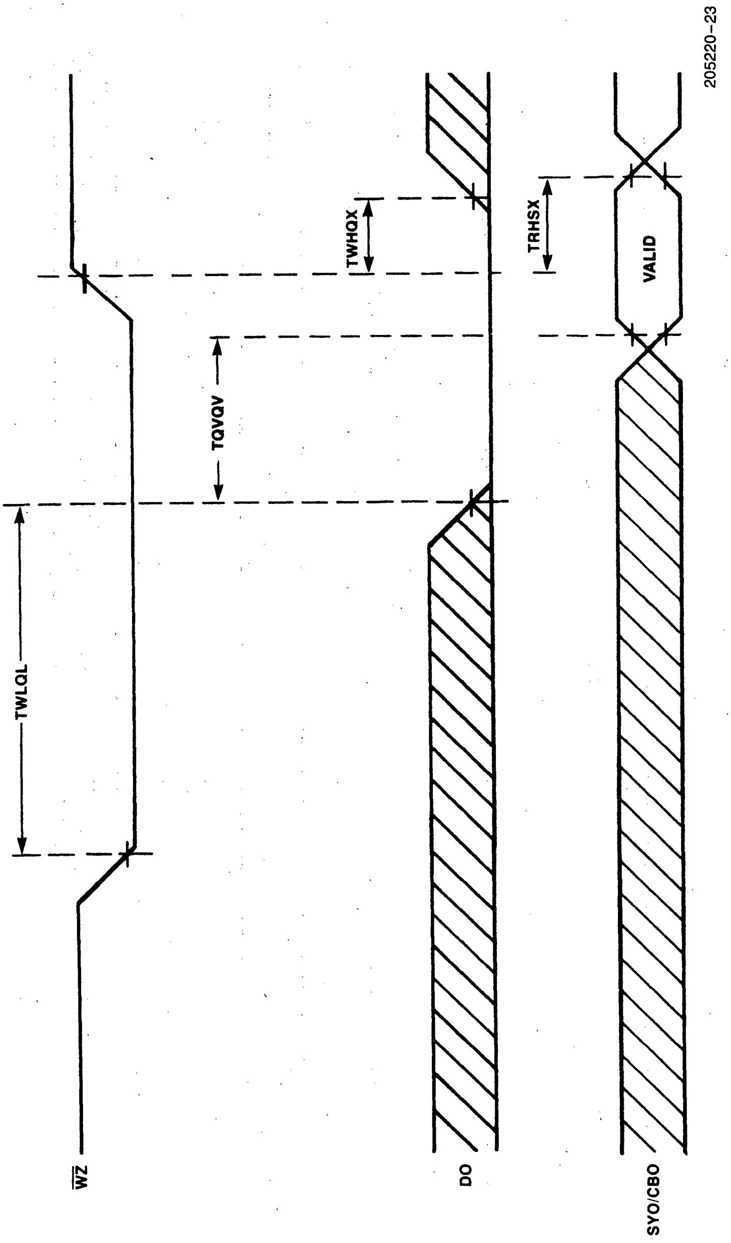
WAVEFORMS (Continued)

NON-CORRECTING READ



WAVEFORMS (Continued)

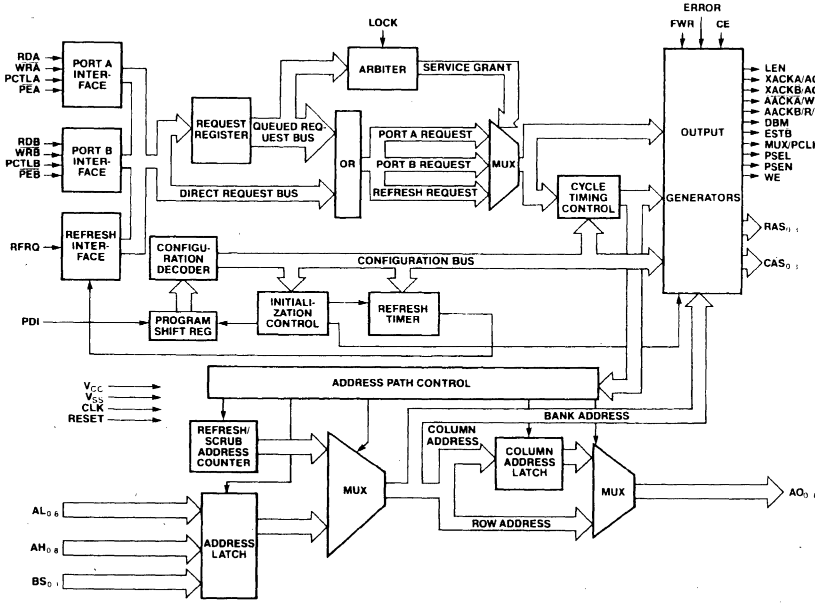
WRITE ZERO



8207
DUAL-PORT DYNAMIC RAM CONTROLLER

- **Provides All Signals Necessary to Control 16K, 64K and 256K Dynamic RAMs**
- **Directly Addresses and Drives up to 2 Megabytes without External Drivers**
- **Supports Single and Dual-Port Configurations**
- **Automatic RAM Initialization in All Modes**
- **Four Programmable Refresh Modes**
- **Transparent Memory Scrubbing in ECC Mode**
- **Fast Cycle Support for 8 MHz 80286 with 8207-16**
- **Slow Cycle Support for 8 MHz, 10 MHz 8086/88, 80186/188 with 8207-8, 8207-10**
- **Provides Signals to Directly Control the 8206 Error Detection and Correction Unit**
- **Supports Synchronous or Asynchronous Operation on Either Port**
- **68 Lead JEDEC Type A Leadless Chip Carrier (LCC) and Pin Grid Array (PGA), Both in Ceramic.**

The Intel 8207 Dual-Port Dynamic RAM Controller is a high-performance, systems-oriented, Dynamic RAM controller that is designed to easily interface 16K, 64K and 256K Dynamic RAMs to Intel and other microprocessor systems. A dual-port interface allows two different busses to independently access memory. When configured with an 8206 Error Detection and Correction Unit the 8207 supplies the necessary logic for designing large error-corrected memory arrays. This combination provides automatic memory initialization and transparent memory error scrubbing.



210463–1

Figure 1. 8207 Block Diagram

Table 1. Pin Description

Symbol	Pin	Type	Name and Function
LEN	1	O	**ADDRESS LATCH ENABLE:** In two-port configurations, when Port A is running with iAPX 286 Status interface mode, this output replaces the ALE signal from the system bus controller of port A and generates an address latch enable signal which provides optimum setup and hold timing for the 8207. This signal is used in Fast Cycle operation only.
$\overline{\text{XACKA}}$/ $\overline{\text{ACKA}}$	2	O	**TRANSFER ACKNOWLEDGE PORT A/ACKNOWLEDGE PORT A:** In non-ECC mode, this pin is $\overline{\text{XACKA}}$ and indicates that data on the bus is valid during a read cycle or that data may be removed from the bus during a write cycle for Port A. $\overline{\text{XACKA}}$ is a Multibus-compatible signal. In ECC mode, this pin is $\overline{\text{ACKA}}$ which can be configured, depending on the programming of the X program bit, as an $\overline{\text{XACK}}$ or $\overline{\text{AACK}}$ strobe. The SA programming bit determines whether the $\overline{\text{AACK}}$ will be an early $\overline{\text{EAACKA}}$ or a late $\overline{\text{LAACKA}}$ interface signal.
$\overline{\text{XACKB}}$/ $\overline{\text{ACKB}}$	3	O	**TRANSFER ACKNOWLEDGE PORT B/ACKNOWLEDGE PORT B:** In non-ECC mode, this pin is XACKB and indicates that data on the bus is valid during a read cycle or that data may be removed from the bus during a write cycle for Port B. $\overline{\text{XACKB}}$ is a Multibus-compatible signal. In ECC mode, this pin is $\overline{\text{ACKB}}$ which can be configured, depending on the programming of the X program bit, as an $\overline{\text{XACK}}$ or $\overline{\text{AACK}}$ strobe. The SB programming bit determines whether the $\overline{\text{AACK}}$ will be an early $\overline{\text{EAACKB}}$ or a late $\overline{\text{LAACKB}}$ interface signal.
$\overline{\text{AACKA}}$/ $\overline{\text{WZ}}$	4	O	**ADVANCED ACKNOWLEDGE PORT A/WRITE ZERO:** In non-ECC mode, this pin is $\overline{\text{AACKA}}$ and indicates that the processor may continue processing and that data will be available when required. This signal is optimized for the system by programming the SA program bit for synchronous or asynchronous operation. In ECC mode, after a RESET, this signal will cause the 8206 to force the data to all zeros and generate the appropriate check bits.
$\overline{\text{AACKB}}$/ R/$\overline{\text{W}}$	5	O	**ADVANCED ACKNOWLEDGE PORT B/READ/WRITE:** In non-ECC mode, this pin is $\overline{\text{AACKB}}$ and indicates that the processor may continue processing and that data will be available when required. This signal is optimized for the system by programming the SB program bit for synchronous or asynchronous operation. In ECC mode, this signal causes the 8206 EDCU to latch the syndrome and error flags and generate check bits.
$\overline{\text{DBM}}$	6	O	**DISABLE BYTE MARKS:** This is an ECC control output signal indicating that a read or refresh cycle is occurring. This output forces the byte address decoding logic to enable all 8206 data output buffers. In ECC mode, this output is also asserted during memory initialization and the 8-cycle dynamic RAM wake-up exercise. In non-ECC systems this signal indicates that either a read, refresh or 8-cycle warm-up is in progress.
$\overline{\text{ESTB}}$	7	O	**ERROR STROBE:** In ECC mode, this strobe is activated when an error is detected and allows a negative-edge triggered flip-flop to latch the status of the 8206 EDCU CE for systems with error logging capabilities. $\overline{\text{ESTB}}$ will not be issued during refresh cycles.
LOCK	8	I	**LOCK:** This input instructs the 8207 to lock out the port not being serviced at the time LOCK was issued.
V_{CC}	9 43	I	**DRIVER POWER:** +5 volts. Supplies V_{CC} for the output drivers. **LOGIC POWER:** +5 volts. Supplies V_{CC} for the internal logic circuits.
CE	10	I	**CORRECTABLE ERROR:** This is an ECC input from the 8206 EDCU which instructs the 8207 whether a detected error is correctable or not. A high input indicates a correctable error. A low input inhibits the 8207 from activating WE to write the data back into RAM. This should be connected to the CE output of the 8206.

Table 1. Pin Description (Continued)

Symbol	Pin	Type	Name and Function
$\overline{\text{ERROR}}$	11	I	**ERROR:** This is an ECC input from the 8206 EDCU and instructs the 8207 that an error was detected. This pin should be connected to the ERROR output of the 8206.
MUX/ PCLK	12	O	**MULTIPLEXER CONTROL/PROGRAMMING CLOCK:** Immediately after a RESET this pin is used to clock serial programming data into the PDI pin. In normal two-port operation, this pin is used to select memory addresses from the appropriate port. When this signal is high, port A is selected and when it is low, port B is selected. This signal may change state before the completion of a RAM cycle, but the RAM address hold time is satisfied.
PSEL	13	O	**PORT SELECT:** This signal is used to select the appropriate port for data transfer. When this signal is high port A is selected and when it is low port B is selected.
PSEN	14	O	**PORT SELECT ENABLE:** This signal used in conjunction with PSEL provides contention-free port exchange on the data bus. When PSEN is low, port selection is allowed to change state.
WE	15	O	**WRITE ENABLE:** This signal provides the dynamic RAM array the write enable input for a write operation.
$\overline{\text{FWR}}$	16	I	**FULL WRITE:** This is an ECC input signal that instructs the 8207, in an ECC configuration, whether the present write cycle is normal RAM write (full write) or a RAM partial write (read-modify-write) cycle.
RESET	17	I	**RESET:** This signal causes all internal counters and state flip-flops to be reset and upon release of RESET, data appearing at the PDI pin is clocked in by the PCLK output. The states of the PDI, PCTLA, PCTLB and RFRQ pins are sampled by RESET going inactive and are used to program the 8207. An 8-cycle dynamic RAM warm-up is performed after clocking PDI bits into the 8207.
$\overline{\text{CAS0}}$-$\overline{\text{CAS3}}$	18–21	O	**COLUMN ADDRESS STROBE:** These outputs are used by the dynamic RAM array to latch the column address, present on the AO0–8 pins. These outputs are selected by the BS0 and BS1 as programmed by program bits RB0 and RB1. These outputs drive the dynamic RAM array directly and need no external drivers.
$\overline{\text{RAS0}}$-$\overline{\text{RAS3}}$	22–25	O	**ROW ADDRESS STROBE:** These outputs are used by the dynamic RAM array to latch the row address, present on the AO0–8 pins. These outputs are selected by the BS0 and BS1 as programmed by program bits RB0 and RB1. These outputs drive the dynamic RAM array directly and need no external drivers.
V_{SS}	26 60	I I	**DRIVER GROUND:** Provides a ground for the output drivers. **LOGIC GROUND:** Provides a ground for the remainder of the device.
AO0–AO8	35–27	O	**ADDRESS OUTPUTS:** These outputs are designed to provide the row and column addresses of the selected port to the dynamic RAM array. These outputs drive the dynamic RAM array directly and need no external drivers.
BS0–BS1	36–37	I	**BANK SELECT:** These inputs are used to select one of four banks of the dynamic RAM array as defined by the program bits RB0 and RB1.
AL0–AL8	38–42 44–47	I	**ADDRESS LOW:** These lower-order address inputs are used to generate the row address for the internal address multiplexer.
AH0–AH8	48–56	I	**ADDRESS HIGH:** These higher-order address inputs are used to generate the column address for the internal address multiplexer.

Table 1. Pin Description (Continued)

Symbol	Pin	Type	Name and Function
PDI	57	I	**PROGRAM DATA INPUT:** This input programs the various user-selectable options in the 8207. The PCLK pin shifts programming data into the PDI input from optional external shift registers. This pin may be strapped high or low to a default ECC (PDI = Logic "1") or non-ECC (PDI = Logic "0") mode configuration.
RFRQ	58	I	**REFRESH REQUEST:** This input is sampled on the falling edge of RESET. If it is high at RESET, then the 8207 is programmed for internal refresh request or external refresh request with failsafe protection. If it is low at RESET, then the 8207 is programmed for external refresh without failsafe protection or burst refresh. Once programmed the RFRQ pin accepts signals to start an external refresh with failsafe protection or external refresh without failsafe protection or a burst refresh.
CLK	59	I	**CLOCK:** This input provides the basic timing for sequencing the internal logic.
$\overline{\text{RDB}}$	61	I	**READ FOR PORT B:** This pin is the read memory request command input for port B. This input also directly accepts the $\overline{\text{S1}}$ status line from Intel processors.
$\overline{\text{WRB}}$	62	I	**WRITE FOR PORT B:** This pin is the write memory request command input for port B. This input also directly accepts the $\overline{\text{S0}}$ status line from Intel processors.
$\overline{\text{PEB}}$	63	I	**PORT ENABLE FOR PORT B:** This pin serves to enable a RAM cycle request for port B. It is generally decoded from the port address.
PCTLB	64	I	**PORT CONTROL FOR PORT B:** This pin is sampled on the falling edge of RESET. If low after RESET, the 8207 is programmed to accept memory read and write commands, Multibus commands or iAPX 286 status inputs. If high after RESET, the 8207 is programmed to accept status inputs from iAPX 86 or iAPX 186 processors. The $\overline{\text{S2}}$ status line should be connected to this input if programmed to accept iAPX 86 or iAPX 186 status inputs. When programmed to accept commands or iAPX 286 status, it should be tied low or it may be used as a Multibus-compatible inhibit signal.
$\overline{\text{RDA}}$	65	I	**READ FOR PORT A:** This pin is the read memory request command input for port A. This input also directly accepts the $\overline{\text{S1}}$ status line from Intel processors.
$\overline{\text{WRA}}$	66	I	**WRITE FOR PORT A:** This pin is the write memory request command input for port A. This input also directly accepts the $\overline{\text{S0}}$ status line from Intel processors.
$\overline{\text{PEA}}$	67	I	**PORT ENABLE FOR PORT A:** This pin serves to enable a RAM cycle request for port A. It is generally decoded from the port address.
PCTLA	68	I	**PORT CONTROL FOR PORT A:** This pin is sampled on the falling edge of RESET. If low after RESET, the 8207 is programmed to accept memory read and write commands, Multibus commands or iAPX 286 status inputs. If high after RESET, the 8207 is programmed to accept status inputs from iAPX 86 or iAPX 186 processors. The $\overline{\text{S2}}$ status line should be connected to this input if programmed to accept iAPX 86 or iAPX 186 status inputs. When programmed to accept commands or iAPX 286 status, it should be tied low or it may be connected to INHIBIT when operating with Multibus.

GENERAL DESCRIPTION

The Intel 8207 Dual-Port Dynamic RAM Controller is a microcomputer peripheral device which provides the necessary signals to address, refresh and directly drive 16K, 64K and 256K dynamic RAMs. This controller also provides the necessary arbitration circuitry to support dual-port access of the dynamic RAM array.

The 8207 supports several microprocessor interface options including synchronous and asynchronous connection to iAPX 86, iAPX 88, iAPX 186, iAPX 188, iAPX 286 and Multibus.

This device may be used with the 8206 Error Detection and Correction Unit (EDCU). When used with the 8206, the 8207 is programmed in the Error Checking and Correction (ECC) mode. In this mode, the 8207 provides all the necessary control signals for the 8206 to perform memory initialization and transparent error scrubbing during refresh.

FUNCTIONAL DESCRIPTION

Processor Interface

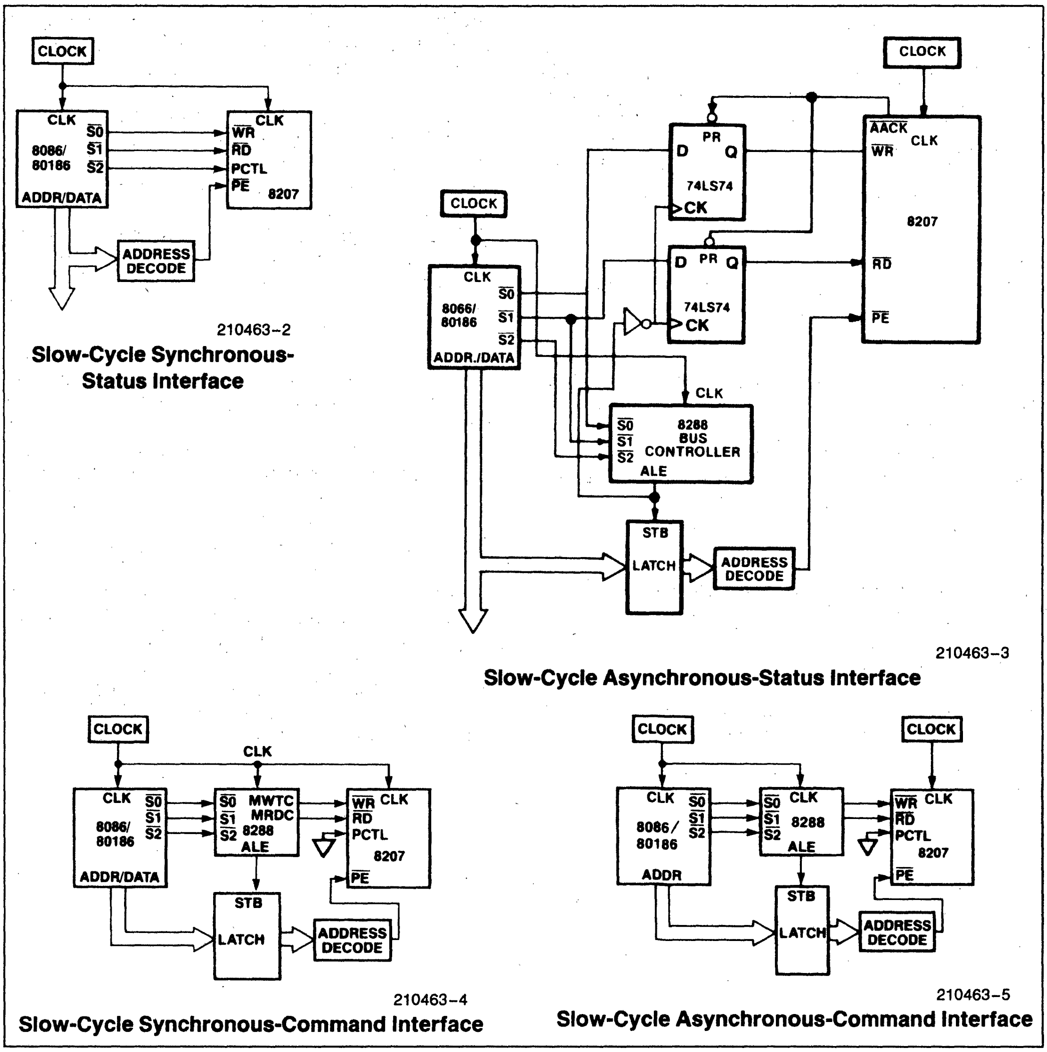
The 8207 has control circuitry for two ports each capable of supporting one of several possible bus structures. The ports are independently configurable allowing the dynamic RAM to serve as an interface between two different bus structures.

Each port of the 8207 may be programmed to run synchronous or asynchronous to the processor clock. (See Synchronous/Asynchronous Mode.) The 8207 has been optimized to run synchronously with Intel's iAPX 86, iAPX 88, iAPX 186, iAPX 188, and iAPX 286. When the 8207 is programmed to run in asynchronous mode, the 8207 inserts the necessary synchronization circuitry for the $\overline{RD}$, $\overline{WR}$, $\overline{PE}$, and PCTL inputs.

The 8207 achieves high performance (i.e., no wait states) by decoding the status lines directly from the iAPX 86, iAPX 88, iAPX 186, iAPX 188 and iAPX 286 processors. The 8207 can also be programmed to receive read or write Multibus commands or commands from a bus controller. (See Status/Command Mode.)

The 8207 may be programmed to accept the clock of the iAPX 86, 88, 186, 188 or 286. The 8207 adjusts its internal timing to allow for the different clock frequencies of these microprocessors. (See Microprocessor Clock Frequency Option.)

Figures 2A and 2B show the different processor interfaces to the 8207 using the synchronous or asynchronous mode and status or command interface.



210463-2
Slow-Cycle Synchronous-Status Interface

210463-3
Slow-Cycle Asynchronous-Status Interface

210463-4
Slow-Cycle Synchronous-Command Interface

210463-5
Slow-Cycle Asynchronous-Command Interface

Figure 2A. Slow-Cycle (CFS = 0) Port Interfaces Supported by the 8207

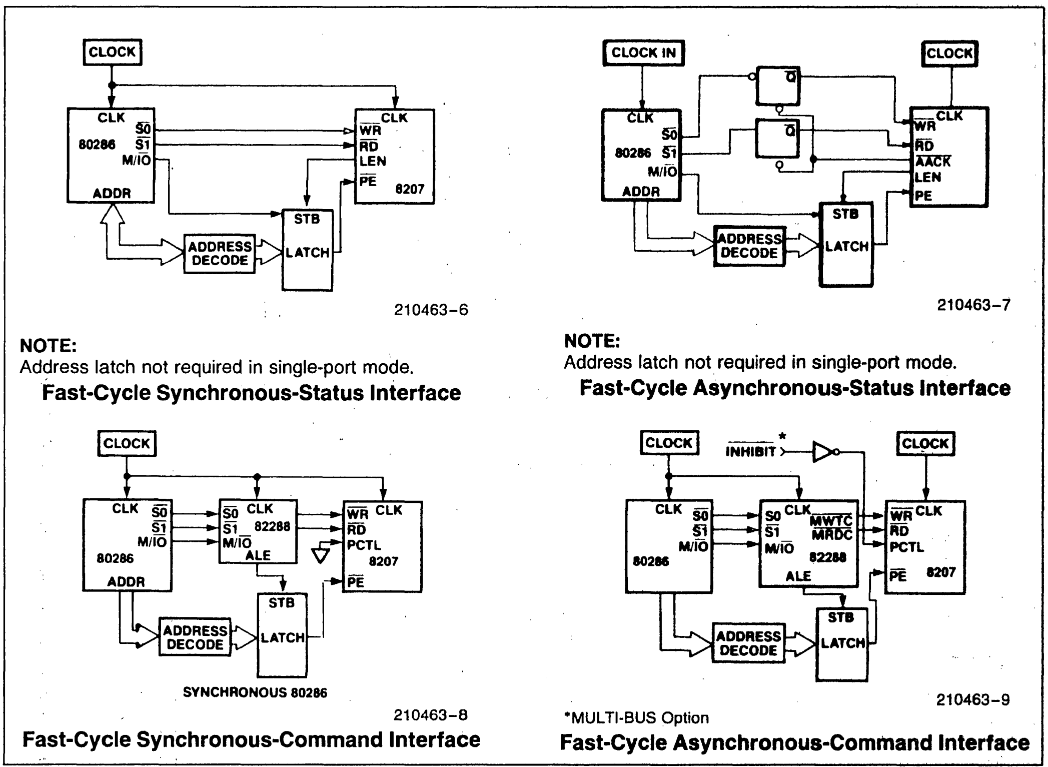
Single-Port Operation

The use of an address latch with the iAPX 286 status interface is not needed since the 8207 can internally latch the addresses with an internal signal similar in behavior to the LEN output. This operation is active only in single-port applications when the processor is interfaced to port A.

Dual-Port Operation

The 8207 provides for two-port operation. Two independent processors may access memory controlled by the 8207. The 8207 arbitrates between each of the processor requests and directs data to or from the appropriate port. Selection is done on a priority concept that reassigns priorities based upon past history. Processor requests are internally queued.

Figure 3 shows a dual-port configuration with two iAPX 86 systems interfacing to dynamic RAM. One of the processor systems is interfaced synchronously using the status interface and the other is interfaced asynchronously also using the status interface.



210463-6

NOTE:
Address latch not required in single-port mode.

Fast-Cycle Synchronous-Status Interface

210463-7

NOTE:
Address latch not required in single-port mode.

Fast-Cycle Asynchronous-Status Interface

210463-8

Fast-Cycle Synchronous-Command Interface

210463-9

*MULTI-BUS Option

Fast-Cycle Asynchronous-Command Interface

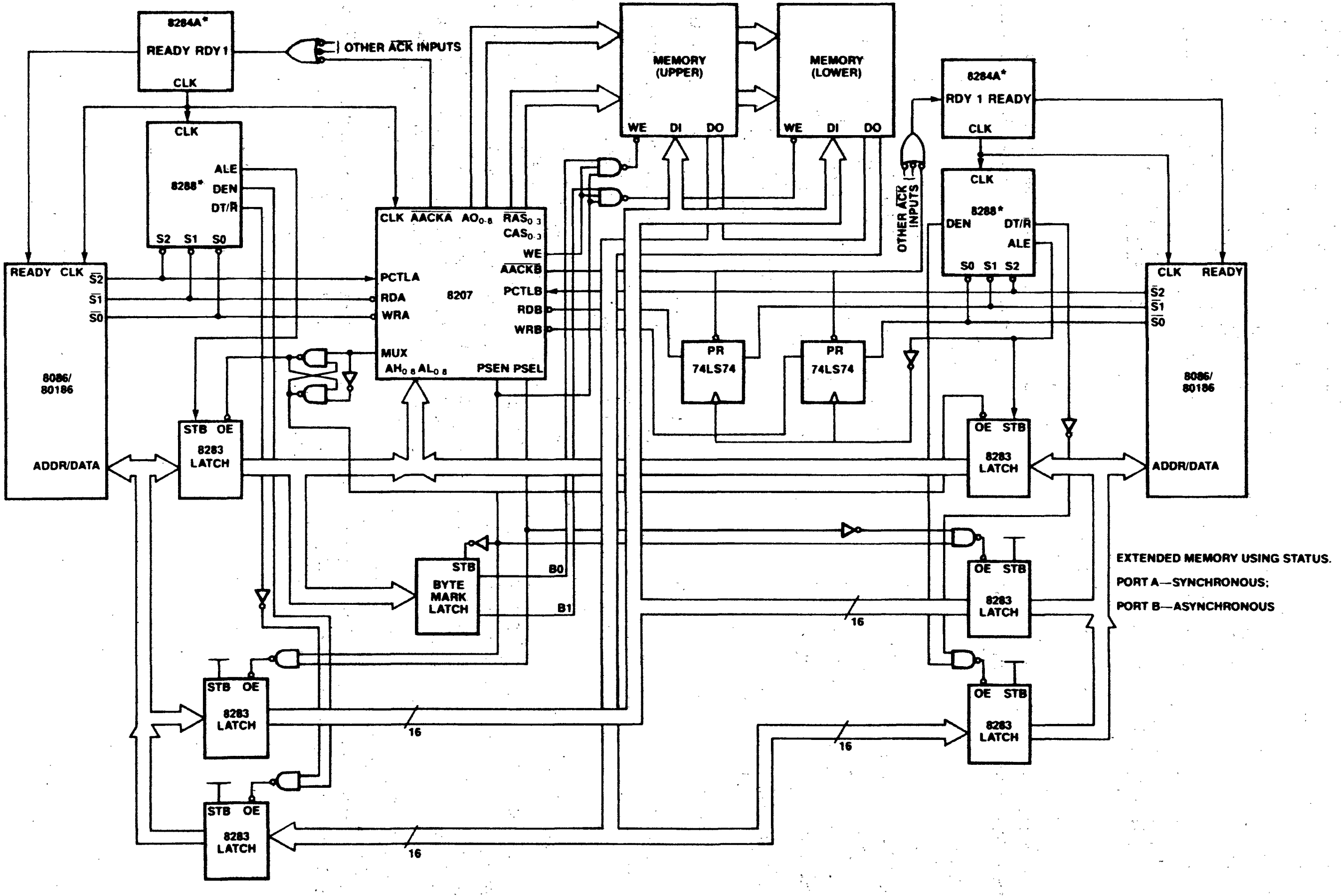
Figure 2B. Fast-Cycle (CFS = 1) Port Interfaces Supported by the 8207

Dynamic RAM Interface

The 8207 is capable of addressing 16K, 64K and 256K dynamic RAMs. Figure 4 shows the connection of the processor address bus to the 8207 using the different RAMs.

The 8207 divides memory into as many as four banks, each bank having its own Row (RAS) and Column (CAS) Address Strobe pair. This organization permits RAM cycle interleaving and permits error scrubbing during ECC refresh cycles. RAM cycle interleaving overlaps the start of the next RAM cycle with the RAM Precharge period of the previous cycle. Hiding the precharge period of one RAM cycle behind the data access period of the next RAM cycle optimizes memory bandwidth and is effective as long as successive RAM cycles occur in alternate banks.

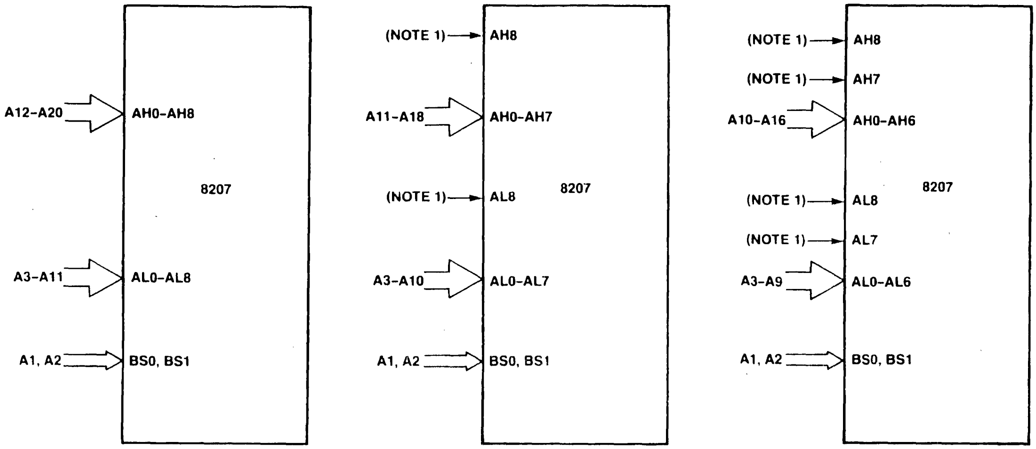
Successive data access to the same bank will cause the 8207 to wait for the precharge time of the previous RAM cycle.



210463-10

Figure 3. 8086/80186 Dual Port System

NOTE:
*These components are not necessary when using the 80186 components. These functions are provided directly by the 80186.



210463-11 **256K RAM Interface**

210463-12 **64K RAM Interface**

210463-13 **16K RAM Interface**

NOTES:

1. Unassigned address input pins should be strapped high or low.
2. A0 along with BHE are used to select a byte within a processor word.
3. Low order address bits are used as bank select inupts so that consecutive memory access requests are to alternate banks allowing bank interleaving of memory cycles.

Figure 4. Processor Address Interface to the 8207 Using 16K, 64K, and 256K RAMs

If not all RAM banks are occupied, the 8207 reassigns the RAS and CAS strobes to allow using wider data words without increasing the loading on the RAS and CAS drivers. Table 2 shows the bank selection decoding and the word expansion, including RAS and CAS assignments. For example, if only two RAM banks are occupied, then two RAS and two CAS strobes are activated per bank. Program bits RB1 and RB0 are not used to check the bank select inputs BS1 and BS0. The system design must protect from accesses to "illegal", non-existent banks of memory, by deactivating the PEA, PEB inputs when addressing an illegal bank.

The 8207 can interface to fast or slow RAMs. The 8207 adjusts and optimizes internal timings for either the fast or slow RAMs as programmed. (See RAM Speed Option.)

Memory Initialization

After programming, the 8207 performs eight RAM "warm-up" cycles to prepare the dynamic RAM for proper device operation. During "warm-up" some RAM parameters, such as tRAH, tASC, may not be met. This causes no harm to the dynamic RAM array. If configured for operation with error correction, the 8207 and 8206 EDCU will proceed to initialize all of memory (memory is written with zeros with corresponding check bits).

Table 2. Bank Selection Decoding and Word Expansion

Program Bits		Bank Input		RAS/CAS Pair Allocation
RB1	RB0	BS1	BS0	
0	0	0	0	RAS$_{0-3}$, CAS$_{0-3}$ to Bank 0
0	0	0	1	Illegal
0	0	1	0	Illegal
0	0	1	1	Illegal
0	1	0	0	RAS$_{0,1}$, CAS$_{0,1}$ to Bank 0
0	1	0	1	RAS$_{2,3}$, CAS$_{2,3}$ to Bank 1
0	1	1	0	Illegal
0	1	1	1	Illegal
1	0	0	0	RAS$_0$, CAS$_0$ to Bank 0
1	0	0	1	RAS$_1$, CAS$_1$ to Bank 1
1	0	1	0	RAS$_2$, CAS$_2$ to Bank 2
1	0	1	1	Illegal
1	1	0	0	RAS$_0$, CAS$_0$ to Bank 0
1	1	0	1	RAS$_1$, CAS$_1$ to Bank 1
1	1	1	0	RAS$_2$, CAS$_2$ to Bank 2
1	1	1	1	RAS$_3$, CAS$_3$ to Bank 3

Because the time to initialize memory is fairly long, the 8207 may be programmed to skip initialization in ECC mode. The time required to initialize all of memory is dependent on the clock cycle time to the 8207 and can be calculated by the following equation:

$$T_{INIT} = (2^{23})\ T_{CLCL} \qquad (1)$$

if $T_{CLCL} = 125$ ns then $T_{INIT} \approx 1$ sec.

8206 ECC Interface

For operation with Error Checking and Correction (ECC), the 8207 adjusts its internal timing and changes some pin functions to optimize performance and provide a clean dual-port memory interface between the 8206 EDCU and memory. The 8207 directly supports a master-only (16-bit word plus 6 check bits) system. Under extended operation and reduced clock frequency, the 8207 will support any ECC master-slave configuration up to 80 data bits, which is the maximum set by the 8206 EDCU. (See Extend Option.)

Correctable errors detected during memory read cycles are corrected immediately and then written back into memory.

In a synchronous bus environment, ECC system performance has been optimized to enhance processor throughput, while in an asynchronous bus environment (the Multibus), ECC performance has been optimized to get valid data onto the bus as quickly as possible. Performance optimization, processor throughput or quick data access may be selected via the Transfer Acknowledge Option.

The main difference between the two ECC implementations is that, when optimized for processor throughput, RAM data is always corrected and an advanced transfer acknowledge is issued at a point when, by knowing the processor characteristics, data is guaranteed to be valid by the time the processor needs it.

When optimized for quick data access, (valid for Multibus) the 8206 is configured in the uncorrecting mode where the delay associated with error correction circuitry is transparent, and a transfer acknowledge is issued as soon as valid data is known to exist. If the $\overline{\text{ERROR}}$ flag is activated, then the transfer acknowledge is delayed until after the 8207 has instructed the 8206 to correct the data and the corrected data becomes available on the bus. Figure 5 illustrates a dual-port ECC system.

Figure 6 illustrates the interface required to drive the $\overline{\text{CRCT}}$ pin of the 8206, in the case that one port (PORT A) receives an advanced acknowledge (not Multibus-compatible), while the other port (PORT B) receives $\overline{\text{XACK}}$ (which is Multibus-compatible).

Error Scrubbing

The 8207/8206 performs error correction during refresh cycles (error scrubbing). Since the 8207 must refresh RAM, performing error scrubbing during refresh allows it to be accomplished without additional performance penalties.

Upon detection of a correctable error during refresh, the RAM refresh cycle is lengthened slightly to permit the 8206 to correct the error and for the corrected word to be rewritten into memory. Uncorrectable errors detected during scrubbing are ignored.

Refresh

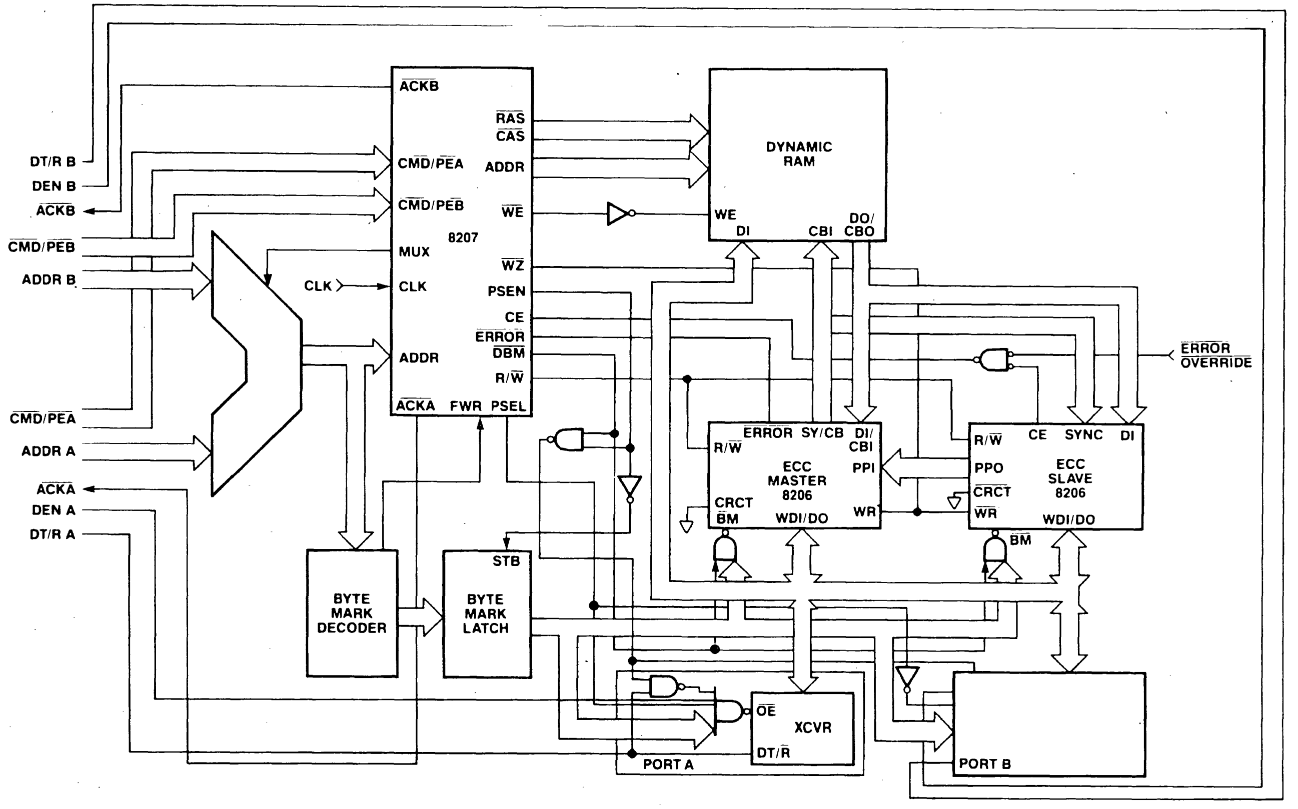
The 8207 provides an internal refresh interval counter and a refresh address counter to allow the 8207 to refresh memory. The 8207 will refresh 128 rows every 2 milliseconds or 256 rows every 4 milliseconds, which allows all RAM refresh options to be supported. In addition, there exists the ability to refresh 256 row address locations every 2 milliseconds via the Refresh Period programming option.

The 8207 may be programmed for any of four different refresh options: Internal refresh only, External refresh with failsafe protection, External refresh without failsafe protection, Burst Refresh mode, or no refresh. (See Refresh Options.)

It is possible to decrease the refresh time interval by 10%, 20% or 30%. This option allows the 8207 to compensate for reduced clock frequencies. Note that an additional 5% interval shortening is built-in in all refresh interval options to compensate for clock variations and non-immediate response to the internally generated refresh request. (See Refresh Period Options.)

External Refresh Requests after RESET

External refresh requests are not recognized by the 8207 until after it is finished programming and preparing memory for access. Memory preparation includes 8 RAM cycles to prepare and ensure proper



210463-14

Figure 5. Two-Port ECC Implementation Using the 8207 and the 8206

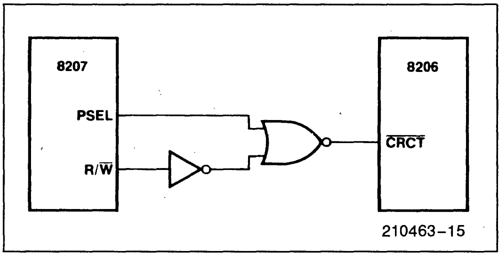


Figure 6. Interface to 8206 $\overline{\text{CRCT}}$ Input when Port A Receives $\overline{\text{AACK}}$ and Port B Receives $\overline{\text{XACK}}$

dynamic RAM operation, and memory initialization if error correction is used. Many dynamic RAMs require this warm-up period for proper operation. The time it takes for the 8207 to recognize a request is shown below.

Non-ECC Systems:

$$T_{RESP} = T_{PROG} + T_{PREP} \tag{2}$$

where:

$$T_{PROG} = (66)\ (T_{CLCL}) \tag{3}$$

which is programming time

$$T_{PREP} = (8)\ (32)\ (T_{CLCL})$$

which is the RAM warm-up time

if $T_{CLCL} = 125$ ns then $T_{RESP} \approx 41\ \mu s$

ECC Systems:

$$T_{RESP} = T_{PROG} + T_{PREP} + T_{INIT} \tag{5}$$

if $T_{CLCL} = 125$ ns then $T_{RESP} \approx 1$ sec.

RESET

RESET is an asynchronous input, the falling edge of which is used by the 8207 to directly sample to logic levels of the PCTLA, PCTLB, RFRQ, and PDI inputs. The internally synchronized falling edge of RESET is used to begin programming operations (shifting in the contents of the external shift register into the PDI input).

Until programming is complete the 8207 registers but does not respond to command or status inputs. A simple means of preventing commands or status from occurring during this period is to differentiate the system reset pulse to obtain a smaller reset pulse for the 8207. The total time of the reset pulse and the 8207 programming time must be less than the time before the first command in systems that alter the default port synchronization programming bits (default is Port A synchronous, Port B asynchronous). Differentiated reset is unnecessary when the default port synchronization programming is used.

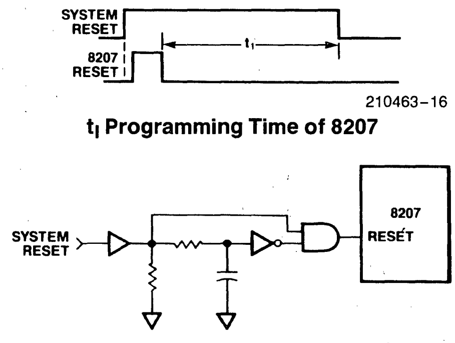
The differentiated reset pulse would be shorter than the system reset pulse by at least the programming period required by the 8207. The differentiated reset pulse first resets the 8207, and system reset would reset the rest of the system. While the rest of the system is still in reset, the 8207 completes its programming. Figure 7 illustrates a circuit to accomplish this task.

Within four clocks after RESET goes active, all the 8207 outputs will go high, except for PSEN, WE, and AO0-2, which will go low.

OPERATIONAL DESCRIPTION

Programming the 8207

The 8207 is programmed after reset. On the falling edge of RESET, the logic states of several input pins are latched internally. The falling edge of RESET actually performs the latching, which means that the logic levels on these inputs must be stable prior to that time. The inputs whose logic levels are latched at the end of reset are the PCTLA, PCTLB, REFRQ, and PDI pins. Figure 8 shows the necessary timing for programming the 8207.

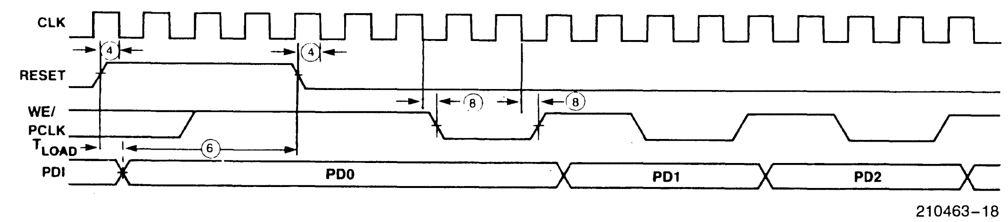


t_I Programming Time of 8207

Differentiated Reset

NOTES:
1. Required only when the port synchronization options (SA & SB) are altered from their initial default values.
2. V_{CC} must be stable before system reset is activated when using this circuit.

Figure 7. 8207 Differentiated Reset Circuit



NOTES:
TRTVCL—Reset is an asynchronous input, if reset occurs before T_1, then it is guaranteed to be recognized.
TPGVCL—Minimum PDI valid time prior to reset going low.
TCLPC—MUX/PCLK delay.
TLOAD—Asynchronous load data propagation delay.

Figure 8. Timing Illustrating External Shift Register Requirements for Programming the 8207

Status/Command Mode

The two processor ports of the 8207 are configured by the states of the PCTLA and PCTLB pins. Which interface is selected depends on the state of the individual port's PCTL pin at the end of reset. If PCTL is high at the end of the reset, the 8086 Status interface is selected; if it is low, then the Command interface is selected.

The status lines of the 80286 are similar in code and timing to the Multibus command lines, while the status code and timing of the 8086 and 8088 are identical to those of the 80186 and 80188 (ignoring the differences in clock duty cycle). Thus there exists two interface configurations, one for the 80286 status or Multibus memory commands, which is called the Command interface, and one for 8086, 8088, 80186 or 80188 status, called the 8086 Status interface. The Command interface can also directly interface to the command lines of the bus controllers for the 8086, 8088, 80186 and the 80286.

The 8086 Status interface allows direct decoding of the status of the iAPX 86, iAPX 88, iAPX 186 and the iAPX 188. Table 3 shows how the status lines are decoded. While in the Command mode the iAPX 286 status can be directly decoded. Microprocessor bus controller read or write commands or Multibus commands can also be directed to the 8207 when in Command mode.

Refresh Options

Immediately after system reset, the state of the REFRQ input pin is examined. If REFRQ is high, the 8207 provides the user with the choice between self-refresh or user-generated refresh with failsafe protection. Failsafe protection guarantees that if the

Table 3A. Status Coding of 8086, 80186 and 80286

Status Code			Function	
$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	8086/80186	80286
0	0	0	Interrupt	Interrupt
0	0	1	I/O Read	I/O Read
0	1	0	I/O Write	I/O Write
0	1	1	Halt	Idle
1	0	0	Instruction Fetch	Halt
1	0	1	Memory Read	Memory Read
1	1	0	Memory Write	Memory Write
1	1	1	Idle	Idle

Table 3B. 8207 Response

8207 Command			Function	
PCTL	$\overline{RD}$	$\overline{WR}$	8086/80186 Status Interface	80286 Status or Command Interface
0	0	0	Ignore	Ignore*
0	0	1	Ignore	Read
0	1	0	Ignore	Write
0	1	1	Ignore	Ignore*
1	0	0	Read	Ignore
1	0	1	Read	Inhibit
1	1	0	Write	Inhibit
1	1	1	Ignore	Ignore

*Illegal with CFS = 0

user does not come back with another refresh request before the internal refresh interval counter times out, a refresh request will be automatically generated. If the REFRQ pin is low immediately after a reset, the 8207 is programmed in a non-failsafe refresh mode. In this mode the refresh cycle is initiated only upon receipt of an external refresh request. The user has the choice of a single external refresh cycle, burst refresh or no refresh.

Internal Refresh Only

For the 8207 to generate internal refresh requests, it is necessary only to strap the REFRQ input pin high.

External Refresh with Failsafe

To allow user-generated refresh requests with failsafe protection, it is necessary to hold the REFRQ input high until after reset. Thereafter, a low-to-high transition on this input causes a refresh request to be generated and the internal refresh interval counter to be reset. A high-to-low transition has no effect on the 8207. A refresh request is not recognized until a previous request has been serviced.

External Refresh without Failsafe

To generate single external refresh requests without failsafe protection, it is necessary to hold REFRQ low until after reset. Thereafter, bringing REFRQ high for one clock period causes refresh request to be generated. A refresh request is not recognized unitl a previous request has been serviced.

Burst Refresh

Burst refresh is implemented through the same procedure as a single external refresh without failsafe (i.e., REFRQ is kept low until after reset). Thereafter, bringing REFRQ high for at least two clock periods causes a burst of up to 128 row address locations to be refreshed.

The ECC-configured systems, 128 locations are scrubbed. Any refresh request is not recognized until a previous request has been serviced (i.e., burst completed).

No Refresh

It is necessary to hold REFRQ low until after reset. This is the same as programming External Refresh without Failsafe. No refresh is accomplished by keeping REFRQ low.

Option Program Data Word

The program data word consists of 16 program data bits, PD0–PD15. If the first program data bit shifted into the 8207 (PD0) is set to logic 1, the 8207 is configured to support ECC. If it is logic 0, the 8207 is configured to support a non-ECC system. The remaining bits, PD1–PD15, may then be programmed to optimize a selected configuration. Figures 9 and 10 show the Program words for non-ECC and ECC operation.

Using an External Shift Register

The 8207 may be configured to use an external shift register with asynchronous load capability such as a 74LS165. The reset pulse serves to parallel load the shift register and the 8207 supplies the clocking signal to shift the data in. Figure 11 shows a sample circuit diagram of an external shift register circuit.

Serial data is shifted into the 8207 via the PDI pin (57), and clock is provided by the MUX/PCLK pin (12), which generates a total of 16 clock pulses. After programming is complete, data appearing at the input of the PDI pin is ignored. MUX/PCLK is a dual-function pin. During programming, it serves to clock the external shift register, and after programming is completed, it reverts to a MUX conrol pin. As the pin changes state to select different port addresses, it continues to clock the shift register. This does not present a problem because data at the PDI pin is ignored after programming. Figure 8 illustrates the timing requirements of the shift register circuitry.

ECC Mode (ECC Program Bit)

The state of PDI (Program Data In) pin at reset determines whether the system is an ECC or non-ECC configuration. It is used internally by the 8207 to begin configuring timing circuits, even before programming is completely finished. The 8207 then begins programming the rest of the options.

Default Programming Options

After reset, the 8207 serially shifts in a program data word via the PDI pin. This pin may be strapped either high or low, or connected to an external shift register. Strapping PDI high causes the 8207 to default to a particular system configuration with error correction, and strapping it low causes the 8207 to default to a particular system configuration without error correction. Table 4 shows the default configurations.

PD15							PD8	PD7							PD0
0	0	TM 1	$\overline{\text{PPR}}$	$\overline{\text{FFS}}$	EXT	$\overline{\text{PLS}}$	CI0	CI1	$\overline{\text{RB1}}$	$\overline{\text{RB0}}$	$\overline{\text{RFS}}$	$\overline{\text{CFS}}$	SB	$\overline{\text{SA}}$	0

Program Data Bit	Name	Polarity/Function
PD0	ECC	ECC = 0 For Non-ECC Mode
PD1	$\overline{\text{SA}}$	$\overline{\text{SA}}$ = 0 Port A is Synchronous $\overline{\text{SA}}$ = 1 Port A is Asnychronous

PD2	SB	SB = 0 Port B is Asynchronous
		SB = 1 Port B is Synchronous

PD3	$\overline{\text{CFS}}$	$\overline{\text{CFS}}$ = 0 Fast-Cycle iAPX 286 Mode $\overline{\text{CFS}}$ = 1 Slow-Cycle iAPX 86 Mode
PD4	$\overline{\text{RFS}}$	$\overline{\text{RFS}}$ = 0 Fast RAM $\overline{\text{RFS}}$ = 1 Slow RAM
PD5 PD6	$\overline{\text{RB0}}$ $\overline{\text{RB1}}$	RAM Bank Occupancy See Table 2
PD7 PD8	CI1 CI0	Count Interval Bit 1; see Table 6 Count Interval Bit 0; see Table 6
PD9	$\overline{\text{PLS}}$	$\overline{\text{PLS}}$ = 0 Long Refresh Period $\overline{\text{PLS}}$ = 1 Short Refresh Period

PD10	EXT	EXT = 0 Not Extended
		EXT = 1 Extended

PD11	$\overline{\text{FFS}}$	$\overline{\text{FFS}}$ = 0 Fast CPU Frequency $\overline{\text{FFS}}$ = 1 Slow CPU Frequency
PD12	$\overline{\text{PPR}}$	$\overline{\text{PPR}}$ = 0 Most Recently Used Port Priority $\overline{\text{PPR}}$ = 1 Port A Preferred Priority

PD13	TM1	TM1 = 0 Test Mode 1 Off
		TM1 = 1 Test Mode 1 Enabled

PD14	0	Reserved, Must be Zero
PD15	0	Reserved, Must be Zero

Figure 9. Non-ECC Mode Program Data Word

PD15							PD8	PD7							PD0
$\overline{\text{TM2}}$	RB1	RB0	PPR	FFS	$\overline{\text{EXT}}$	PLS	$\overline{\text{CI0}}$	$\overline{\text{CI1}}$	XB	$\overline{\text{XA}}$	RFS	CFS	$\overline{\text{SB}}$	SA	1

Program Data Bit	Name	Polarity/Function
PD0	ECC	ECC = 1 ECC Mode

PD1	SA	SA = 0 Port A Asynchronous
		SA = 1 Port A Synchronous

PD2	$\overline{\text{SB}}$	$\overline{\text{SB}}$ = 0 Port B Synchronous $\overline{\text{SB}}$ = 1 Port B Asynchronous

PD3	CFS	CFS = 0 Slow-Cycle iAPX 86 Mode
		CFS = 1 Fast-Cycle iAPX 286 Mode

PD4	RFS	RFS = 0 Slow RAM
		RFS = 1 Fast RAM

PD5	$\overline{\text{XA}}$	$\overline{\text{XA}}$ = 0 MULTIBUS-Compatible ACKA $\overline{\text{XA}}$ = 1 Advanced ACKA Not Multibus-Compatible
PD6	XB	Advanced ACKB Not Multibus-Compatible XB = 1 Multibus-Compatible ACKB
PD7 PD8	$\overline{\text{CI1}}$ $\overline{\text{CI0}}$	Count Interval Bit 1; see Table 6 Count Interval Bit 0; see Table 6

PD9	PLS	PLS = 0 Short Refresh Period
		PLS = 1 Long Refresh Period

PD10	$\overline{\text{EXT}}$	$\overline{\text{EXT}}$ = 0 Master and Slave EDCU $\overline{\text{EXT}}$ = 1 Master EDCU Only

PD11	FFS	FFS = 0 Slow CPU Frequency
		FFS = 1 Fast CPU Frequency

PD12	PPR	PPR = 0 Port A Preferred Priority
		PPR = 1 Most Recently Used Port Priority

PD13 PD14	RB0 RB1	RAM Bank Occupancy See Table 2
PD15	$\overline{\text{TM2}}$	$\overline{\text{TM2}}$ = 0 Test Mode 2 Enabled $\overline{\text{TM1}}$ = 1 Test Mode 2 Off

Figure 10. ECC Mode Program Data Word

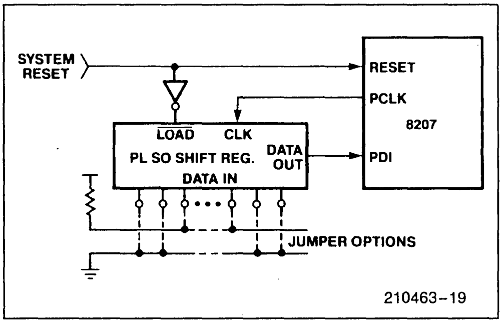


Figure 11. External Shift Register Interface

Table 4A. Default Non-ECC Programming, PDI Pin (57) Tied to Ground

Port A is Synchronous (EAACKA and XACKA)
Port B is Asynchronous (LAACKB and XACKB)
Fast-Cycle Processor Interface (iAPX 286)
Fast RAM
Refresh Interval uses 236 Clocks
128 Row Refresh in 2 ms; 256 Row Refresh in 4 ms
Fast Processor Clock Frequency (16 MHz)
"Most Recently Used" Priority Scheme
4 RAM banks occupied

Table 4B. Default ECC Programming, PDI Pin (57) Tied to V_{CC}

Port A is Synchronous
Port B is Asynchronous
Fast-Cycle Processor Interface (iAPX 286)
Fast RAM
Port A has EAACKA strobe (non-multibus)
Port B has XACKB strobe (multibus)
Refresh interval uses 236 clocks
128 Row refresh in 2 ms; 256 Row refresh in 4 ms
Master EDCU only (16-bit system)
Fast Processor Clock Frequency (16 MHz)
"Most Recently Used" Priority Scheme
4 RAM banks ocuppied

If further system flexibility is needed, one or two external shift registers can be used to tailor the 8207 to its operating environment.

Synchronous/Asynchronous Mode (SA and SB Program Bits)

Each port of the 8207 may be independently configured to accept synchronous or asynchronous port commands (RD, WR, PCTL) and Port Enable (PE) via the program bits SA and SB. The state of the SA and SB programming bits determine whether their associated ports are synchronous or asynchronous.

While a port may be configured with either the Status or Command interface in the synchronous mode, certain restrictions exist in the asynchronous mode. An asynchronous Command interface using the control lines of the Multibus is supported, and an asynchronous 8086 interface using the control lines of the 8086 is supported, with the use of TTL gates as illustrated in Figure 2. In the 8086 case, the TTL gates are needed to guarantee that status does not appear at the 8207's inputs too much before address, so that a cycle would start before address was valid.

Microprocessor Clock Frequency Option (CFS and FFS Program Bits)

The 8207 can be programmed to interface with slow-cycle microprocessors like the 8086, 8088, 81088 and 80186 or fast-cycle microprocessors like the 80286. The CFS bit configures the microprocessor interface to accept slow or fast cycle signals from either microprocessor group.

The FFS bit is used to select the speed of the microprocessor clock. Table 5 shows the various microprocessor clock frequency options that can be programmed.

Table 5. Microprocessor Clock Frequency Options

Program Bits		Processor	Clock Frequency
CFS	FFS		
0	0	iAPX 86, 88, 186, 188	≤ 6 MHz
0	1	iAPX 86, 88, 186, 188	> 6 MHz
1	0	iAPX 286	≤12 MHz
1	1	iAPX 286	>12 MHz

The external clock frequency must be programmed so that the failsafe refresh repetition circuitry can adjust its internal timing accordingly to produce a refresh request as programmed.

RAM Speed Option (RFS Program Bit)

The RAM Speed programming option determines whether RAM timing will be optimized for a fast or slow RAM.

Refresh Period Options (CI0, CI1 and PLS Program Bits)

The 8207 refreshes with either 128 rows every 2 milliseconds or 256 rows every 4 milliseconds. This translates to one refresh cycle being executed approximately once every 15.6 microseconds. This rate can be changed to 256 rows every 2 milliseconds or a refresh approximately once every 7.8 microseconds via the Period Long/Short, program bit PLS, programming option. The 7.8 microsecond refresh request rate is intended for those RAMs, 64K and above, which may require a faster refresh rate.

In addition to PLS program option, two other programming bits for refresh exist: Count Interval 0 (CI0) and Count Interval 1 (CI1). These two programming bits allow the rate at which refresh requests are generated to be increased in order to permit refresh requests to be generated close to the same 15.6 or 7.8 microsecond period when the 8207 is operating at reduced frequencies. The interval between refreshes is decreased by 0%, 10%, 20%, or 30% as a function of how the count interval bits are programmed. A 5% guardband is built-in to allow for any clock frequency variations. Table 6 shows the refresh period options available.

The numbers tabulated under Count Interval represent the number of clock periods between internal refresh requests. The percentages in parentheses represent the decrease in the interval between refresh requests. Note that all intervals have a built-in 5% (approximately) safety factor to compensate for minor clock frequency deviations and non-immediate response to internal refresh requests.

Extend Option (EXT Program Bit)

The Extend option lengthens the memory cycle to allow longer access time which may be required by the system. Extend alters the RAM timing to compensate for increased loading on the Row and Column Address Strobes, and in the multiplexed Address Out lines.

Port Priority Option and Arbitration (PPR Program Bit)

The 8207 has to internally arbitrate among three ports: Port A, Port B and Port C—the refresh port. Port C is an internal port dedicated to servicing refresh requests, whether they are generated internally by the refresh interval counter, or externally by the user. Two arbitration approaches are available via

Table 6. Refresh Count Interval Table

Ref. Period (μs)	CFS	PLS	FFS	Count Interval CI1, CI0 (8207 Clock Periods)			
				00 (0%)	01 (10%)	10 (20%)	11 (30%)
15.6	1	1	1	236	212	188	164
7.8	1	0	1	118	106	94	82
15.6	1	1	0	148	132	116	100
7.8	1	0	0	74	66	58	50
15.6	0	1	1	118	106	94	82
7.8	0	0	1	59	53	47	41
15.6	0	1	0	74	66	58	50
7.8	0	0	0	37	33	29	25

NOTE:
Refresh period = clock period × refresh count interval.

the Port Priority programming option, program bit PPR. PPR determines whether the most recently used port will remain selected (PPR = 1) or whether Port A will be favored or preferred over Port B (PPR = 0).

A port is selected if the arbiter has given the selected port direct access to the timing generators. The front-end logic, which includes the arbiter, is designed to operate in parallel with the selected port. Thus a request on the selected port is serviced immediately. In contrast, an unselected port only has access to the timing generators through the front-end logic. Before a RAM cycle can start for an unselected port, that port must first become selected (i.e., the MUX output now gates that port's address into the 8207 in the case of Port A or B). Also, in order to allow its address to stabilize, a newly selected port's first RAM cycle is started by the front-end logic. Therefore, the selected port has direct access to the timing generators. What all this means is that a request on a selected port is started immediately, while a request on an unselected port is started two to three clock periods after the request, assuming that the other two ports are idle. Under normal operating conditions, this arbitration time is hidden behind the RAM cycle of the selected port so that as soon as the present cycle is over a new cycle is started. Table 7 lists the arbitration rules for both options.

Port LOCK Function

The LOCK function provides each port with the ability to obtain uninterrupted access to a critical region of memory and, thereby, to guarantee that the opposite port cannot "sneak in" and read from or write to the critical region prematurely.

Only one LOCK pin is present and is multiplexed between the two ports as follows: when MUX is high, the 8207 treats the LOCK input as originating at PORT A, while when MUX is low, the 8207 treats LOCK as originating at PORT B. When the 8207 recognizes a LOCK, the MUX output will remain pointed to the locking port until LOCK is deactivated. Refresh is not affected by LOCK and can occur during a locked memory cycle.

Table 7. The Arbitration Rules for the Most Recently Used Port Priority and for Port A Priority Options Are As Follows:

1.	If only one port requests service, then that port—if not already selected—becomes selected.
2a.	When no service requests are pending, the last selected processor port (Port A or B) will remain selected. (Most Recently Used Port Priority Option.)
2b.	When no service requests are pending, Port A is selected whether it requests service or not. (Port A Priority Option.)
3.	During reset initialization only Port C, the refresh port, is selected.
4.	If no processor requests are pending after reset initialization, Port A will be selected.
5b.	If Ports A and B simultaneously(*) request service while Port C is selected, then the next port to be selected is Port A. (Port A Priority Option.)
6.	If a port simultaneously requests service with the currently selected port, service is granted to the selected port.
7.	The MUX output remains in its last state whenever Port C is selected.
8.	If Port C and either Port A or Port B (or both) simultaneously request service, then service is granted to the requester whose port is already selected. If the selected port is not requesting service, then service is granted to Port C.
9.	If during the servicing of one port, the other port requests service before or simultaneously with the refresh port, the refresh port is selected. A new port is not selected before the presently selected port is deactivated.
10.	Activating LOCK will mask off service requests from Port B if the MUX output is high, or from Port A if the MUX output is low.

NOTE:
*By "simultaneous" it is meant that two or more requests are valid at the clock edge at which the internal arbiter samples them.

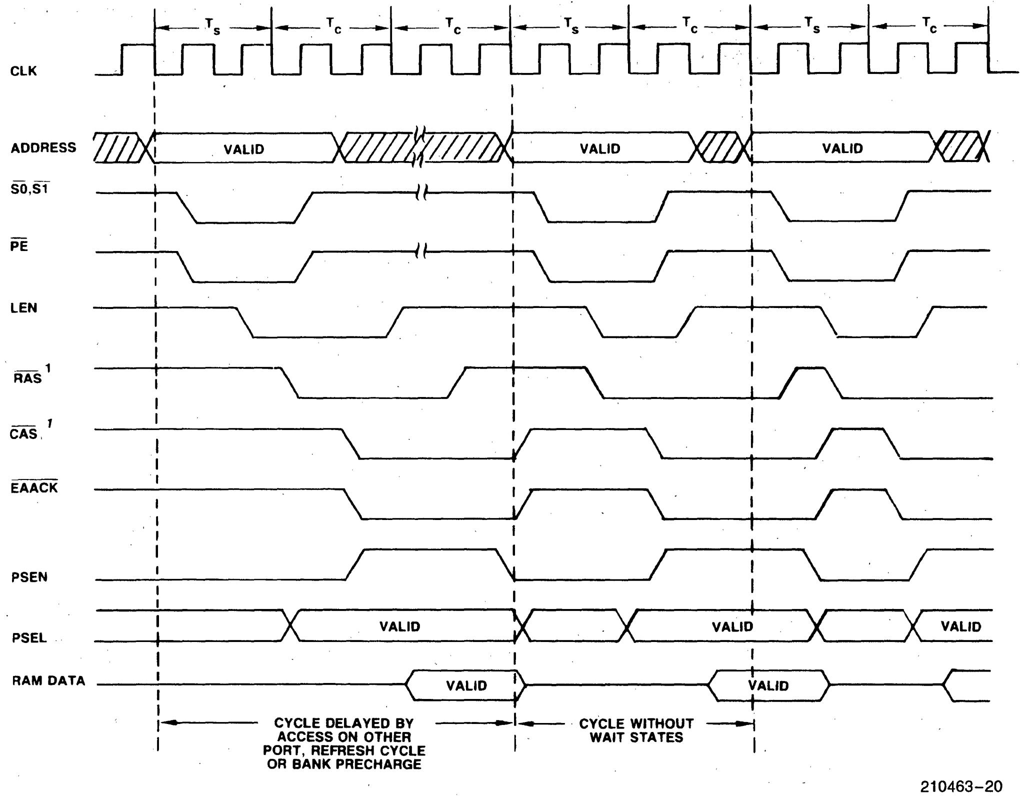
Dual-Port Considerations

For both ports to be operated synchronously, several conditions must be met. The processors must be the same type (Fast or Slow Cycle) as defined by Table 8 and they must have synchronized clocks. Also when processor types are mixed, even though the clocks may be in phase, one frequency may be twice that of the other. So to run both ports synchronous using the status interface, the processors must have related timings (both phase and frequency). If these conditions cannot be met, then one port must run synchronous and the other asynchronous.

Figure 3 illustrates an example of dual-port operation using the processors in the slow cycle group. Note the use of cross-coupled NAND gates at the MUX output for minimizing contention between the two latches, and the use of flip flops on the status lines of the asynchronous processor for delaying the status and thereby guaranteeing RAS will not be issued, even in the worst case, until address is valid.

Processor Timing

In order to run without wait states, $\overline{\text{AACK}}$ must be used and connected to the $\overline{\text{SRDY}}$ input of the appropriate bus controller. $\overline{\text{AACK}}$ is issued relative to a point within the RAM cycle and has no fixed relationship to the processor's request. The timing is such, however, that the processor will run without wait states, barring refresh cycles, bank precharge, and RAM accesses from the other port. In non-ECC fast cycle, fast RAM, non-extended configurations (80286), $\overline{\text{AACK}}$ is issued on the next falling edge of



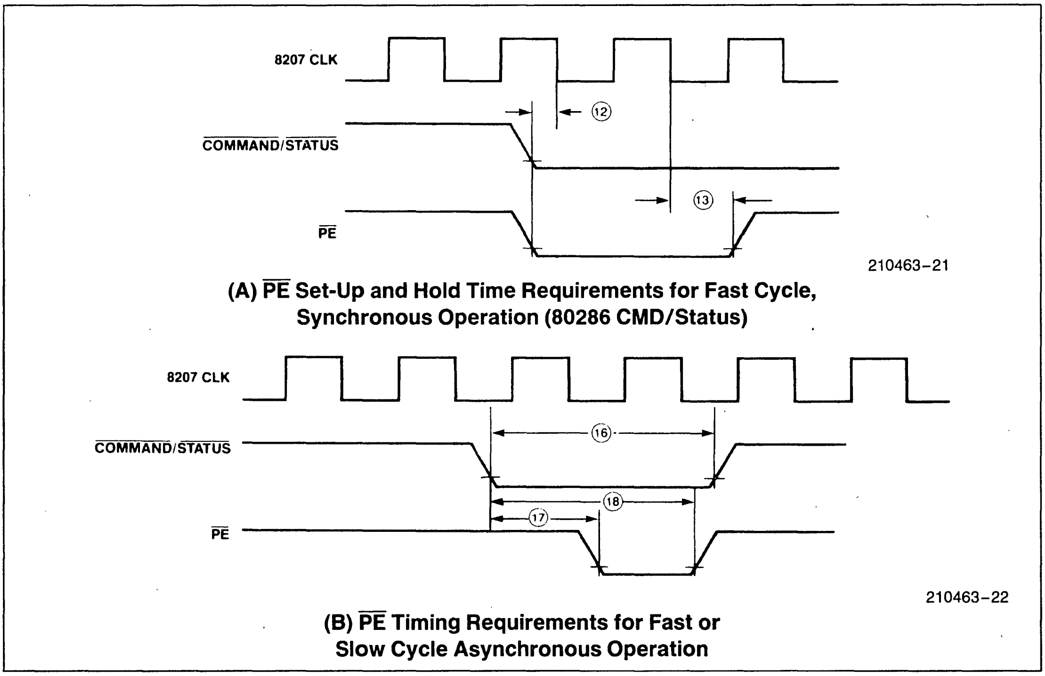
NOTE:
1. The RAS and CAS shown in figure are different banks being accessed.

Figure 14. iAPX 286/8207 Synchronous-Status Timing Programmed in non-ECC Mode, C0 Configuration (Read Cycle)

the clock after the edge that issues RAS. In non-ECC, slow cycle, non-extended, or extended with fast RAM cycle configurations (8086, 80188, 80186), $\overline{AACK}$ is issued on the same clock cycle that issues RAS. Figure 14 illustrates the timing relationship between $\overline{AACK}$, the RAM cycle, and the processor cycle for several different situations.

Port Enable ($\overline{PE}$) setup time requirements depend on whether the associated port is configured for synchronous or asynchronous fast or slow cycle operation. In a synchronous fast cycle configuration, $\overline{PE}$ is required to be setup to the same clock edge as the status or commands. If $\overline{PE}$ is true (low), a RAM cycle is started; if not, the cycle is aborted. The memory cycle will only begin when both valid signals ($\overline{PE}$ and $\overline{RD}$ or WR) are recognized at a particular clock edge. In asynchronous operation. $\overline{PE}$ is required to be setup to the same clock edge as the internally synchronized status or commands. Externally, this allows the internal synchronization delay to be added to the status (or command)-to-$\overline{PE}$ delay time, thus allowing for more external decode time that is available in synchronous operation.

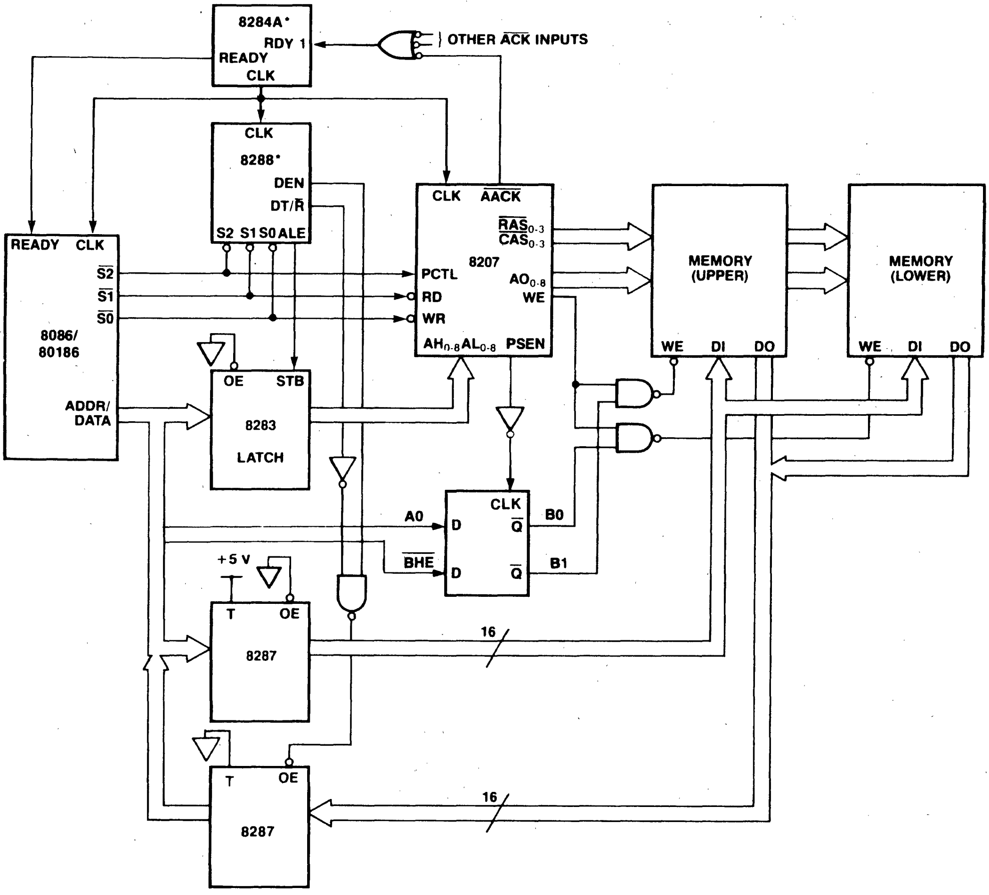
The minimum synchronization delay is the additional amount that $\overline{PE}$ must be held valid. If $\overline{PE}$ is not held valid for the maximum synchronization delay time, it is possible that $\overline{PE}$ will go invalid prior to the status or command being synchronized. In such a case the 8207 aborts the cycle. If a memory cycle intended for the 8207 is aborted, then no acknowledge ($\overline{AACK}$ or $\overline{XACK}$) is issued and the processor locks up in endless wait states. Figure 15 illustrates the status (command) timing requirements for synchronous and asynchronous systems. Figures 16 and 17 show a more detailed hook-up of the 8207 to the 8086 and the 80286, respectively.



(A) $\overline{PE}$ Set-Up and Hold Time Requirements for Fast Cycle, Synchronous Operation (80286 CMD/Status)

(B) $\overline{PE}$ Timing Requirements for Fast or Slow Cycle Asynchronous Operation

Figure 15



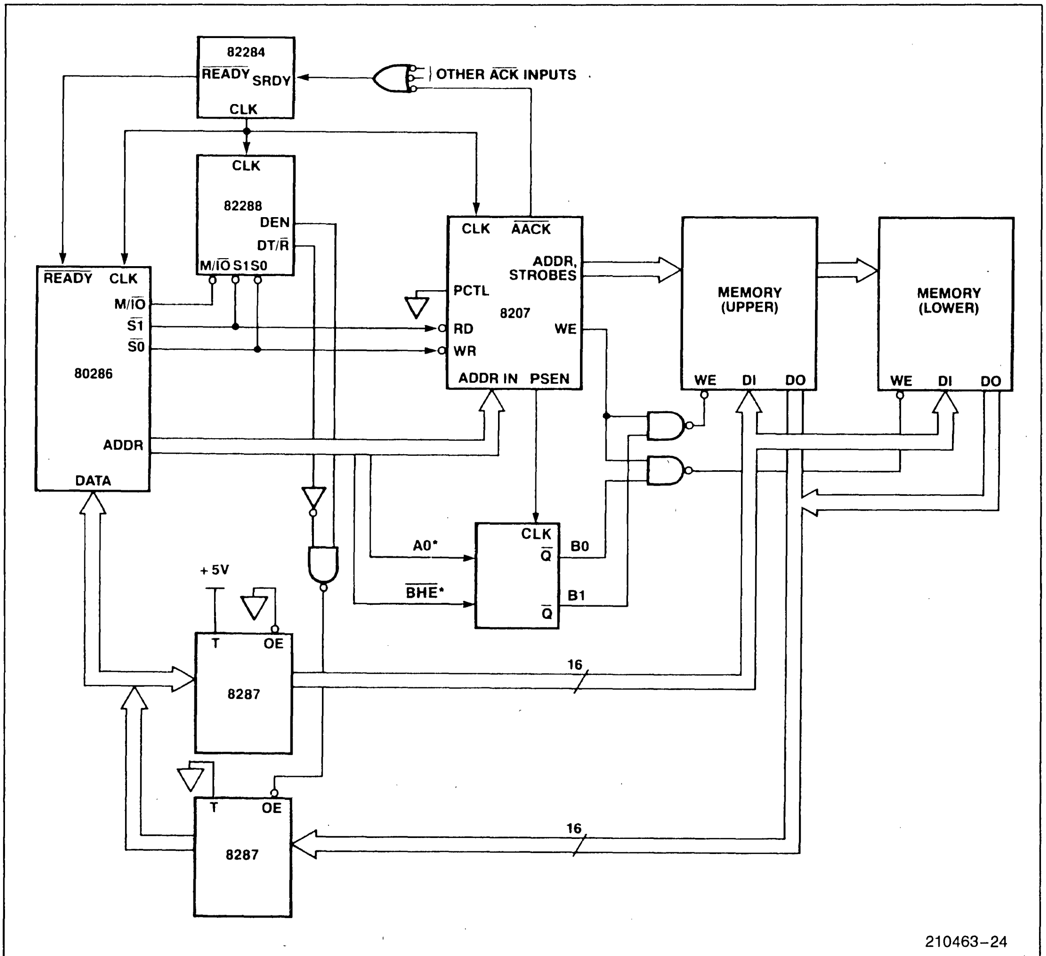
NOTE:
*These components are not necessary when using the 80186. These functions are provided directly by the 80186.

Figure 16. 8086/80186, 8207 Single Port Non-ECC Synchronous Systems

Memory Acknowledge (AACK, XACK)

In system configurations without error correction, two memory acknowledge signals per port are supplied by the 8207. They are the Advanced Acknowledge strobe (AACK) and the Transfer Acknowledge strobe (XACK). The CFS programming bit determines for which processor AACKA and AACKB are optimized, either 80286 (CFS = 1) or 8086/186 (CFS = 0), while the SA and SB programming bits optimize AACK for synchronous operation ("early" AACK) or asynchronous operation ("late" AACK).

Both the early and late AACK strobes are three clocks long for CFS = 1 and two clocks long for CFS = 0. The XACK strobe is asserted when data is valid (for reads) or when data may be removed (for writes) and meets the Multibus requirements. XACK is removed asynchronously by the command going



NOTE:
While the 8207 does not need the input addresses latched, A0, BHE must come from the latched address bus.

Figure 17. 80286 Hook-Up to 8207 Non-ECC Synchronous System-Single Port

inactive. Since in asynchronous operation the 8207 removes read data before late AACK or XACK is recognized by the CPU, the user must provide for data latching in the system until the CPU reads the data. In synchronous operation, data latching is unnecessary since the 8207 will not remove data until the CPU has read it.

In ECC-based systems there is one memory acknowledge (XACK or AACK) per port and a programming bit associated with each acknowledge. If the X programming bit is active, the strobe is configured as XACK, while if the bit is inactive, the strobe is configured as AACK. As in non-ECC, the SA and SB programming bits determine whether the AACK strobe is early or late (EAACK or LAACK).

Data will always be valid a fixed time after the occurrence of the advanced acknowledge. Table 9 summarizes the various transfer acknowledge options.

Table 8. Processor Interface/Acknowledge Summary

Cycle	Processor	Request Type	Sync/Async Interface	Acknowledge Type
Fast Cycle CFS=1	80286	Status	Sync	$\overline{\text{EAACK}}$
	80286	Status	Async	$\overline{\text{LAACK}}$
	80286	Command	Sync	$\overline{\text{EAACK}}$
	80286	Command	Async	$\overline{\text{LAACK}}$
	8086/80186	Status	Async	$\overline{\text{LAACK}}$
	8086/80186	Command	Async	$\overline{\text{LAACK}}$
	Multibus	Command	Async	$\overline{\text{XACK}}$
Slow Cycle CFS=0	8086/80186	Status	Sync	$\overline{\text{EAACK}}$
	8086/80186	Status	Async	$\overline{\text{LAACK}}$
	8086/80186	Command	Sync	$\overline{\text{EAACK}}$
	8086/80186	Command	Async	$\overline{\text{LAACK}}$
	Multibus	Command	Async	$\overline{\text{XACK}}$

Table 9. Memory Acknowledge Option Summary

	Synchronous	Asynchronous	XACK
Fast Cycle	AACK Optimized for Local 80286	AACK Optimized for Remote 80286	Multibus Compatible
Slow Cycle	AACK Optimized for Local 8086/186	AACK Optimized for Remote 8086/186	Multibus Compatible

Test Modes

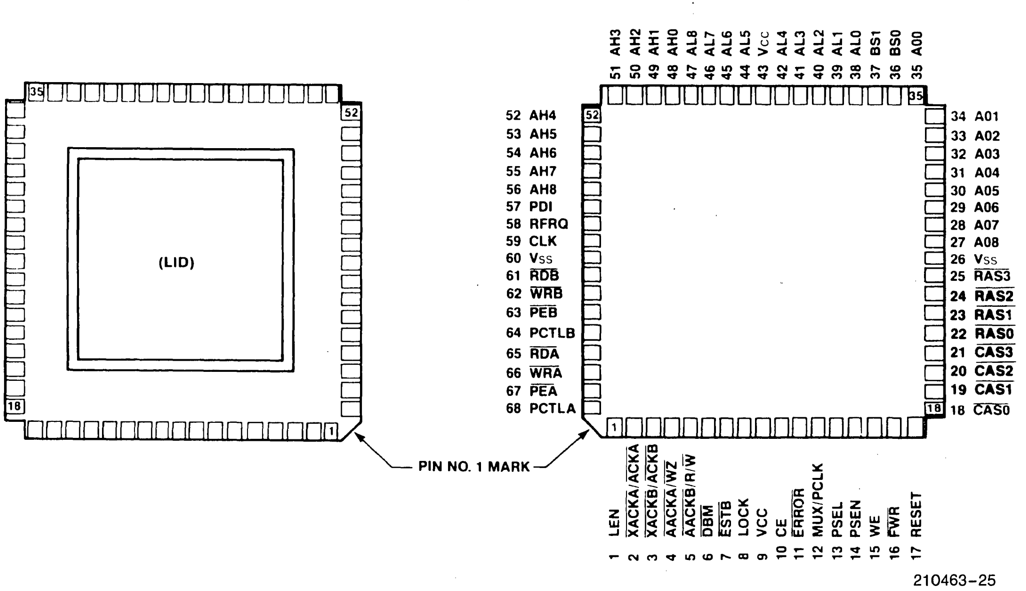
Two special test modes exist in the 8207 to facilitate testing. Test Mode 1 (non-ECC mode) splits the refresh address counter into two separate counters and Test Mode 2 (ECC mode) presets the refresh address counter to a value slightly less than rollover.

Test Mode 1 splits the address counter into two, and increments both counters simultaneously with each refresh address update. By generating external refresh requests, the tester is able to check for proper operation of both counters. Once proper individual counter operation has been established, the 8207 must be returned to normal mode and a second test performed to check that the carry from the first counter increments the second counter. The outputs of the counters are presented on the address out bus with the same timing as the row and column addresses of a normal scrubbing operation. During Test Mode 1, memory initialization is inhibited, since the 8207, be definition, is in non-ECC mode.

Test Mode 2 sets the internal refresh counter to a value slightly less than rollover. During functional testing other than that covered in Test Mode 1, the 8207 will normally be set in Test Mode 2. Test Mode 2 eliminates memory initialization in ECC mode. This allows quick examination of the circuitry which brings the 8207 out of memory initialization and into normal operation.

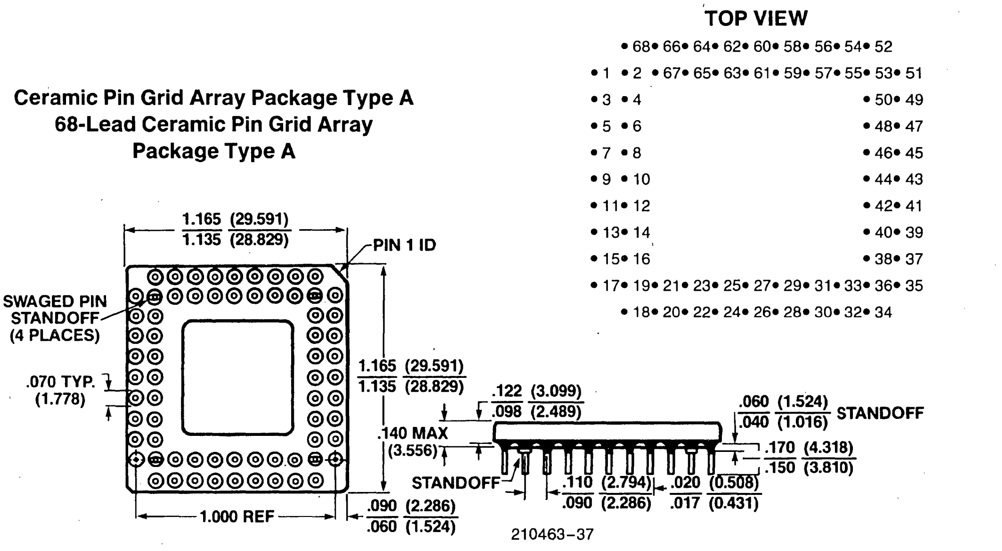
General System Considerations

The RAS$_{0-3}$, CAS$_{0-3}$, AO$_{0-8}$, output buffers were designed to directly drive the heavy capacitive loads associated with dynamic RAM arrays. To keep the RAM driver outputs from ringing excessively in the system environment and causing noise in other output pins it is necessary to match the output impedance of the RAM output buffers with the RAM array by using series resistors and to add series resistors to other control outputs for noise reduction if necessary. Each application may have different impedance characteristics and may require different series resistance values. The series resistance values should be determined for each application. In non-ECC systems unused ECC input pins should be tied high or low to improve noise immunity.



NOTE:
LCC is mounted lid-down into socket.

Figure 19. 8207 Pinout Diagram



TOP VIEW

```
    • 68• 66• 64• 62• 60• 58• 56• 54• 52
• 1  • 2  • 67• 65• 63• 61• 59• 57• 55• 53• 51
• 3  • 4                              • 50• 49
• 5  • 6                              • 48• 47
• 7  • 8                              • 46• 45
• 9  • 10                             • 44• 43
• 11• 12                              • 42• 41
• 13• 14                              • 40• 39
• 15• 16                              • 38• 37
• 17• 19• 21• 23• 25• 27• 29• 31• 33• 36• 35
    • 18• 20• 22• 24• 26• 28• 30• 32• 34
```

8207 Pin Grid Array (PGA) Pin-Out

Packaging

The 8207 is packaged in a 68 lead JEDEC Type A Leadless Chip Carrier (LCC) and in Pin Grid Array (PGA), both in Ceramic. The package designations are R and A respectively.

eg:	R 8207-8	LCC, 8 MHz DRAM Controller
eg:	A 8207-16	PGA, 16 MHz DRAM Controller

NOTE:
The pin-out of the PGA is the same as the socketed pinout of the LCC.

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 (Note 2) 2.5W

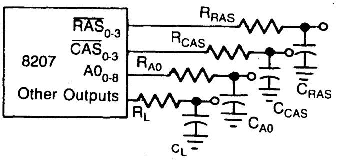
******Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.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

D.C. CHARACTERISTICS V_{CC} = 5.0V ±10% for 8207-10, 8207-8; T_A = 0°C to 70°C; V_{SS} = GND; V_{CC} = 5.0V ±5% for 8207-16 (Note 2)

Symbol	Parameter	Min	Max	Units	Comments
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	(Note 1)
V_{OH}	Output High Voltage	2.4		V	(Note 1)
V_{ROL}	RAM Output Low Voltage		0.45	V	(Note 1)
V_{ROH}	RAM Output High Voltage	2.6		V	(Note 1)
I_{CC}	Supply Current		455	mA	T_A = 0°C
I_{LI}	Input Leakage Current		±10	μA	0V ≤ V_{IN} ≤ V_{CC}
V_{CL}	Clock Input Low Voltage	−0.5	±0.6	V	
V_{CH}	Clock Input High Voltage	3.8	V_{CC} + 0.5	V	
C_{IN}	Input Capacitance		20	pF	fc = 1 MHz(2)

NOTE:
1. I_{OL} = 5 mA and I_{OH} = −0.2 mA (Typically I_{OL} = 10 mA and I_{OH} = −0.88 mA). WE: I_{OL} = 8 mA.
2. Sampled, not 100% tested.

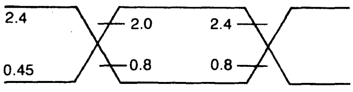
A.C. TESTING LOAD CIRCUIT(2)



210463-26

R_{RAS} = 39Ω	C_{RAS} = 150 pF
R_{CAS} = 39Ω	C_{CAS} = 150 pF
R_{A0} = 22Ω	C_{A0} = 380 pF
R_L = 39Ω	C_L = 100 pF

A.C. TESTING INPUT, OUTPUT WAVEFORM



210463-27

A.C. Testing inputs (except clock) are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0" (clock is driven at 4.0V and 0.45V for Logic "1" and "0" respectively). Timing measurements are made at 2.0V, 2.4V for Logic "1" and 0.8V for Logic "0".

A.C. CHARACTERISTICS

V_{CC} = 5V ±10% for 8207-8; T_A = 0°C to 70°C; V_{CC} = +5V ±5% for 8207-16

Measurements made with respect to RAS_{0-3}, CAS_{0-3}, AO_{0-8}, are a +2.4V and 0.8V. All other pins are measured at 2.0V and 0.8V. All times are ns unless otherwise indicated. Testing done with specified test load.

Ref	Symbol	Parameter	8207-16, -8		8207-10		Units	Notes
			Min	Max	Min	Max		
CLOCK AND PROGRAMMING								
—	tF	Clock Fall Time		10		10	ns	3
—	tR	Clock Rise Time		10		10	ns	3
1	TCLCL	Clock Period 8207-16 8207-10 8207-8	62.5 125	200 500	 100 	 250 	ns ns ns	1 2 2
2	TCL	Clock Low Time 8207-16 8207-10 8207-8	15 TCLCL/2−12	180	 TCLCL/2−12 		ns ns ns	1 2 2
3	TCH	Clock High Time 8207-16 8207-10 8207-8	20 TCLCL/3−3	180	 TCLCL/3−3 		ns ns ns	1 2 2
4	TRTVCL	Reset to CLK ↓ Setup	20		40		ns	4
5	TRTH	Reset Pulse Width	4TCLCL		4TCLCL		ns	
6	TPGVRTL	PCTL, PDI, RFRQ to RESET ↓ Setup	125		125		ns	5
7	TRTLPGX	PCTL,RFRQ to RESET ↓ Hold	10		10		ns	
8	TCLPC	PCLK from CLK ↓ Delay		45		45	ns	
9	TPDVCL	PDin to CLK ↓ Setup	60		60		ns	
10	TCLPDX	PDin to CLK ↓ Hold	40		40		ns	6
RAM WARM-UP AND INITIALIZATION								
64	TCLWZL	$\overline{WZ}$ from CLK ↓ Delay		40		40	ns	7
SYNCHRONOUS μP PORT INTERFACE								
11	TPEVCL	$\overline{PE}$ to CLK ↓ Setup	27		27		ns	2
12	TKVCL	$\overline{RD}$, $\overline{WR}$, $\overline{PE}$, PCTL to CLK ↓ Setup	20				ns	1
13	TCLKX	$\overline{RD}$, $\overline{WR}$, $\overline{PE}$, PCTL to CLK ↓ Hold	0		0		ns	
14	TKVCH	$\overline{RD}$, $\overline{WR}$, PCTL to CLK ↑ Setup	20		20		ns	2

A.C. CHARACTERISTICS (Continued)

V_{CC} = 5V ±10% for 8207-8; T_A = 0°C to 70°C; V_{CC} = +5V ±5% for 8207-16

Measurements made with respect to RAS_{0-3}, CAS_{0-3}, AO_{0-8}, are a +2.4V and 0.8V. All other pins are measured at 2.0V and 0.8V. All times are ns unless otherwise indicated. Testing done with specified test load.

Ref	Symbol	Parameter	8207-16, -8		8207-10		Units	Notes
			Min	Max	Min	Max		
ASYNCHRONOUS µP PORT INTERFACE								
15	TRWVCL	$\overline{\text{RD}}$, $\overline{\text{WR}}$ to CLK ↓ Setup	20		20		ns	8, 9
16	TRWL	$\overline{\text{RD}}$, $\overline{\text{WR}}$ Pulse Width	2TCLCL+30		2TCLCL+30		ns	
17	TRWLPEV	$\overline{\text{PE}}$ from $\overline{\text{RD}}$, CFS=1 $\overline{\text{WR}}$ ↓ Delay CFS=0		TCLCL−20 TCLCL−30		 TCLCL−20	ns ns	1 2
18	TRWLPEX	$\overline{\text{PE}}$ to $\overline{\text{RD}}$, $\overline{\text{WR}}$ ↓ Hold	2TCLCL+30		2TCLCL+30		ns	
19	TRWLPTV	PCTL from $\overline{\text{RD}}$, $\overline{\text{WR}}$ ↓ Delay		TCLCL−30		TCLCL−30	ns	2
20	TRWLPTX	PCTL to $\overline{\text{RD}}$, $\overline{\text{WR}}$ ↓ Hold	2TCLCL+30		2TCLCL+30		ns	2
21	TRWLPTV	PCTL from $\overline{\text{RD}}$, $\overline{\text{WR}}$ ↓ Delay		2TCLCL−20		2TCLCL−30	ns	1
22	TRWLPTX	PCTL to $\overline{\text{RD}}$, $\overline{\text{WR}}$ ↓ Hold	3TCLCL+30		3TCLCL+40		ns	1
RAM INTERFACE								
23	TAVCL	AL, AH, BS to CLK ↓ Setup	35+tASR		35+tASR		ns	10
24	TCLAX	AL, AH, BS to CLK ↓ Hold	0		0		ns	
25	TCLLN	LEN from CLK ↓ Delay		35			ns	1
26	TCLRSL	$\overline{\text{RAS}}$ ↓ from CLK ↓ Delay		35		35	ns	
27	TRCD	$\overline{\text{RAS}}$ to $\overline{\text{CAS}}$ CFS=1 Delay CFS=0	TCLCL−25 TCLCL/2−25		 25		ns ns ns	1, 14 11, 14
28	TCLRSH	$\overline{\text{RAS}}$ ↑ from CLK ↓ Delay		50		50	ns	
29	TRAH	Row A0 to CFS=1 $\overline{\text{RAS}}$ Hold CFS=0	TCLCL/2−11 TCLCL/4−11		 18		ns ns	1, 13, 15 11, 15
30	TASR	Row A0 to $\overline{\text{RAS}}$ Setup	0		0			10, 18
31	TASC	Column A0 to CFS=1 $\overline{\text{CAS}}$ ↓ Setup CFS=0	0 5		 5		ns ns	13, 19, 20 13, 19, 20

A.C. CHARACTERISTICS (Continued)

V_{CC} = 5V ±10% for 8207-8; T_A = 0°C to 70°C; V_{CC} = +5V ±5% for 8207-16

Measurements made with respect to RAS_{0-3}, CAS_{0-3}, AO_{0-8}, are a +2.4V and 0.8V. All other pins are measured at 2.0V and 0.8V. All times are ns unless otherwise indicated. Testing done with specified test load.

Ref	Symbol	Parameter	8207-16, -8		8207-10		Units	Notes
			Min	Max	Min	Max		
RAM INTERFACE (Continued)								
32	TCAH	Column A0 to CAS Hold	(See DRAM Interface Tables)					21
33	TCLCSL	CAS ↓ from CLK ↓ Delay	TCLCL/4+30	TCLCL/1.8+53	TCLCL/4+30	100	ns	11, 12
34	TCLCSL	CAS ↓ from CLK ↓ Delay		35		40	ns	1
35	TCLCSH	CAS ↑ from CLK ↓ Delay		50		50	ns	
36	TCLW	WE from CLK ↓ Delay		35		35	ns	
37	TCLTKL	XACK ↓ from CLK ↓ Delay		35		35	ns	
38	TRWLTKH	XACK ↑ from RD ↑, WR ↑ Delay		50		50	ns	
39	TCLAKL	AACK ↓ from CLK ↓ Delay		35		35	ns	
40	TCLAKH	AACK ↑ from CLK ↓ Delay		50		50	ns	
41	TCLDL	DBM from CLK ↓ Delay		35		35	ns	
ECC INTERFACE								
42	TWRLFV	FWR from CFS=1 WR ↓ Delay CFS=0		2TCLCL−40 TCLCL+TCL−40		 100	ns ns	1, 22 2, 22
43	TFVCL	FWR to CLK ↓ Setup	40		30		ns	23
44	TCLFX	FWR to CLK ↓ Hold	0		0		ns	24
45	TEVCL	ERROR to CLK ↓ Setup	20		20		ns	25, 26
46	TCLEX	ERROR to CLK ↓ Hold	0		0		ns	
47	TCLRL	R/W ↓ from CLK ↓ Delay		40		40	ns	
48	TCLRH	R/W ↑ from CLK ↓ Delay		50		50	ns	
49	TCEVCL	CE to CLK ↓ Setup	20		20		ns	25, 27
50	TCLCEX	CE to CLK ↓ Hold	0		0		ns	
51	TCLES	ESTB from CLK ↓ Delay		35		45	ns	

A.C. CHARACTERISTICS (Continued)

V_{CC} = 5V ±10% for 8207-8; T_A = 0°C to 70°C; V_{CC} = +5V ±5% for 8207-16

Measurements made with respect to RAS_{0-3}, CAS_{0-3}, AO_{0-8}, are a +2.4V and 0.8V. All other pins are measured at 2.0V and 0.8V. All times are ns unless otherwise indicated. Testing done with specified test load.

Ref	Symbol	Parameter	8207-16, -8		8207-10		Units	Notes
			Min	Max	Min	Max		
PORT SWITCHING AND LOCK								
52	TCLMV	MUX from CLK ↓ Delay		45		45	ns	
53	TCLPNV	PSEN from CLK ↓ Delay	TCL	TCL+35	TCL	TCL+35	ns	28
54	TCLPSV	PSEL from CLK ↓		35		35	ns	
55	TLKVCL	LOCK to CLK ↓ Setup	30		30		ns	30, 31
56	TCLLKX	LOCK to CLK ↓ Hold	10		10		ns	30, 31
57	TRWLLKV	LOCK from $\overline{RD}$ ↓, $\overline{WR}$ ↓ Delay		2TCLCL−30		2TCLCL−30	ns	31, 32
58	TRWHLKX	LOCK to $\overline{RD}$ ↓, $\overline{WR}$ ↓ Hold	3TCLCL+30		3TCLCL+30		ns	31, 32
REFRESH REQUEST								
59	TRFVCL	RFRQ to CLK ↓ Setup	20		20		ns	
60	TCLRFX	RFRQ to CLK ↓ Hold	10		10		ns	
61	TFRFH	Failsafe RFRQ Pulse Width	TCLCL+30		TCLCL+30		ns	33
62	TRFXCL	Single RFRQ Inactive to CLK ↓ Setup	20		20		ns	34
63	TBRFH	Burst RFRQ Pulse Width	2TCLCL+30		2TCLCL+30		ns	33

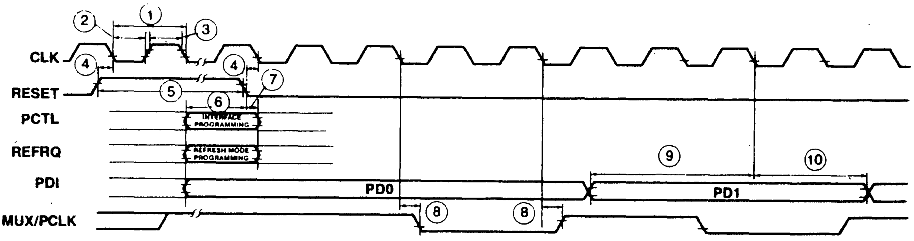
NOTES:

1. Specification when programmed in the Fast Cycle processor mode (iAPX 286 mode).
2. Specification when programmed in the Slow Cycle processor mode (iAPX 186 mode).
3. tR and tF are referenced from the 3.5V and 1.0V levels.
4. RESET is internally synchronized to CLK. Hence a set-up time is required only to guarantee its recognition at a particular clock edge.
5. The first programming bit (PD0) is also sampled by RESET going low.
6. TCLPDX is guaranteed if programming data is shifted using PCLK.
7. WZ is issued only in ECC mode.
8. TRWVCL is not required for an asynchronous command except to guarantee its recognition at a particular clock edge.
9. Valid when programmed in either Fast or Slow Cycle mode.
10. tASR is a user specified parameter and its value should be added accordingly to TAVCL.
11. When programmed in Slow Cycle mode and 125 ns ≤ TCLCL < 200 ns.
12. When programmed in Slow Cycle mode and 200 ns ≤ TCLCL.
13. Specification for Test Load conditions.
14. tRCD (actual) = tRCD (specification) + 0.06 (ΔC_{RAS}) − 0.6 (ΔC_{CAS}) where ΔC = C (test load) − C (actual) in pF (These are first order approximations).
15. tRAH (actual) = tRAH (specification) + 0.06 (ΔC_{RAS}) − 0.022 (ΔC_{A0}) where ΔC = C (test load) − C (actual) in pF. (These are first order approximations.)
18. tASR (actual) = tASR (specification) + 0.06 (ΔC_{A0}) − 0.025 (ΔC_{RAS}) where ΔC = C (test load) − C (actual) in pF. (These are first order approximations.)
19. tASC (actual) = tASC (specification) + 0.06 (ΔC_{A0}) − 0.025 (ΔC_{CAS}) where ΔC = C (test load) − C (actual) in pF. (These are first order approximations.)

20. tASC is a function of clock frequency and thus varies with changes in frequency. A minimum value is specified.
21. See 8207 DRAM Interface Tables 14–18.
22. TWRLFV is defined for both synchronous and asynchronous $\overline{FWR}$. In systems in which $\overline{FWR}$ is decoded directly from the address inputs to the 8207, TCLFV is automatically guaranteed by TCLAV.
23. TFVCL is defined for synchronous $\overline{FWR}$.
24. TCLFV is defined for both synchronous and asynchronous $\overline{FWR}$. In systems in which $\overline{FWR}$ is decoded directly from the address inputs to the 8207 TCLFV is automatically guaranteed by TCLAV.
25. $\overline{ERROR}$ and CE are set-up to CLK ↓ in fast cycle mode and CLK ↑ in slow cycle mode.
26. $\overline{ERROR}$ is set-up to the same edge as R/$\overline{W}$ is referenced to, in RMW cycles.
27. CE is set-up to the same edge as WE is referenced to in RMW cycles.
28. Specification when TCL < 25 ns.
29. Synchronous operation only. Must arrive by the second clock falling edge after the clock edge which recognizes the command in order to be effective.
30. LOCK must be held active for the entire period the opposite port must be locked out. One clock after the release of LOCK the opposite port will be able to obtain access to memory.
31. Asynchronous mode only. In this mode a synchronizer stage is used internally in the 8207 to synchronize up LOCK. TRWLLKV and TRWHLKX are only required for guaranteeing that LOCK will be recognized for the requesting port, but these parameters are not required for correct 8207 operation.
32. TFRFH and TBRFH pertain to asynchronous operation only.
33. Single RFRQ cannot be supplied asynchronously.

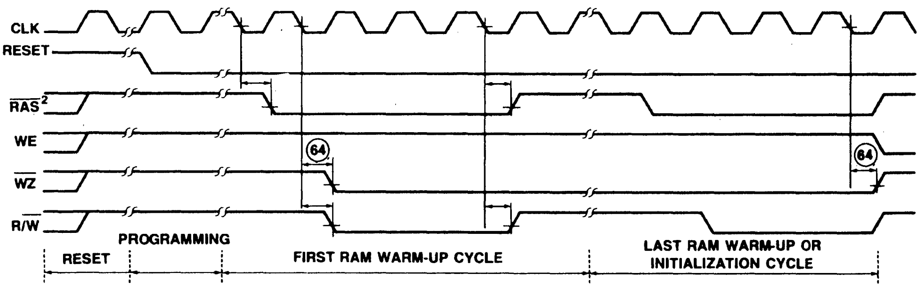
WAVEFORMS

CLOCK AND PROGRAMMING TIMINGS



210463–28

RAM WARM-UP AND MEMORY INITIALIZATION CYCLES

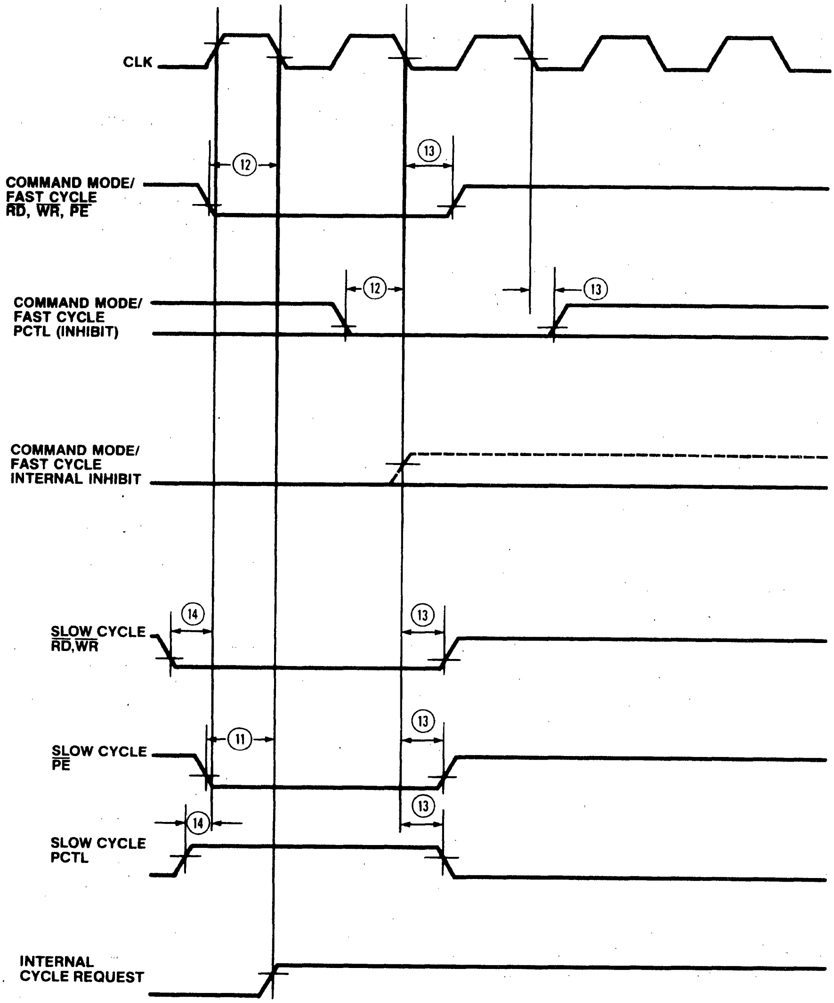


210463–29

NOTES:
1. When in non-ECC mode or in ECC mode with the TM2 programming bit on, there are no initialization cycles, when in ECC mode with TM2 off, the dummy cycles are followed by initialization cycles.
2. The present example assumes a RAS four clocks long.

WAVEFORMS (Continued)

SYNCHRONOUS PORT INTERFACE

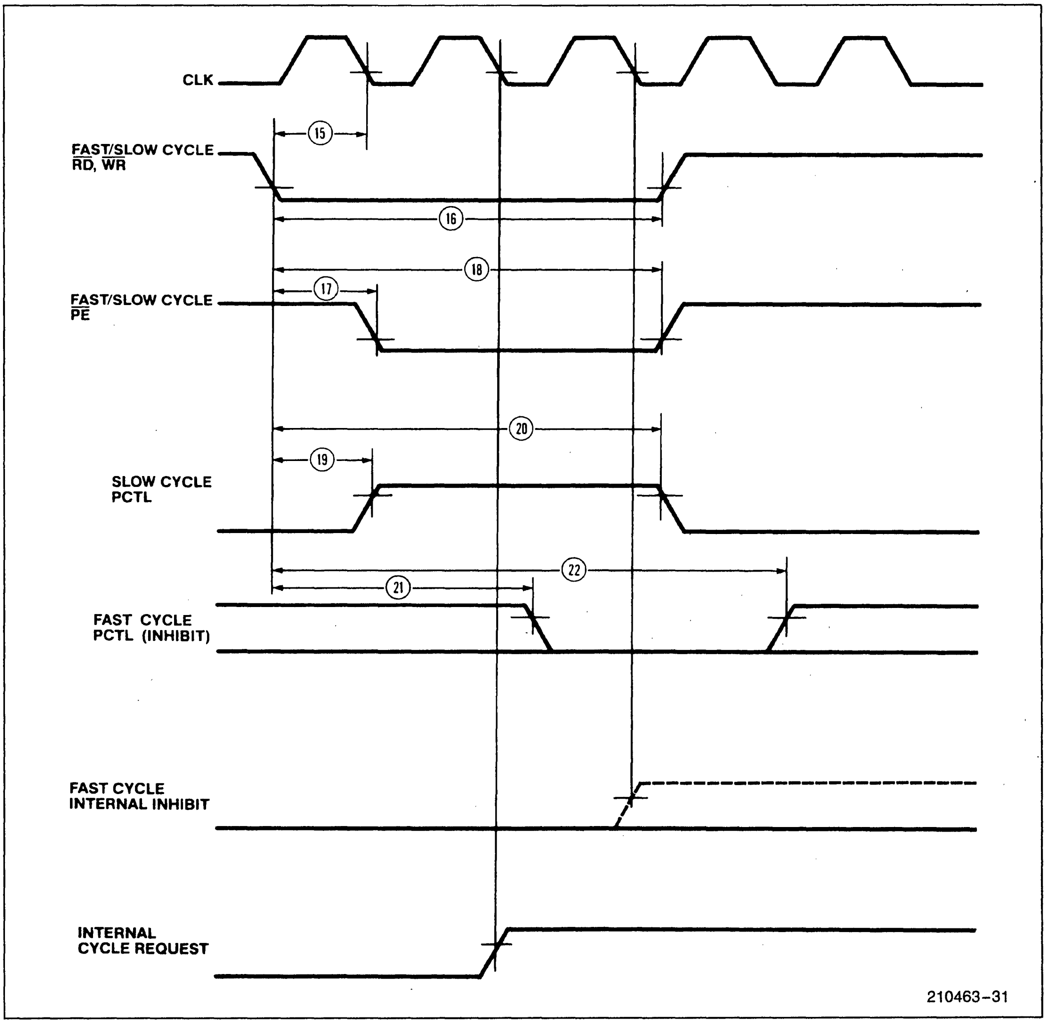

CLK
COMMAND MODE/ FAST CYCLE RD, WR, PE
COMMAND MODE/ FAST CYCLE PCTL (INHIBIT)
COMMAND MODE/ FAST CYCLE INTERNAL INHIBIT
SLOW CYCLE RD,WR
SLOW CYCLE PE
SLOW CYCLE PCTL
INTERNAL CYCLE REQUEST
12
13
12
13
14
13
11
13
14
13

210463–30

NOTE:
Actual transitions are programmable. Refer to Tables 12 and 13.

WAVEFORMS (Continued)

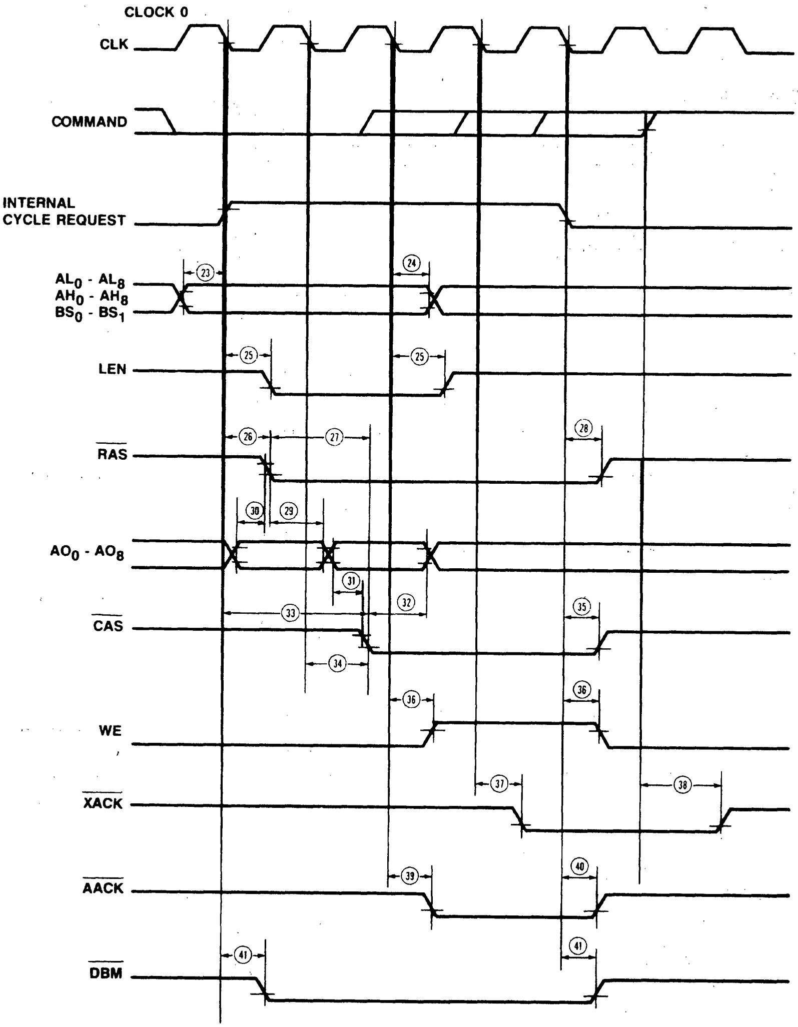
ASYNCHRONOUS PORT INTERFACE



210463-31

WAVEFORMS (Continued)

RAM INTERFACE TIMING
ECC AND NON-ECC MODE

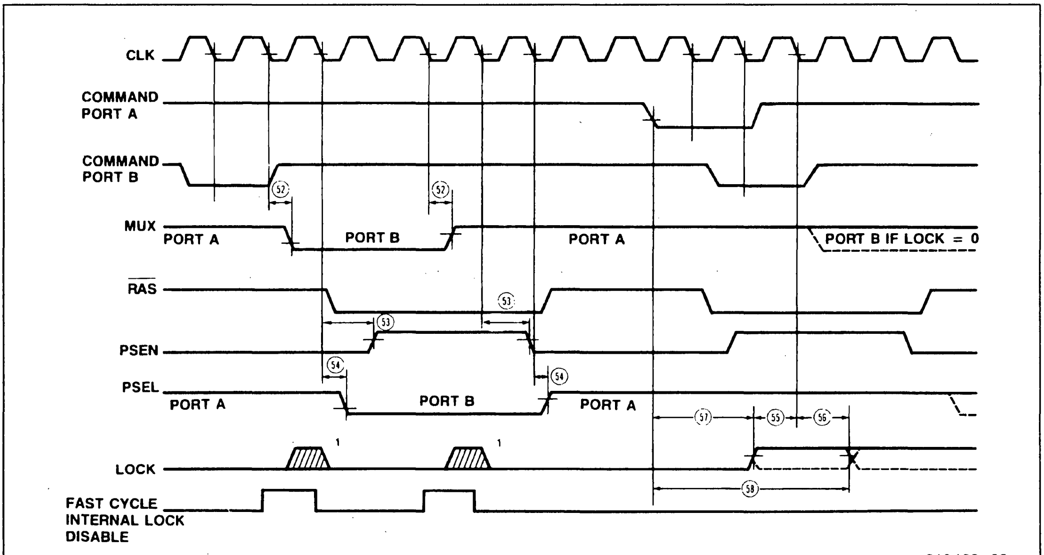


210463-32

NOTE:
Actual transitions are programmable. Refer to Tables 12 and 13.

WAVEFORMS (Continued)

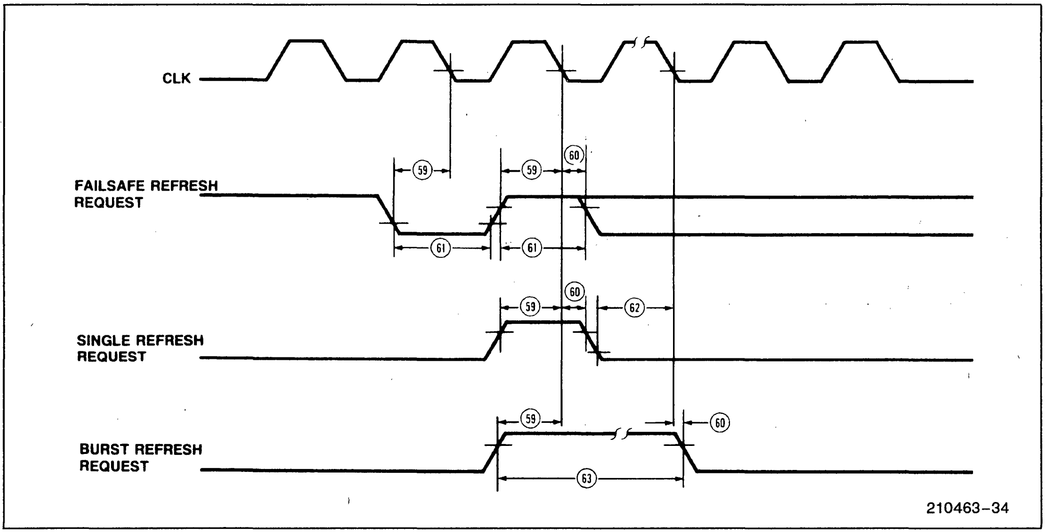
PORT SWITCHING AND LOCK TIMING



210463-33

NOTE:
Transients during MUX switching.

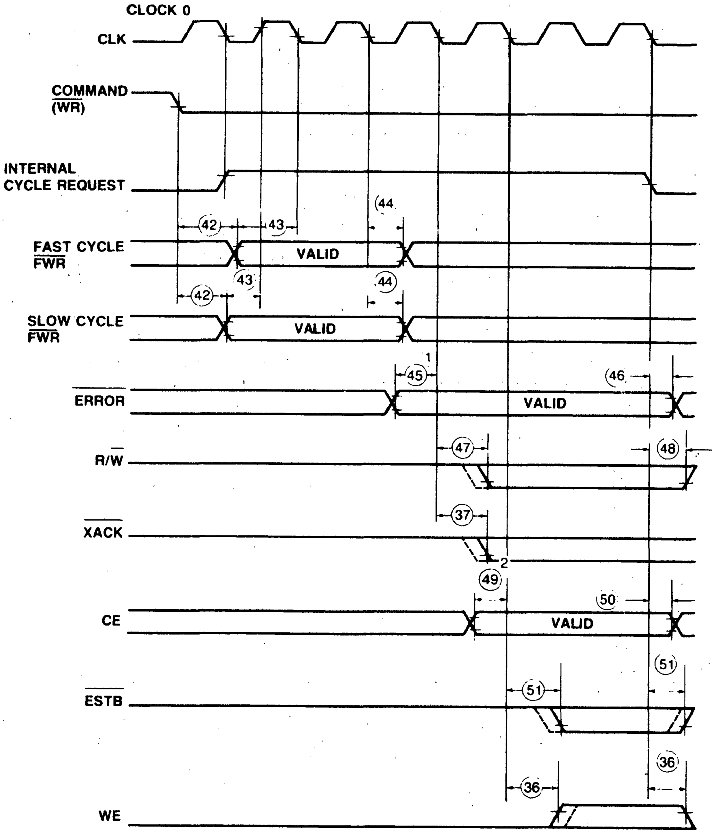
REFRESH REQUEST TIMING



210463-34

WAVEFORMS (Continued)

ECC INTERFACE TIMING



210463-35

NOTES:

1. This parameter is set-up to the falling edge of clock, as shown, for fast cycle configurations. It is set-up to the rising edge of clock if in slow cycle configurations. Table 13A shows which clock and clock edge these signals are set-up in the R/W L column.
2. CE is set-up to the same edge as WE is referenced to in RMW cycles.

CONFIGURATION TIMING CHARTS

The timing charts that follow are based on 8 basic system configurations where the 8207 operates.

Tables 10 and 11 give a description of non-ECC and ECC system configurations based on the 8207's PD0, PD3, PD4, PD10 and PD11 programming bits.

Table 10. Non-ECC System Configurations

Non-ECC Mode: PD0 = 0

Timing Conf.	$\overline{\text{CFS}}$(PD3)	$\overline{\text{RFS}}$(PD4)	EXT(PD10)	$\overline{\text{FFS}}$(PD11)
C_0	iAPX286(0)	Fast RAM(0)	Not EXT(0)	12 MHz(1)
C_0	iAPX286(0)	Fast RAM(0)	EXT(1)	12 MHz(1)
C_0	iAPX286(0)	Slow RAM(1)	Not EXT(0)	12 MHz(1)
C_0	iAPX286(0)	Slow RAM(1)	EXT(1)	12 MHz(1)
C_0	iAPX286(0)	Fast RAM(0)	Not EXT(0)	16 MHz(0)
C_1	iAPX286(0)	Slow RAM(1)	Not EXT(0)	16 MHz(0)
C_1	iAPX286(0)	Fast RAM(0)	EXT(1)	16 MHz(0)
C_2	iAPX286(0)	Slow RAM(1)	EXT(1)	16 MHz(0)
C_3	iAPX186(1)	Fast RAM(0)	Not EXT(0)	10, 8 MHz(0)
C_3	iAPX186(1)	Slow RAM(1)	Not EXT(0)	10, 8 MHz(0)
C_3	iAPX186(1)	Fast RAM(0)	EXT(1)	10, 8 MHz(0)
C_3	iAPX186(1)	Fast RAM(0)	Not EXT(0)	6 MHz(1)
C_3	iAPX186(1)	Fast RAM(0)	EXT(1)	6 MHz(1)
C_3	iAPX186(1)	Slow RAM(1)	Not EXT(0)	6 MHz(1)
C_3	iAPX186(1)	Slow RAM(1)	EXT(1)	6 MHz(1)
C_4	iAPX186(1)	Slow RAM(1)	EXT(1)	10, 8 MHz(0)

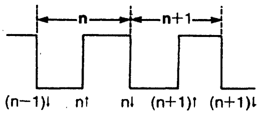
Table 11. ECC System Configurations

ECC Mode: PD0 = 1

Timing Conf.	CFS(PD3)	RFS(PD4)	$\overline{\text{EXT}}$(PD10)	FFS(PD11)
C_0	iAPX286(1)	Slow RAM(0)	M/S EDCU(0)	12 MHz(0)
C_0	iAPX286(1)	Slow RAM(0)	M EDCU(1)	12 MHz(0)
C_0	iAPX286(1)	Fast RAM(1)	M/S EDCU(0)	12 MHz(0)
C_0	iAPX286(1)	Fast RAM(1)	M EDCU(1)	12 MHz(0)
C_0	iAPX286(1)	Fast RAM(1)	M EDCU(1)	16 MHz(1)
C_1	iAPX286(1)	Slow RAM(0)	M EDCU(1)	16 MHz(1)
C_2	iAPX286(1)	Fast RAM(1)	M/S EDCU(0)	16 MHz(1)
C_3	iAPX286(1)	Slow RAM(0)	M/S EDCU(0)	16 MHz(1)
C_4	iAPX186(0)	Slow RAM(0)	M/S EDCU(0)	6 MHz(0)
C_4	iAPX186(0)	Fast RAM(1)	M/S EDCU(0)	6 MHz(0)
C_4	iAPX186(0)	Slow RAM(0)	M EDCU(1)	10, 8 MHz(1)
C_4	iAPX186(0)	Fast RAM(1)	M EDCU(1)	10, 8 MHz(1)
C_5	iAPX186(0)	Slow RAM(0)	M/S EDCU(0)	10, 8 MHz(1)
C_5	iAPX186(0)	Fast RAM(1)	M/S EDCU(0)	10, 8 MHz(1)
C_6	iAPX186(0)	Slow RAM(0)	M EDCU(1)	6 MHz(0)
C_6	iAPX186(0)	Fast RAM(1)	M EDCU(1)	6 MHz(0)

Using the Timing Charts

The notation used to indicate which clock edge triggers an output transition is "n ↑" or "n ↓", where "n" is the number of clock periods that have passed since clock 0, the reference clock, and " ↑ " refers to rising edge and " ↓ " to falling edge. A clock period is defined as the interval from a clock falling edge to the following falling edge. Clock edges are defined as shown below.



210463-36

The clock edges which trigger transitions on each 8207 output are tabulated in Table 12 for non-ECC mode and Table 13 for ECC mode. "H" refers to the high-going transition, and "L" to low-going transition; "V" refers to valid, and "$\overline{\text{V}}$" to non-valid.

Clock 0 is defined as the clock in which the 8207 begins a memory cycle, either as a result of a port request which has just arrived, or of a port request which was stored previously but could not be serviced at the time of its arrival because the 8207 was performing another memory cycle. Clock 0 may be identified externally by the leading edge of RAS, which is always triggered on 0 ↓.

Notes for interpreting the timing charts:

1. **PSEL - valid** is given as the latest time it can occur. It is entirely possible for PSEL to become valid before the time given in a refresh cycle. PSEL can switch as defined in the chart, but it has no bearing on the refresh cycle itself, but only on a subsequent cycle for one of the external ports.
2. **LEN - low** is given as the latest time it can occur. LEN is only activated by port A configured in Fast Cycle iAPX286 mode, and thus it is not activated by a refresh cycle, although it may be activated by port A during a refresh cycle.
3. **ADDRESS - col** is the time column address becomes valid.
4. In non-ECC mode the $\overline{\text{CAS}}$, $\overline{\text{EAACK}}$, $\overline{\text{LAACK}}$ and $\overline{\text{XACK}}$ outputs are not issued during refresh.
5. In ECC mode there are really seven types of cycles: Read without error, read with error, full write, partial write without error, partial write with error, refresh without error, and refresh with error. These cycles may be derived from the timing chart as follows:
   A. Read without error: Use row marked 'RD, RF'.
   B. Read with error: Use row marked 'RMW', except for $\overline{\text{EAACK}}$ and $\overline{\text{LAACK}}$, which,should be taken from 'RD, RF'. If the error is uncorrectable. WE will not be issued.
   C. Full write: Use row marked 'WR'.
   D. Partial write without error: Use row marked 'RMW', except that $\overline{\text{DBM}}$ and $\overline{\text{ESTB}}$ will not be issued.
   E. Partial write with error: Use row marked 'RMW', except that $\overline{\text{DBM}}$ will not be issued. If the error is uncorrectable, WE will not be issued.
   F. Refresh without error: Use row marked 'RD, RF', except that $\overline{\text{ESTB}}$, $\overline{\text{EAACK}}$, $\overline{\text{LAACK}}$, and $\overline{\text{XACK}}$ will not be issued.
   G. Refresh with error: Use row marked 'RMW', except that $\overline{\text{EAACK}}$, $\overline{\text{LAACK}}$, $\overline{\text{ESTB}}$, and $\overline{\text{XACK}}$ will not be issued. If the error is uncorrectable WE will not be issued.
6. **XACK - high** is reset asynchronously by command going inactive and not by a clock edge.
7. **MUX - valid** is given as the latest time it can occur.

Table 12A. Timing Chart—Non-ECC Mode

		PSEN		PSEL		$\overline{\text{DBM}}$		LEN		$\overline{\text{RAS}}$		$\overline{\text{CAS}}$		WE	
C_n	Cycle	H	L	V	$\overline{V}$	L	H	L	H	L	H	L	H	H	L
C_0	RD, RF	0↓	3↓	0↓	4↓	0↓	4↓	0↓	2↓	0↓	3↓	1↓	4↓		
	WR	0↓	4↓	0↓	5↓			0↓	2↓	0↓	5↓	1↓	5↓	2↓	5↓
C_1	RD, RF	0↓	5↓	0↓	6↓	0↓	6↓	0↓	2↓	0↓	4↓	1↓	6↓		
	WR	0↓	4↓	0↓	5↓			0↓	2↓	0↓	5↓	1↓	5↓	2↓	5↓
C_2	RD, RF	0↓	5↓	0↓	6↓	0↓	6↓	0↓	2↓	0↓	4↓	1↓	6↓		
	WR	0↓	4↓	0↓	5↓			0↓	2↓	0↓	5↓	1↓	5↓	2↓	5↓
C_3	RD, RF	0↓	2↓	0↓	3↓	0↓	3↓			0↓	3↓	0↓	3↓		
	WR	0↓	3↓	0↓	4↓					0↓	4↓	0↓	4↓	2↑	4↓
C_4	RD, RF	0↓	3↓	0↓	4↓	0↓	4↓			0↓	4↓	0↓	4↓		
	WR	0↓	3↓	0↓	4↓					0↓	4↓	0↓	4↓	2↑	4↓

Table 12B. Timing Chart—Non-ECC Mode

		Col Addr		$\overline{\text{EAACK}}$		$\overline{\text{LAACK}}$		$\overline{\text{XACK}}$		MUX	
C_n	Cycle	V	$\overline{V}$	L	H	L	H	L	H	V	$\overline{V}$
C_0	RD, RF	0↓	2↓	1↓	4↓	2↓	5↓	3↓	$\overline{\text{RD}}$	−2↓	2↓
	WR	0↓	2↓	1↓	4↓	1↓	4↓	3↓	$\overline{\text{WR}}$	−2↓	2↓
C_1	RD, RF	0↓	3↓	2↓	5↓	2↓	5↓	4↓	$\overline{\text{RD}}$	−2↓	2↓
	WR	0↓	3↓	1↓	4↓	1↓	4↓	3↓	$\overline{\text{WR}}$	−2↓	2↓
C_2	RD, RF	0↓	3↓	2↓	5↓	3↓	6↓	4↓	$\overline{\text{RD}}$	−2↓	2↓
	WR	0↓	3↓	1↓	4↓	1↓	4↓	3↓	$\overline{\text{WR}}$	−2↓	2↓
C_3	RD, RF	0↓	2↓	0↓	2↓	1↓	3↓	2↓	$\overline{\text{RD}}$	−1↓	2↓
	WR	0↓	2↓	0↓	2↓	1↑	3↑	2↓	$\overline{\text{WR}}$	−1↓	2↓
C_4	RD, RF	0↓	2↓	1↓	3↓	1↓	3↓	3↑	$\overline{\text{RD}}$	−1↓	2↓
	WR	0↓	2↓	0↓	2↓	1↑	3↑	2↓	$\overline{\text{WR}}$	−1↓	2↓

Table 13A. Timing Chart—ECC Mode

		PSEN		PSEL		$\overline{DBM}$		$\overline{LEN}$		$\overline{RAS}$		$\overline{CAS}$		R/$\overline{W}$		WE	
C_n	Cycle	H	L	V	$\overline{V}$	L	H	L	H	L	H	L	H	L	H	H	L
C_0	RD, RF	0↓	5↓	0↓	6↓	0↓	6↓	0↓	2↓	0↓	4↓	1↓	6↓				
	WR	0↓	5↓	0↓	6↓			0↓	2↓	0↓	6↓	1↓	6↓	1↓	6↓	3↓	6↓
	RMW	0↓	8↓	0↓	9↓	0↓	9↓	0↓	2↓	0↓	9↓	1↓	9↓	4↓	9↓	6↓	9↓
C_1	RD, RF	0↓	5↓	0↓	6↓	0↓	6↓	0↓	2↓	0↓	4↓	1↓	6↓				
	WR	0↓	5↓	0↓	6↓			0↓	2↓	0↓	6↓	1↓	6↓	1↓	6↓	3↓	6↓
	RMW	0↓	8↓	0↓	9↓	0↓	9↓	0↓	2↓	0↓	9↓	1↓	9↓	4↓	9↓	6↓	9↓
C_2	RD, RF	0↓	6↓	0↓	7↓	0↓	7↓	0↓	2↓	0↓	5↓	1↓	7↓				
	WR	0↓	6↓	0↓	7↓			0↓	2↓	0↓	7↓	1↓	7↓	1↓	7↓	4↓	7↓
	RMW	0↓	10↓	0↓	11↓	0↓	11↓	0↓	2↓	0↓	11↓	1↓	11↓	5↓	11↓	8↓	11↓
C_3	RD, RF	0↓	6↓	0↓	7↓	0↓	7↓	0↓	2↓	0↓	5↓	1↓	7↓				
	WR	0↓	6↓	0↓	7↓			0↓	2↓	0↓	7↓	1↓	7↓	1↓	7↓	4↓	7↓
	RMW	0↓	10↓	0↓	11↓	0↓	11↓	0↓	2↓	0↓	11↓	1↓	11↓	5↓	11↓	8↓	11↓
C_4	RD, RF	0↓	3↓	0↓	4↓	0↓	4↓			0↓	3↓	0↓	4↓				
	WR	0↓	4↓	0↓	5↓					0↓	5↓	0↓	5↓	1↑	5↓	3↑	5↓
	RMW	0↓	6↓	0↓	7↓	0↓	7↓			0↓	7↓	0↓	7↓	3↑	7↓	5↑	7↓
C_5	RD, RF	0↓	3↓	0↓	4↓	0↓	4↓			0↓	3↓	0↓	4↓				
	WR	0↓	4↓	0↓	5↓					0↓	5↓	0↓	5↓	1↑	5↓	3↑	5↓
	RMW	0↓	6↓	0↓	7↓	0↓	7↓			0↓	7↓	0↓	7↓	3↑	7↓	5↑	7↓
C_6	RD, RF	0↓	3↓	0↓	4↓	0↓	4↓			0↓	3↓	0↓	4↓				
	WR	0↓	3↓	0↓	4↓					0↓	4↓	0↓	4↓	1↑	4↓	2↑	4↓
	RMW	0↓	4↓	0↓	5↓	0↓	5↓			0↓	5↓	0↓	5↓	2↑	5↓	3↑	5↓

Table 13B. Timing Chart—ECC Mode

		Col Addr		$\overline{\text{ESTB}}$		$\overline{\text{EAACK}}$		$\overline{\text{LAACK}}$		$\overline{\text{XACK}}$		$\overline{\text{MUX}}$	
C_n	Cycle	V	$\overline{V}$	L	H	L	H	L	H	L	H	V	$\overline{V}$
C_0	RD, RF	0↓	2↓			2↓	5↓	3↓	6↓	4↓	RD	−2↓	2↓
	WR	0↓	2↓			2↓	5↓	2↓	5↓	4↓	WR	−2↓	2↓
	RMW	0↓	2↓	6↓	8↓	5↓	8↓	5↓	8↓	7↓	WR	−2↓	2↓
C_1	RD, RF	0↓	3↓			3↓	6↓	3↓	6↓	4↓	RD	−2↓	2↓
	WR	0↓	3↓			2↓	5↓	2↓	5↓	4↓	WR	−2↓	2↓
	RMW	0↓	3↓	6↓	8↓	5↓	8↓	5↓	8↓	7↓	WR	−2↓	2↓
C_2	RD, RF	0↓	3↓			4↓	7↓	4↓	7↓	5↓	RD	−2↓	2↓
	WR	0↓	3↓			3↓	6↓	3↓	6↓	5↓	WR	−2↓	2↓
	RMW	0↓	3↓	8↓	10↓	7↓	10↓	7↓	10↓	9↓	WR	−2↓	2↓
C_3	RD, RF	0↓	3↓			4↓	7↓	5↓	8↓	5↓	RD	−2↓	2↓
	WR	0↓	3↓			3↓	6↓	3↓	6↓	5↓	WR	−2↓	2↓
	RMW	0↓	3↓	8↓	10↓	7↓	10↓	7↓	10↓	9↓	WR	−2↓	2↓
C_4	RD, RF	0↓	2↓			1↓	3↓	2↑	4↑	3↑	RD	−1↓	2↓
	WR	0↓	2↓			1↓	3↓	2↑	4↑	3↓	WR	−1↓	2↓
	RMW	0↓	2↓	5↑	6↑	3↓	5↓	4↑	6↑	5↓	WR	−1↓	2↓
C_5	RD, RF	0↓	2↓			2↓	4↓	3↑	5↑	3↑	RD	−1↓	2↓
	WR	0↓	2↓			1↓	3↓	2↑	4↑	3↓	WR	−1↓	2↓
	RMW	0↓	2↓	5↑	6↑	3↓	5↓	4↑	6↑	5↓	WR	−1↓	2↓
C_2	RD, RF	0↓	2↓			1↓	3↓	1↑	3↑	2↑	RD	−1↓	2↓
	WR	0↓	2↓			1↓	3↓	1↑	3↑	2↓	WR	−1↓	2↓
	RMW	0↓	2↓	3↑	4↑	1↓	3↓	2↑	4↑	3↓	WR	−1↓	2↓

8207—DRAM Interface Parameter Equations

Several DRAM parameters, but not all, are a direct function of 8207 timings, and the equations for these parameters are given in the following tables. The following is a list of those DRAM parameters which have NOT been included in the following tables, with an explanation for their exclusion.

READ, WRITE, READ-MODIFY-WRITE & REFRESH CYCLES

tRAC: response parameter.

tCAC: response parameter.

tREF: See "Refresh Period Options"

tCRP: must be met only if $\overline{CAS}$-only cycles, which do not occur with 8207, exist.

tRAH: See "A.C. Characteristics"

tRCD: See "A.C. Characteristics"

tASC: See "A.C. Characteristics"

tASR: See "A.C. Characteristics"

tOFF: response parameter.

READ & REFRESH CYCLES

tRCH: WE always goes active after $\overline{CAS}$ goes active, hence tRCH is guaranteed by tCPN.

WRITE CYCLE

tRC: guaranteed by tRWC.

tRAS: guaranteed by tRRW.

tCAS: guaranteed by tCRW.

tWCS: WE always activated after $\overline{CAS}$ is activated, except in memory initialization, hence tWCS is always negative (this is important for RMW only) except in memory initialization; in memory initialization tWCS is positive and has several clocks of margin.

tDS: system-dependent parameter.

tDH: system-dependent parameter.

tDHR: system-dependent parameter.

READ-MODIFY-WRITE CYCLE

tRWD: don't care in 8207 write cycles, but tabulated for 8207 RMW cycles.

tCWD: don't care in 8207 write cycles, but tabulated for 8207 RMW cycles.

Table 14. Non-ECC Mode—RD, RF Cycles

Parameter	Fast Cycle Configurations			Slow Cycle Configurations		Notes
	C_0	C_1	C_2	C_3	C_4	
tRP	3TCLCL−T26	4TCLCL−T26	4TCLCL−T26	2TCLCL−T26	2TCLCL−T26	1
tCPN	3TCLCL−T35	3TCLCL−T35	3TCLCL−T35	2.5TCLCL−T35	2.5TCLCL−T35	1
tRSH	2TCLCL−T34	3TCLCL−T34	3TCLCL−T34	3TCLCL−T34	4TCLCL−T34	1
tCSH	4TCLCL−T26	6TCLCL−T26	6TCLCL−T26	3TCLCL−T26	4TCLCL−T26	1
tCAH	TCLCL−T34	2TCLCL−T34	2TCLCL−T34	2TCLCL−T34	2TCLCL−T34	1
tAR	2TCLCL−T26	3TCLCL−T26	3TCLCL−T26	2TCLCL−T26	2TCLCL−T26	1
tT	3/30	3/30	3/30	3/30	3/30	2
tRC	6TCLCL	8TCLCL	8TCLCL	5TCLCL	6TCLCL	1
tRAS	3TCLCL−T26	4TCLCL−T26	4TCLCL−T26	3TCLCL−T26	4TCLCL−T26	1
tCAS	3TCLCL−T34	5TCLCL−T34	5TCLCL−T34	3TCLCL−T34	4TCLCL−T34	1
tRCS	2TCLCL−TCL −T36−TBUF	2TCLCL−TCL −T36−TBUF	2TCLCL−TCL −T36−TBUF	1.5TCLCL−TCL −T36−TBUF	1.5TCLCL−TCL −T36−TBUF	1

Table 15. Non-ECC Mode—WR Cycle

Parameter	Fast Cycle Configurations			Slow Cycle Configurations		Notes
	C_0	C_1	C_2	C_3	C_4	
tRP	3TCLCL−T26	3TCLCL−T26	3TCLCL−T26	2TCLCL−T26	2TCLCL−T26	1
tCPN	4TCLCL−T35	4TCLCL−T35	4TCLCL−T35	2.5TCLCL−T35	2.5TCLCL−T35	1
tRSH	4TCLCL−T34	4TCLCL−T34	4TCLCL−T34	4TCLCL−T34	4TCLCL−T34	1
tCSH	5TCLCL−T26	5TCLCL−T26	5TCLCL−T26	4TCLCL−T26	4TCLCL−T26	1
tCAH	TCLCL−T34	2TCLCL−T34	2TCLCL−T34	2TCLCL−T34	2TCLCL−T34	1
tAR	2TCLCL−T26	3TCLCL−T26	3TCLCL−T26	2TCLCL−T26	2TCLCL−T26	1
tT	3/30	3/30	3/30	3/30	3/30	2
tRWC	8TCLCL	8TCLCL	8TCLCL	6TCLCL	6TCLCL	1
tRRW	5TCLCL−T26	5TCLCL−T26	5TCLCL−T26	4TCLCL−T26	4TCLCL−T26	1
tCRW	4TCLCL−T34	4TCLCL−T34	4TCLCL−T34	4TCLCL−T34	4TCLCL−T34	1
tWCH	3TCLCL+TCL −T34	3TCLCL+TCL −T34	3TCLCL+TCL −T34	3TCLCL+TCL −T34	3TCLCL+TCL −T34	1, 3
tWCR	4TCLCL+TCL −T26	4TCLCL+TCL −T26	4TCLCL+TCL −T26	3TCLCL+TCL −T26	3TCLCL+TCL −T26	1, 3
tWP	2TCLCL+TCL −T36−TBUF	2TCLCL+TCL −T36−TBUF	2TCLCL+TCL −T36−TBUF	2TCLCL−T36 −TBUF	2TCLCL−T36 −TBUF	1
tRWL	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	3TCLCL−TCL −T36−TBUF	3TCLCL−TCL −T36−TBUF	1
tCWL	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	3TCLCL−TCL −T36−TBUF	3TCLCL−TCL −T36−TBUF	1

Table 16A. ECC Mode—RD, RF Cycles

Parameter	Fast Cycle Mode				Notes
	C_0	C_1	C_2	C_3	
tRP	4TCLCL−T26	4TCLCL−T26	4TCLCL−T26	4TCLCL−T26	1
tCPN	3TCLCL−T35	3TCLCL−T35	3TCLCL−T35	3TCLCL−T35	1
tRSH	3TCLCL−T34	3TCLCL−T34	4TCLCL−T34	4TCLCL−T34	1
tCSH	6TCLCL−T26	6TCLCL−T26	7TCLCL−T26	7TCLCL−T26	1
tCAH	TCLCL−T34	2TCLCL−T34	2TCLCL−T34	2TCLCL−T34	1
tAR	2TCLCL−T26	3TCLCL−T26	3TCLCL−T26	3TCLCL−T26	1
tT	3/30	3/30	3/30	3/30	2
tRC	8TCLCL	8TCLCL	9TCLCL	9TCLCL	1
tRAS	4TCLCL−T26	4TCLCL−T26	5TCLCL−T26	5TCLCL−T26	1
tCAS	5TCLCL−T34	5TCLCL−T34	6TCLCL−T34	6TCLCL−T34	1
tRCS	TCLCL−T36 −TBUF	TCLCL−T36 −TBUF	TCLCL−T36 −TBUF	TCLCL−T36 −TBUF	1

Table 16B. ECC Mode—RD, RF Cycles

Parameter	Slow Cycle Mode			Notes
	C_4	C_5	C_6	
tRP	2TCLCL−T26	2TCLCL−T26	2TCLCL−T26	1
tCPN	1.5TCLCL−T35	1.5TCLCL−T35	1.5TCLCL−T35	1
tRSH	3TCLCL−T34	3TCLCL−T34	3TCLCL−T34	1
tCSH	4TCLCL−T26	4TCLCL−T26	4TCLCL−T26	1
tCAH	2TCLCL−T34	2TCLCL−T34	2TCLCL−T34	1
tAR	2TCLCL−T26	2TCLCL−T26	2TCLCL−T26	1
tT	3/30	3/30	3/30	2
tRC	5TCLCL	5TCLCL	5TCLCL	1
tRAS	3TCLCL−T26	3TCLCL−T26	3TCLCL−T26	1
tCAS	4TCLCL−T34	4TCLCL−T34	4TCLCL−T34	1
tRCS	0.5TCLCL−T36 −TBUF	0.5TCLCL−T36 −TBUF	0.5TCLCL−T36 −TBUF	1

Table 17A. ECC Mode—WR Cycle

Parameter	Fast Cycle Mode				Notes
	C_0	C_1	C_2	C_3	
tRP	3TCLCL−T26	3TCLCL−T26	3TCLCL−T26	3TCLCL−T26	1
tCPN	4TCLCL−T35	4TCLCL−T35	4TCLCL−T35	4TCLCL−T35	1
tRSH	5TCLCL−T34	5TCLCL−T34	6TCLCL−T34	6TCLCL−T34	1
tCSH	6TCLCL−T26	6TCLCL−T26	7TCLCL−T26	7TCLCL−T26	1
tCAH	TCLCL−T34	2TCLCL−T34	2TCLCL−T34	2TCLCL−T34	1
tAR	2TCLCL−T26	3TCLCL−T26	3TCLCL−T26	3TCLCL−T26	1
tT	3/30	3/30	3/30	3/30	2
tRWC	9TCLCL	9TCLCL	10TCLCL	10TCLCL	1
tRRW	6TCLCL−T26	6TCLCL−T26	7TCLCL−T26	7TCLCL−T26	1
tCRW	5TCLCL−T34	5TCLCL−T34	6TCLCL−T34	6TCLCL−T34	1
tWCH	5TCLCL−T34	5TCLCL−T34	6TCLCL−T34	6TCLCL−T34	1, 4
tWCR	6TCLCL−T26	6TCLCL−T26	7TCLCL−T26	7TCLCL−T26	1, 4
tWP	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	1
tRWL	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	1
tCWL	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	1

Table 17B. ECC Mode—WR Cycle

Parameter	Slow Cycle Mode			Notes
	C_4	C_5	C_6	
tRP	2TCLCL−T26	2TCLCL−T26	2TCLCL−T26	1
tCPN	2.5TCLCL−T35	2.5TCLCL−T35	2.5TCLCL−T35	1
tRSH	5TCLCL−T34	5TCLCL−T34	4TCLCL−T34	1
tCSH	5TCLCL−T26	5TCLCL−T26	4TCLCL−T26	1
tCAH	2TCLCL−T34	2TCLCL−T34	2TCLCL−T34	1
tAR	2TCLCL−T26	2TCLCL−T26	2TCLCL−T26	1
tT	3/30	3/30	3/30	2
tRWC	7TCLCL	7TCLCL	6TCLCL	1
tRRW	5TCLCL−T26	5TCLCL−T26	4TCLCL−T26	1
tCRW	5TCLCL−T34	5TCLCL−T34	4TCLCL−T34	1
tWCH	5TCLCL−T34	5TCLCL−T34	4TCLCL−T34	1, 4
tWCR	5TCLCL−T26	5TCLCL−T26	4TCLCL−T26	1, 4
tWP	3TCLCL−TCL −T36−TBUF	3TCLCL−TCL −T36−TBUF	3TCLCL−TCL −T36−TBUF	1
tRWL	3TCLCL−TCL −T36−TBUF	3TCLCL−TCL −T36−TBUF	3TCLCL−TCL −T36−TBUF	1
tCWL	3TCLCL−TCL −T36−TBUF	3TCLCL−TCL −T36−TBUF	3TCLCL−TCL −T36−TBUF	1

Table 18A. ECC Mode—RMW

Parameter	Fast Cycle Mode				Notes
	C_0	C_1	C_2	C_3	
tRP	3TCLCL−T26	3TCLCL−T26	3TCLCL−T26	3TCLCL−T26	1
tCPN	4TCLCL−T35	4TCLCL−T35	4TCLCL−T35	4TCLCL−T35	1
tRSH	8TCLCL−T34	8TCLCL−T34	10TCLCL−T34	10TCLCL−T34	1
tCSH	9TCLCL−T26	9TCLCL−T26	11TCLCL−T26	11TCLCL−T26	1
tCAH	TCLCL−T34	2TCLCL−T34	2TCLCL−T34	2TCLCL−T34	1
tAR	2TCLCL−T26	3TCLCL−T26	3TCLCL−T26	3TCLCL−T26	1
tT	3/30	3/30	3/30	3/30	2
tRWC	12TCLCL	12TCLCL	14TCLCL	14TCLCL	1
tRRW	9TCLCL−T26	9TCLCL−T26	11TCLCL−T26	11TCLCL−T26	1
tCRW	8TCLCL−T34	8TCLCL−T34	10TCLCL−T34	10TCLCL−T34	1
tRCS	TCLCL−T36 −TBUF	TCLCL−T36 −TBUF	TCLCL−T36 −TBUF	TCLCL−T36 −TBUF	1
tRWD	6TCLCL−T26	6TCLCL−T26	8TCLCL−T26	8TCLCL−T26	1, 4
tCWD	5TCLCL−T34	5TCLCL−T34	7TCLCL−T34	7TCLCL−T34	1
tWP	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	1
tRWL	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	1
tCWL	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	3TCLCL−T36 −TBUF	1

Table 18B. ECC Mode—RMW

Parameter	Slow Cycle Mode			Notes
	C_4	C_5	C_6	
tRP	2TCLCL−T26	2TCLCL−T26	2TCLCL−T26	1
tCPN	2.5TCLCL−T35	2.5TCLCL−T35	2.5TCLCL−T35	1
tRSH	7TCLCL−T34	7TCLCL−T34	5TCLCL−T34	1
tCSH	7TCLCL−T26	7TCLCL−T26	5TCLCL−T26	1
tCAH	2TCLCL−T34	2TCLCL−T34	2TCLCL−T34	1
tAR	2TCLCL−T26	2TCLCL−T26	2TCLCL−T26	1
tT	3/30	3/30	3/30	2
tRWC	9TCLCL	9TCLCL	7TCLCL	1
tRRW	7TCLCL−T26	7TCLCL−T26	5TCLCL−T26	1
tCRW	7TCLCL−T34	7TCLCL−T34	5TCLCL−T34	1
tRCS	0.5TCLCL−T36 −TBUF	0.5TCLCL−T36 −TBUF	0.5TCLCL−T36 −TBUF	1
tRWD	4TCLCL+TCL −T26	4TCLCL+TCL −T26	2TCLCL+TCL −T26	1
tCWD	4TCLCL+TCL −T34	4TCLCL+TCL −T34	2TCLCL+TCL −T34	1
tWP	3TCLCL−TCL −T36−TBUF	3TCLCL−TCL −T36−TBUF	3TCLCL−TCL −T36−TBUF	1
tRWL	3TCLCL−TCL −T36−TBUF	3TCLCL−TCL −T36−TBUF	3TCLCL−TCL −T36−TBUF	1
tCWL	3TCLCL−TCL −T36−TBUF	3TCLCL−TCL −T36−TBUF	3TCLCL−TCL −T36−TBUF	1

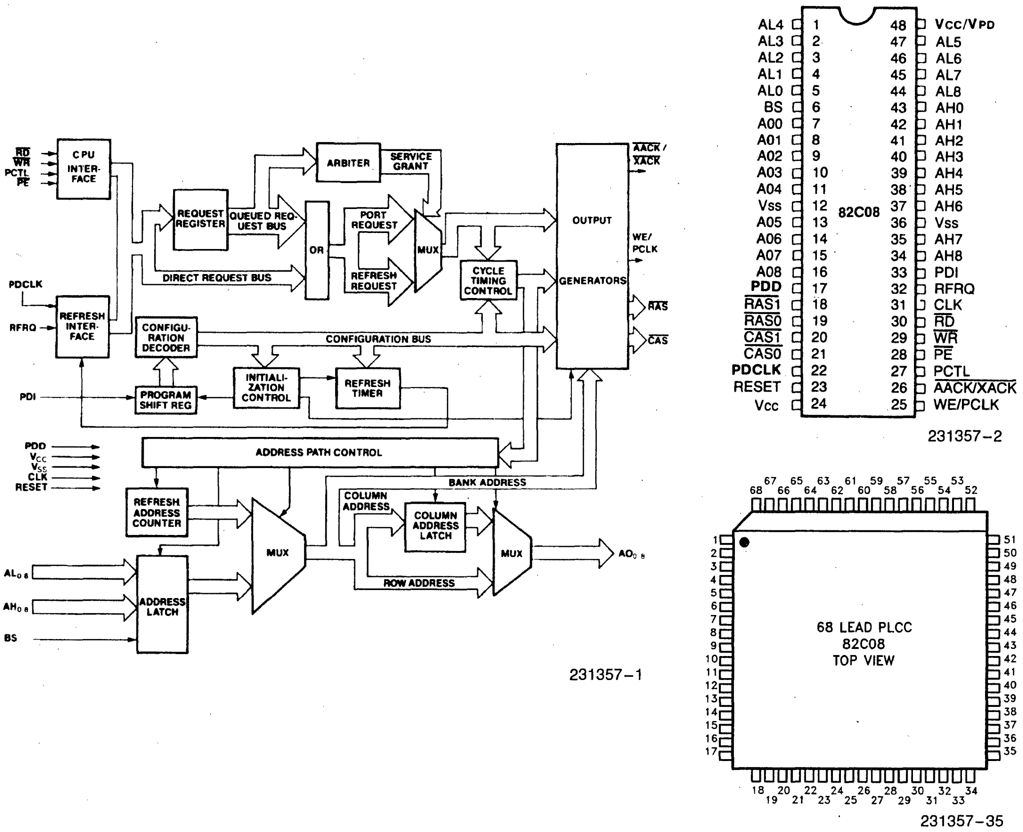
NOTES:

1. Minimum.
2. Value on right is maximum; value on left is minimum.
3. Applies to the eight warm-up cycles during initialization only.
4. Applies to the eight warm-up cycles and to the memory initialization cycles during initialization only.
5. TP = TCLCL
   T26 = TCLRSL
   T34 = TCLCSL
   T35 = TCLCSH
   T36 = TCLW
   TBUF = TTL Buffer delay.

82C08
CHMOS DYNAMIC RAM CONTROLLER

- 0 Wait State with INTEL μProcessors
- iAPX 286 (10, 8 MHz) } 82C08-20 20 MHz, 82C08-16 16 MHz
  iAPX 186/88, 86/88 } 82C08-10 10 MHz, 82C08-8 8 MHz
- Supports 64K and 256K DRAMs (256K x 1 and 256K x 4 Organizations)
- Power Down Mode with Programmable Memory Refresh using Battery Backup
- Directly Addresses and Drives up to 1 Megabyte without External Drivers
- Microprocessor Data Transfer and Advance Acknowledge Signals
- Five Programmable Refresh Modes
- Automatic RAM Warm-up
- Pin-Compatible with 8208
- 48 Lead Plastic DIP; 68 Lead PLCC
  (See Intel Packaging; Order Number: 231369-001)
- Compatible with Normal Modes of Static Column and Ripplemode DRAMs

The Intel 82C08 Dynamic RAM Controller is a CMOS, high performance, systems oriented, Dynamic RAM controller that is designed to easily interface 64K and 256K Dynamic RAMs to Intel and other microprocessors. The 82C08 also has a power down mode where only the refresh logic is activated using battery backup.



231357-1

231357-2

231357-35

Figure 1. Block Diagram and Pinout Diagrams

Table 1. Pin Description

Symbol	DIP Pin	PLCC	Type	Name and Function
AL0 AL1 AL2 AL3 AL4 AL5 AL6 AL7 AL8	5 4 3 2 1 47 46 45 44	55 56 57 58 59 63 64 66 67	I I I I I I I I I	**ADDRESS LOW:** These lower order address inputs are used to generate the column address for the internal address multiplexer. In iAPX 286 mode (CFS = 1), these addresses are latched internally.
AH0 AH1 AH2 AH3 AH4 AH5 AH6 AH7 AH8	43 42 41 40 39 38 37 35 34	2 3 4 5 6 7 8 12 13	I I I I I I I I I	**ADDRESS HIGH:** These higher order address inputs are used to generate the row address for the internal address multiplexer. In iAPX 286 mode, these addresses are latched internally.
BS	6	50	I	**BANK SELECT:** This input is used to select one of the two banks of the dynamic RAM array.
AO0 AO1 AO2 AO3 AO4 AO5 AO6 AO7 AO8	7 8 9 10 11 13 14 15 16	49 48 47 46 45 41 40 39 38	O O O O O O O O O	**ADDRESS OUTPUTS:** These outputs are designed to provide the row and column addresses, of either the CPU or the refresh counter, to the dynamic RAM array. These outputs drive the dynamic RAM array directly and need no external drivers. However, they typically need series resistors to match impedances.
$\overline{\text{RAS0}}$ $\overline{\text{RAS1}}$	19 18	33 36	O O	**ROW ADDRESS STROBE:** These outputs are used by the dynamic RAM array to latch the row address, present on the AO0–8 pins. These outputs are selected by the BS pin. These outputs drive the dynamic RAM array directly and need no external drivers.
$\overline{\text{CAS0}}$ $\overline{\text{CAS1}}$	21 20	30 31	O O	**COLUMN ADDRESS STROBE:** These outputs are used by the dynamic RAM array to latch the column address, present on the AO0–8 pins, These outputs are selected by the BS pin. These outputs drive the dynamic RAM array directly and need no external drivers.
RESET	23	28	I	**RESET:** This active high signal causes all internal counters to be reset. Upon release of RESET, data appearing at the PDI pin is clocked-in by the PCLK output. The states of the PDI, PCTL, and RFRQ pins are sampled by RESET going inactive and are used to program the 82C08. An 8-cycle dynamic RAM warm-up is performed after clocking PDI bits into the 82C08.
WE/ PCLK	25	24	O	**WRITE ENABLE/PROGRAMMING CLOCK:** Immediately after a RESET this pin becomes PCLK and is used to clock serial programming data into the PDI pin. After the 82C08 is programmed this active high signal provides the dynamic RAM array the write enable input for a write operation.

Table 1. Pin Description (Continued)

Symbol	DIP Pin	PLCC	Type	Name and Function
$\overline{AACK}$/ $\overline{XACK}$	26	23	O	**ADVANCE ACKNOWLEDGE/TRANSFER ACKNOWLEDGE:** When the X programming bit is set to logic 0 this pin is $\overline{AACK}$ and indicates that the processor may continue processing and that data will be available when required. This signal is optimized for the system by programming the S program-bit for synchronous or asynchronous operation. The S programming bit determines whether this strobe will be early or late. If another dynamic RAM cycle is in progress at the time of the new request, the $\overline{AACK}$ is delayed. When the X programming bit is set to logic 1 this pin is $\overline{XACK}$ and indicates that data on the bus is valid during a read cycle or that data may be removed from the bus during a write cycle. $\overline{XACK}$ is a MULTIBUS compatible signal.
PCTL	27	22	I	**PORT CONTROL:** This pin is sampled on the falling edge of RESET. It configures the 82C08 to accept command inputs or processor status inputs. If PCTL is low after RESET the 82C08 is programmed to accept bus/multibus command inputs or iAPX 286 status inputs. If PCTL is high after RESET the 82C08 is programmed to accept status inputs from iAPX 86 or iAPX 186 type processors. The S2 status line should be connected to this input if programmed to accept iAPX 86 or iAPX 186 inputs. When programmed to accept bus commands or iAPX 286 status inputs, it should be tied low or it may be connected to INHIBIT when operating with MULTIBUS.
$\overline{PE}$	28	21	I	**PORT ENABLE:** This pin serves to enable a RAM cycle request. It is generally decoded from the address bus.
$\overline{WR}$	29	20	I	**WRITE:** This pin is the write memory request command input. This input also directly accepts the $\overline{S0}$ status line from Intel processors.
$\overline{RD}$	30	19	I	**READ:** This pin is the read memory request command pin. This input also directly accepts the $\overline{S1}$ status line from Intel processors.
CLK	31	16	I	**CLOCK:** This input provides the basic timing for sequencing the internal logic.
RFRQ	32	15	I	**REFRESH REQUEST:** This input is sampled on the falling edge of RESET. If RFRQ is high at RESET then the 82C08 is programmed for internal-refresh request or external-refresh request with failsafe protection. If RFRQ is low at RESET then the 82C08 is programmed for external-refresh without failsafe protection or burst refresh. Once programmed the RFRQ pin accepts signals to start an external-refresh with failsafe protection or external-refresh without failsafe protection or a burst refresh. RFRQ is also sampled when PDD is activated. When RFRQ = 1 it will cause 3 burst refresh cycles.
PDI	33	14	I	**PROGRAM DATA INPUT:** This input is sampled by RESET going low. It programs the various user selectable options in the 82C08. The PCLK pin shifts programming data into the PDI input from an external shift register. This pin may be strapped low to a default iAPX 186 mode configuration or high to a default iAPX 286 mode configuration.
*PDD	17	37	I	**POWER DOWN DETECT:** This input is sampled before every memory cycle to inform the 82C08 of system detection of power failure. When active, the 82C08 remains in power down mode and performs memory refresh only ($\overline{RAS}$-only refresh). In power down mode the 82C08 uses PDCLK for timing and VPD for power.

Table 1. **Pin Description** (Continued)

Symbol	DIP Pin	PLCC	Type	Name and Function
*PDCLK	22	29	I	**POWER DOWN CLOCK:** This pin is used as a clock for internal refresh circuits during power down. The input can be asynchronous to pin 31. Extended refresh is achieved by slowing down this clock. This pin should be grounded if not used.
*V_{CC}/V_{PD}	48	61, 62	I	**POWER:** Power supply for internal logic. This should be held active during power down, and normal operation.
V_{CC}	24	26, 27	I	**POWER:** Supply for drivers. Need not be held active during power down.
V_{SS}	12 36	9, 10, 11, 42, 43, 44	I I	GROUND GROUND
NC	—	17, 18, 1, 25, 32, 34, 35, 51, 53, 54, 60, 65, 68		
V_{CCS}		52		Connect to V_{PP}, Pin 48 for PLCC package.

*Different function than the HMOS 8208.

GENERAL DESCRIPTION

The Intel 82C08 Dynamic RAM Controller is a microcomputer peripheral device which provides the necessary signals to address, refresh, and directly drive 64K and 256K dynamic RAMs. It is compatible with static column or ripple mode DRAMs in the normal mode. It does not support the fast transfer mode of these DRAMs.

The 82C08 supports several microprocessor interface options including synchronous and asynchronous operations for iAPX 86, iAPX 186, iAPX 286, and MULTIBUS. The 82C08 will also interface to non-Intel microprocessors.

The 82C08 is a CHMOS version of the 8208 and is pin compatible with it. Three pins—17, 22, and 48—of the 82C08 are different from the 8208. They provide a power down mode that allows the system to run at a much lower ICC. In this mode, the 82C08 refreshes the DRAM using battery backup. The power down current (I_{PD}) that is drawn by the 82C08 is very small compared to the I_{CC} which allows memory to be kept alive with a battery. A separate refresh clock, pin 22, allows the designer to take advantage of RAMs that permit extended memory refresh.

The 82C08 also has some timing changes versus the 8208. In order to eliminate the external bus latches, both WE and $\overline{CAS}$ timings are shortened. These timing changes are backwards-compatible for 8208 designs.

FUNCTIONAL DESCRIPTION

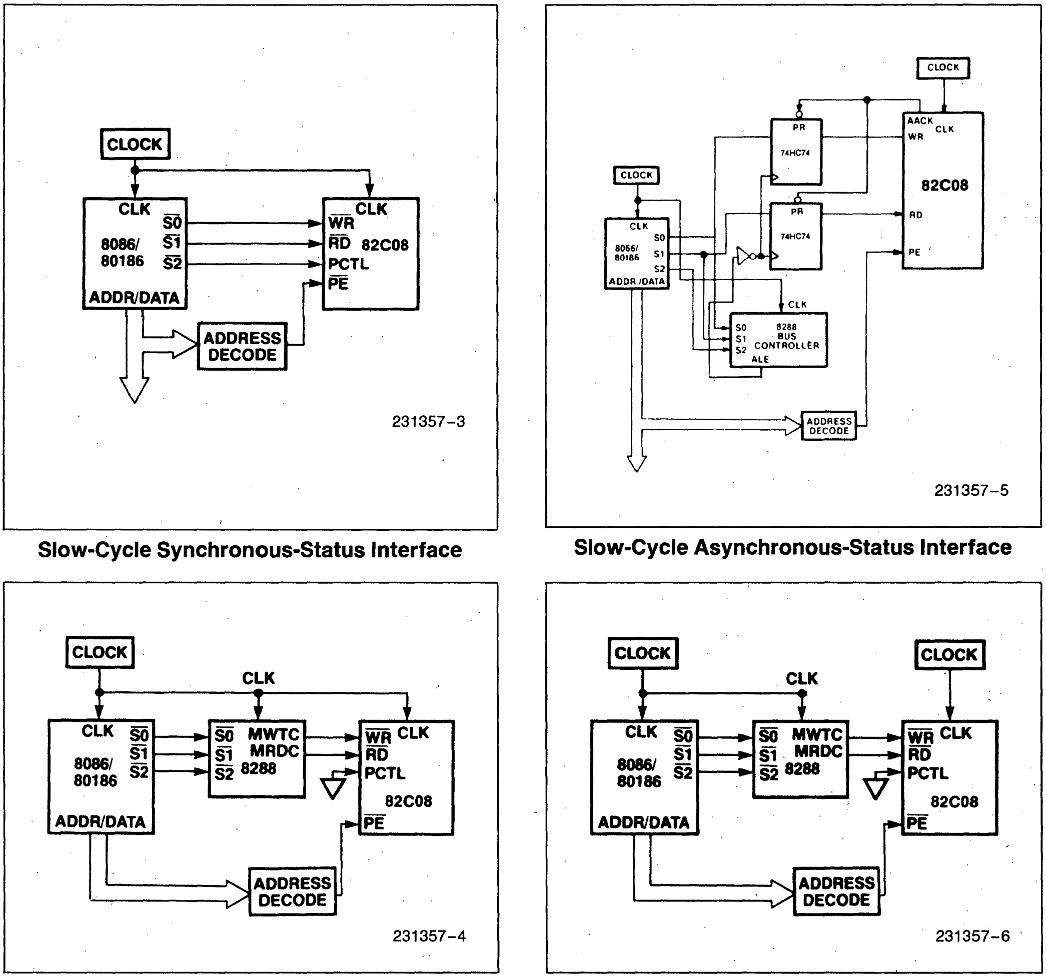
Processor Interface

The 82C08 has control circuitry capable of supporting one of several possible bus structures. The 82C08 may be programmed to run synchronous or asynchronous to the processor clock. The 82C08 has been optimized to run synchronously with Intel's iAPX 86, iAPX 88, iAPX 186/188 and iAPX 286. When the 82C08 is programmed to run in asynchronous mode, the 82C08 inserts the necessary synchronization circuitry for the $\overline{RD}$, $\overline{WR}$ inputs.

The 82C08 achieves high performance (i.e. no wait states) by decoding the status lines directly from the processor. The 82C08 can also be programmed to receive read or write MULTIBUS commands or commands from a bus controller.

The 82C08 may be programmed to operate synchronously to the processor. It can also be programmed to run at various frequencies. (See Microprocessor Clock Frequency Option.)

Figure 2 shows the different processor interfaces to the 82C08 using the synchronous or asynchronous mode and status or command interface. Figure 3 shows detailed interfaces to the iAPX 186 and iAPX 286 processors.



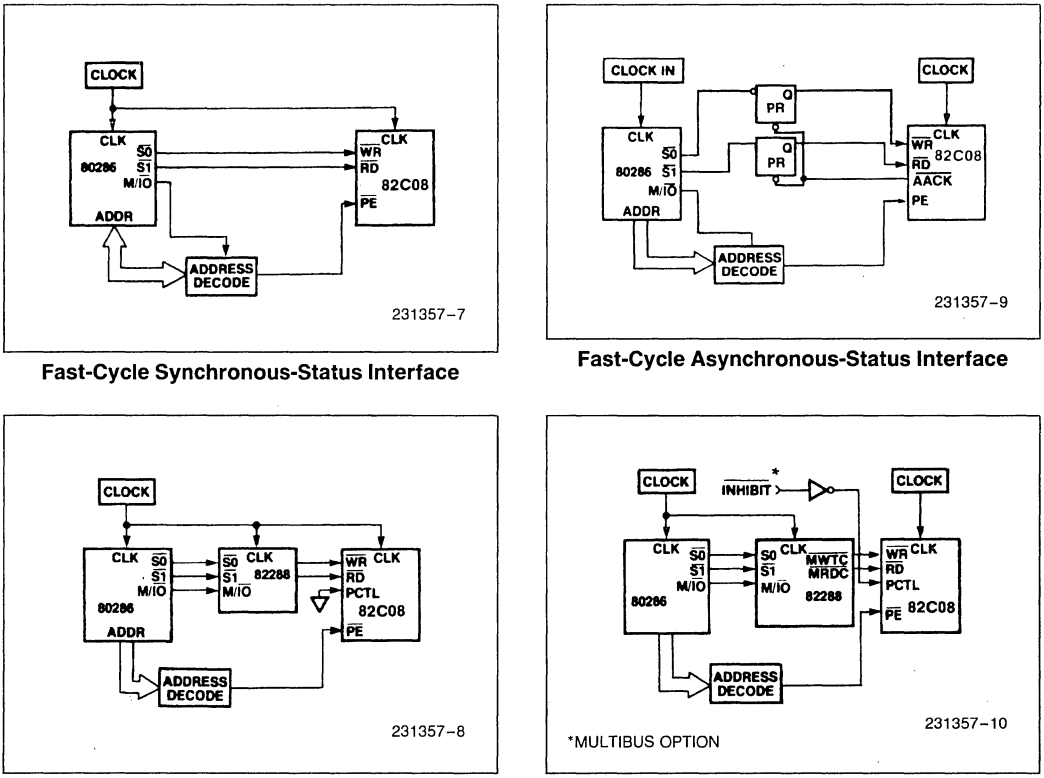
Slow-Cycle Synchronous-Status Interface

Slow-Cycle Asynchronous-Status Interface

Slow-Cycle Synchronous-Command Interface

Slow-Cycle Asynchronous-Command Interface

Figure 2A. Slow-cycle (CFS = 0) Port Interfaces Supported by the 82C08



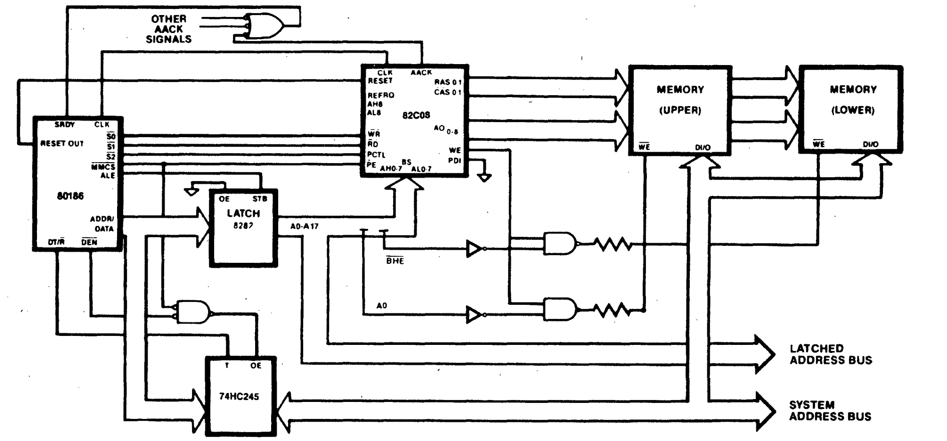
Fast-Cycle Synchronous-Status Interface

Fast-Cycle Asynchronous-Status Interface

Fast-Cycle Synchronous-Command Interface

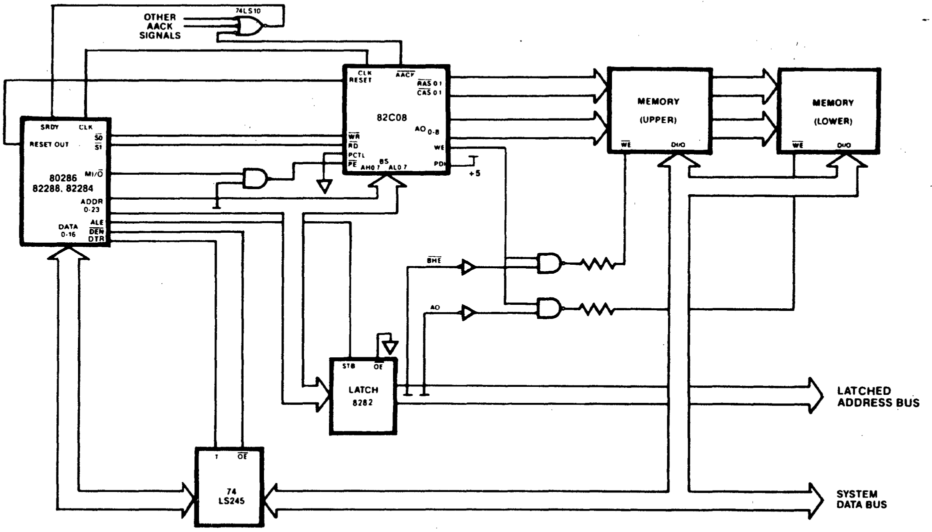
Fast-Cycle Asynchronous-Command Interface

Figure 2B. Fast-cycle (CFS = 1) Port Interfaces Supported by the 82C08



231357-11

Figure 3A. 82C08 Interface to an 80186

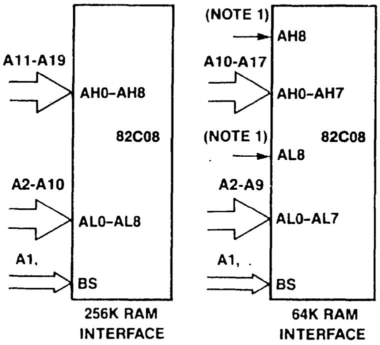


231357-12

Figure 3B. 82C08 Interface to an 80286

Dynamic RAM Interface

The 82C08 is capable of addressing 64K and 256K dynamic RAMs. Figure 3 shows the connection of the processor address bus to the 82C08 using the different RAMs.



231357-13

NOTES:
1. Unassigned address input pins should be strapped high.
2. A0 along with BHE are used to select a byte within a processor word.
3. Low order address bit is used as a bank select input so that consecutive memory access requests are to alternate banks allowing bank interleaving of memory cycles.

Figure 3. Processor Address Interface to the 82C08 Using 64K, and 256K RAMS

The 82C08 divides memory into two banks, each bank having its own Row ($\overline{\text{RAS}}$) and Column ($\overline{\text{CAS}}$) Address Strobe pair. This organization permits RAM cycle interleaving. RAM cycle interleaving overlaps the start of the next RAM cycle with the RAM precharge period of the previous cycle. Hiding the precharge period of one RAM cycle behind the data access period of the next RAM cycle optimizes memory bandwidth and is effective as long as successive RAM cycles occur in the alternate banks.

Successive data access to the same bank cause the 82C08 to wait for the precharge time of the previous RAM cycle. But when the 82C08 is programmed in an iAPX 186 synchronous configuration, consecutive cycles to the same bank do not result in additional wait states (i.e. 0 wait state).

If not all RAM banks are occupied, the 82C08 can be programmed to reassign the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ strobes to allow using wider data words without increasing the loading on the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ drivers. Table 2 shows the bank selection decoding and the corresponding $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ assignments. For example, if only one RAM bank is occupied, then the two $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ strobes are activated with the same timing.

Table 2. Bank Selection Decoding and Word Expansion

Program Bit RB	Bank Input BS	82C08 $\overline{\text{RAS}}/\overline{\text{CAS}}$ Pair Allocation
0	0	$\overline{\text{RAS}}_{0,\,1}$, $\overline{\text{CAS}}_{0,\,1}$ to Bank 0
0	1	Illegal
1	0	$\overline{\text{RAS}}_0$, $\overline{\text{CAS}}_0$ to Bank 0
1	1	$\overline{\text{RAS}}_1$, $\overline{\text{CAS}}_1$ to Bank 1

Program bit RB is not used to check the bank select input BS. The system design must protect from accesses to "illegal", non-existent banks of memory by deactivating the PE input when addressing an "illegal", non-existent bank of memory.

The 82C08 adjusts and optimizes internal timings for either the fast or slow RAMs as programmed. (See RAM Speed Option.)

Memory Initialization

After programming, the 82C08 performs eight RAM "wake-up" cycles to prepare the dynamic RAM for proper device operation.

Refresh

The 82C08 provides an internal refresh interval counter and a refresh address counter to allow the 82C08 to refresh memory. The 82C08 has a 9-bit internal refresh address counter which will refresh 128 rows every 2 milliseconds, 256 rows every 4 milliseconds or 512 rows every 8 milliseconds, which allows all RAM refresh options to be supported. In addition, there exists the ability to refresh 256 row address locations every 2 milliseconds via the Refresh Period programming option.

The 82C08 may be programmed for any of five different refresh options: Internal refresh only, External refresh with failsafe protection, External refresh without failsafe protection, Burst refresh modes, or no refresh. (See Refresh Options.)

It is possible to decrease the refresh time interval by 10%, 20% or 30%. This option allows the 82C08 to compensate for reduced clock frequencies. Note that an additional 5% interval shortening is built-in in all refresh interval options to compensate for clock variations and non-immediate response to the internally generated refresh request. (See Refresh Period Options.)

External Refresh Requests after RESET

External refresh requests are not recognized by the 82C08 until after it is finished programming and preparing memory for access. Memory preparation includes 8 RAM cycles to prepare and ensure proper dynamic RAM operation. The time it takes for the 82C08 to recognize a request is shown below.

eg. 82C08 System Response:

TRESP = TPROG + TPREP
where: TPROG = (40) (TCLCL) programming time
TPREP = (8) (32) (TCLCL) RAM
warm-up time
if TCLCL = 125 ns then TRESP = 37 μs

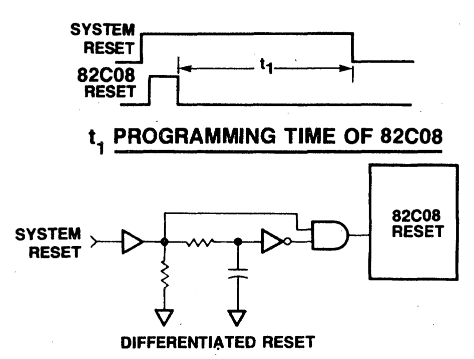
Reset

RESET is an asynchronous input, its falling edge is used by the 82C08 to directly sample the logic levels of the PCTL, RFRQ, and PDI inputs. The internally synchronized falling edge of reset is used to begin programming operations (shifting in the contents of the external shift register, if needed, into the PDI input).

Differentiated reset is unnecessary when the default synchronization programming is used.

Until programming is complete the 82C08 latches but does not respond to command or status inputs. A problem may occur if the S bit is programmed inconsistently from the Command which was latched before programming was completed. A simple means of preventing commands or status from occurring during this period is to differentiate the system reset pulse to obtain a smaller reset pulse for the 82C08.

The differentiated reset pulse would be shorter than the system reset pulse by at least the programming period required by the 82C08. The differentiated reset pulse first resets the 82C08, and system reset would reset the rest of the system. While the rest of the system is still in reset, the 82C08 completes its programming. Figure 4 illustrates a circuit to accomplish this task.



NOTES: 231357-14

1. Required only when the synchronization option is altered from its initial default value.

2. V_{CC} must be stable before system reset is activated when using this circuit.

Figure 4. 82C08 Differentiated Reset Circuit

Within four clocks after RESET goes active, all the 82C08 outputs will go high, except for AO0-2, which will go low.

OPERATIONAL DESCRIPTION

Programming the 82C08

The 82C08 is programmed after reset. On the falling edge of RESET, the logic states of several input pins are latched internally. The falling edge of RESET actually performs the latching, which means that the logic levels on these inputs must be stable prior to that time. The inputs whose logic levels are latched at the end of reset are the PCTL, RFRQ, and PDI pins.

Status/Command Mode

The processor port of the 82C08 is configured by the states of the PCTL pin. Which interface is selected depends on the state of the PCTL pin at the end of reset. If PCTL is high at the end of reset, the 8086/80186 Status interface is selected; if it is low, then the MULTIBUS or Command interface is selected.

The status lines of the 80286 are similar in code and timing to the Multibus command lines, while the status code and timing of the 8086 and 8088 are identical to those of the 80186 and 80188 (ignoring the differences in clock duty cycle). Thus there exists two interface configurations, one for the 80286 status or Multibus memory commands, which is called the Command interface, and one for 8086, 8088, 80186 or 80188 status, called the 8086 Status interface. The Command interface can also directly interface to the command lines of the bus controllers for the 8086, 8088, 80186 and the 80286.

The 80186 Status interface allows direct decoding of the status lines for the iAPX 86, iAPX 88, iAPX 186 and the iAPX 188. Table 3 shows how the status lines are decoded.

Table 3A. Status Coding of 8086, 80186 and 80286

Status Code			Function	
$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	8086/80186	80286*
0	0	0	INTERRUPT	INTERRUPT
0	0	1	I/O READ	I/O READ
0	1	0	I/O WRITE	I/O WRITE
0	1	1	HALT	IDLE
1	0	0	INSTRUCTION FETCH	HALT
1	0	1	MEMORY READ	MEMORY READ
1	1	0	MEMORY WRITE	MEMORY WRITE
1	1	1	IDLE	IDLE

* Refer to 80286 pin description table

Table 3B. 82C08 Response

82C08 Command			Function	
PCTL	$\overline{RD}$	$\overline{WR}$	8086/80186 Status Interface	80286 Status or Command Interface
0	0	0	IGNORE	IGNORE*
0	0	1	IGNORE	READ
0	1	0	IGNORE	WRITE
0	1	1	IGNORE	IGNORE
1	0	0	READ	IGNORE
1	0	1	READ	INHIBIT
1	1	0	WRITE	INHIBIT
1	1	1	IGNORE	IGNORE

*Illegal with CFS = 0

Refresh Options

Immediately after system reset, the state of the RFRQ input pin is examined. If RFRQ is high, the 82C08 provides the user with the choice between self-refresh and user-generated refresh with failsafe protection. Failsafe protection guarantees that if the user does not come back with another refresh request before the internal refresh interval counter times out, a refresh request will be automatically generated. If the RFRQ pin is low immediately after a reset, then the user has the choice of a single external refresh cycle without failsafe, burst refresh or no refresh.

Internal Refresh Only

For the 82C08 to generate internal refresh requests, it is necessary only to strap the RFRQ input pin high.

External Refresh with Failsafe

To allow user-generated refresh requests with failsafe protection, it is necessary to hold the RFRQ input high until after reset. Thereafter, a low-to-high transition on this input causes a refresh request to be generated and the internal refresh interval counter to be reset. A high-to-low transition has no effect on the 82C08. A refresh request is not recognized until a previous request has been serviced.

External Refresh without Failsafe

To generate single external refresh requests without failsafe protection, it is necessary to hold RFRQ low until after reset. Thereafter, bringing RFRQ high for one clock period will cause a refresh request to be generated. A refresh request is not recognized until a previous request has been serviced.

Burst Refresh

Burst refresh is implemented through the same procedure as a single external refresh without failsafe (i.e., RFRQ is kept low until after reset). Thereafter, bringing RFRQ high for at least two clock periods will cause a burst of up to 128 row address locations to be refreshed. A refresh request is not recognized until a previous request has been serviced (i.e. burst is completed).

No Refresh

It is necessary to hold RFRQ low until after reset. This is the same as programming External Refresh without Failsafe. No refresh is accomplished by keeping RFRQ low.

Option Program Data Word

PROGRAMMING FOR SLOW CYCLE

The program data word consists of 9 program data bits, PD0–PD8. If the first program data bit, PD0 is set to logic 0, the 82C08 is configured to support iAPX 186, 188, 86, or 88 systems. The remaining bits, PD1–PD8, may then be programmed to optimize a selected system configuration. A default of all zeros in the remaining program bits optimizes the 82C08 timing for 8 MHz Intel CPUs using 150 ns (or faster) dynamic RAMs with no performance penalty.

PROGRAMMING FOR FAST CYCLE

If the first program data bit is set to logic 1, the 82C08 is configured to support iAPX 286 systems (Command mode). A default of all ones in the program bits optimizes the 82C08 timing for an 8 MHz 286 using 120 ns DRAMs at zero wait states. Note that the programming bits PD1–8 change polarity according to PD0. This ensures the same choice of options for both default modes.

Table 4A shows the various options that can be programmed into the 82C08.

Table 4A. Program Data Word

Program Data Bit	Name		Polarity/Function
	PD0 = 0	PD0 = 1	

PD0	CFS	CFS	CFS = 0 SLOW CYCLE
			CFS = 1 FAST CYCLE

PD1	$\overline{\text{S}}$	S	$\overline{\text{S}}$ = 0 SYNCHRONOUS* $\overline{\text{S}}$ = 1 ASYNCHRONOUS
PD2	$\overline{\text{RFS}}$	RFS	$\overline{\text{RFS}}$ = 0 FAST RAM* $\overline{\text{RFS}}$ = 1 SLOW RAM
PD3	$\overline{\text{RB}}$	RB	RAM BANK OCCUPANCY SEE TABLE 2
PD4	CI1	$\overline{\text{CI1}}$	COUNT INTERVAL BIT 1; SEE TABLE 6
PD5	CI0	$\overline{\text{CI0}}$	COUNT INTERVAL BIT 0; SEE TABLE 6
PD6	$\overline{\text{PLS}}$	PLS	$\overline{\text{PLS}}$ = 0 LONG REFRESH PERIOD* $\overline{\text{PLS}}$ = 1 SHORT REFRESH PERIOD
PD7	$\overline{\text{FFS}}$	FFS	$\overline{\text{FFS}}$ = 0 FAST CPU FREQUENCY* $\overline{\text{FFS}}$ = 1 SLOW CPU FREQUENCY

PD8	X	$\overline{\text{X}}$	X = 0 $\overline{\text{AACK}}$*
			X = 1 $\overline{\text{XACK}}$

* Default in both modes

Using an External Shift Register

The 82C08 may be programmed by using an external shift register with asynchronous load capability such as a 74HC165. The reset pulse serves to parallel load the shift register and the 82C08 supplies the clocking signal (PCLK) to shift the data into the PDI programming pin. Figure 6 shows a sample circuit diagram of an external shift register circuit.

Serial data is shifted into the 82C08 via the PDI pin (33), and clock is provided by the WE/PCLK pin (25), which generates a total of 9 clock pulses.

WE/PCLK is a dual function pin. During programming, it serves to clock the external shift register, and after programming is completed, it reverts to the write enable RAM control output pin. As the pin changes state to provide the write enable signal to the dynamic RAM array, it continues to clock the shift register. This does not present a problem because data at the PDI pin is ignored after programming. Figure 7 illustrates the timing requirements of the shift register.

Default Programming Options

After reset, the 82C08 serially shifts in a program data word via the PDI pin. This pin may be strapped low or high, or connected to an external shift register. Strapping PDI low causes the 82C08 to default to the iAPX 186 system configuration, while high causes a default to the iAPX 286 configuration. Table 4B shows the characteristics of the default configuration for Fast Cycle (PDI = 1) and Slow Cycle (PDI = 0). If further system flexibility is needed, one external shift register, like a 74HC165, can be used to tailor the 82C08 to its operating environment.

Table 4B. Default Programming

Synchronous interface
Fast RAM (Note 1)
2 RAM banks occupied
128 row refresh in 2 ms; 256 in 4 ms, 512 in 8 ms
Fast processor clock frequency
Advanced ACK strobe

NOTE:
1. For iAPX 86/186 systems either slow or fast (150 or 100 ns) RAMS will run at 8 MHz with zero wait states.

Synchronous/Asynchronous Mode (S program bit)

The 82C08 may be configured to accept synchronous or asynchronous commands ($\overline{\text{RD}}$, $\overline{\text{WR}}$, PCTL) and Port Enable ($\overline{\text{PE}}$) via the S program bit. The state of the S programming bit determines whether the interface is synchronous or asynchronous.

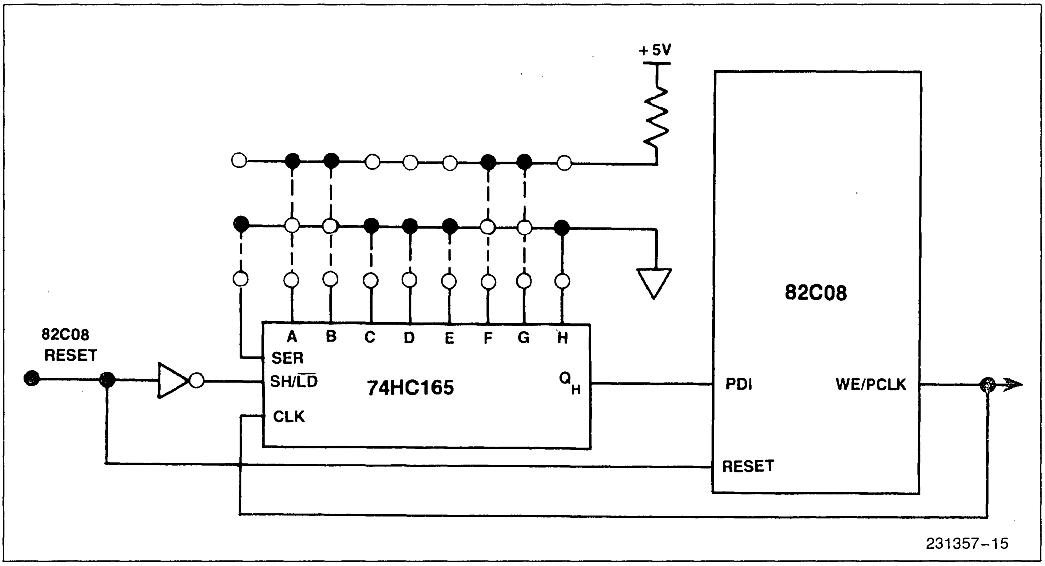
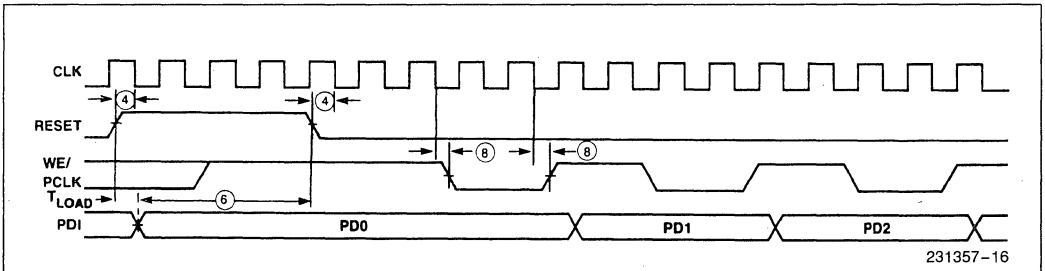


Figure 6. External Shift Register Interface



NOTES:
TRTVCL — Reset is an asychronous input, if reset occurs before TRTVCL, then it is guaranteed to be recognized.
TPGVCL — Minimum PDI valid time prior to reset going low.
TCLPC — MUX/PCLK delay.
TLOAD — Asychronous load data propagation delay.

Figure 7. Timing Illustrating External Shift Register Requirements for Programming the 82C08

While the 82C08 may be configured with either the Status or Command (MULTIBUS) interface in the Synchronous mode, certain restrictions exist in the Asynchronous mode. An Asynchronous-Command interface is directly supported. An Asynchronous-80186/80286 Status interface using the status lines of the 80186/80286 is supported with the use of TTL gates as illustrated in Figure 2. In the 80186 case, the TTL gates are needed to guarantee that status does not appear at the 82C08's inputs too much before address, so that a cycle would start before address was valid. In the case of the 80286, the TTL gates are used for lengthening the Status pulse, as required by the TRWL timing.

Microprocessor Clock Cycle Option (CFS and FFS program bits)

The 82C08 is programmed to interface with microprocessors with "slow cycle" timing like the 8086, 8088, 80186, and 80188, and with "fast cycle" microprocessors like the 80286. The CFS bit is used to select the appropriate timing.

The FFS option is used to select the speed of the microprocessor clock. Table 5 shows the various microprocessor clock frequency options that can be programmed. The external clock frequency must be programmed so that the failsafe refresh repetition circuitry can adjust its internal timing accordingly to produce a refresh request as programmed.

Table 5. Microprocessor Clock Frequency Options

Program Bits		Processor	Clock Frequency
CFS	FFS		
0	0	iAPX 86, 88, 186, 188	≤ 5 MHz
0	1	iAPX 86, 88, 186, 188	> 5 MHz
1	0	iAPX 286	≤ 10 MHz
1	1	iAPX 286	> 10 MHz

RAM Speed Option (RFS program bit)

The RAM Speed programming option determines whether RAM timing will be optimized for a fast or slow RAM. Whether a RAM is fast or slow is measured relative to 100 ns DRAMs (fast) or 150 ns DRAMs (slow). This option is only a factor in Fast cycle Mode (CFS = 1).

Refresh Period Options (CI0, CI1 and PLS program bits)

The 82C08 refreshes with either 128 rows every 2 milliseconds, with 256 rows every 4 milliseconds or 512 rows every 8 milliseconds. This translates to one refresh cycle being executed approximately once every 15.6 microseconds. This rate can be changed to 256 rows every 2 milliseconds or a refresh approximately once every 7.8 microseconds via the Period Long/Short, program bit PLS, programming option.

The Count Interval 0 (CI0) and Count Interval 1 (CI1) programming options allow the rate at which refresh requests are generated to be increased in order to permit refresh requests to be generated close to the 15.6 or 7.8 microsecond period when the 82C08 is operating at reduced frequencies. The interval between refreshes is decreased by 0%, 10%, 20%, or 30% as a function of how the count interval bits are programmed. A 5% guardband is built-in to allow for any clock frequency variations. Table 6 shows the refresh period options available.

The numbers tabulated under Count Interval represent the number of clock periods between internal refresh requests. The percentages in parentheses represent the decrease in the interval between refresh requests.

Table 6. Refresh Count Interval Table

Ref. Period (μs)	CFS	PLS	FFS	Count Interval CI1, CI0 (82C08 Clock Periods)			
				00 (0%)	01 (10%)	10 (20%)	11 (30%)
15.6	1	1	1	236	212	188	164
7.8	1	0	1	118	106	94	82
15.6	1	1	0	148	132	116	100
7.8	1	0	0	74	66	58	50
15.6	0	1	1	118	106	94	82
7.8	0	0	1	59	53	47	41
15.6	0	1	0	74	66	58	50
7.8	0	0	0	37	33	29	25

The refresh count interval is set up for the following basic frequencies:

5 MHz slow cycle

8 MHz slow cycle

10 MHz fast cycle

16 MHz fast cycle

Example: Best 12 MHz fast cycle performance can be achieved using the basic frequency of 16 MHz (CFS = 1, FFS = 1) and the appropriate count interval bits (CI1 = 1, CI0 = 1) to reduce the frequency.

clock period × refresh count interval = refresh period

i.e. 83.3 ns × 164 = 13.6 μs

Example: 10 MHz slow cycle

CFS = 0, FFS = 1, CI1 = 0, CI0 = 0

i.e. 100 ns × 118 = 11.8 μs

Processor Timing

In order to run without wait states, $\overline{\text{AACK}}$ must be used and connected to the $\overline{\text{SRDY}}$ input of the appropriate bus controller. $\overline{\text{AACK}}$ is issued relative to a point within the RAM cycle and has no fixed relationship to the processors's request. The timing is such, however, that the processor will run without wait states, barring refresh cycles. In slow cycle, fast RAM configurations (8086, 80186), $\overline{\text{AACK}}$ is issued on the same clock cycle that issues $\overline{\text{RAS}}$.

Port Enable ($\overline{\text{PE}}$) set-up time requirements depend on whether the 82C08 is configured for synchronous or asynchronous, fast or slow cycle operation. In a synchronous fast cycle configuration, $\overline{PE}$ is required to be set-up to the same clock edge as the commands. If $\overline{PE}$ is true (low), a RAM cycle is started; if not, the cycle is not started until the $\overline{RD}$ or $\overline{WR}$ line goes inactive and active again.

In asychronous operation, $\overline{PE}$ is required to be set-up to the same clock edge as the internally synchronized status or commands. Externally, this allows the internal synchronization delay to be added to the status (or command) -to-$\overline{PE}$ delay time, thus allowing for more external decode time than is available in synchronous operation.

The minimum synchronization delay is the additional amount that $\overline{PE}$ must be held valid. If $\overline{PE}$ is not held valid for the maximum synchronization delay time, it is possible that $\overline{PE}$ will go invalid prior to the status or command being synchronized. In such a case the 82C08 may not start a memory cycle. If a memory cycle intended for the 82C08 is not started, then no acknowledge ($\overline{AACK}$ or $\overline{XACK}$) is issued and the processor locks up in endless wait states.

Memory Acknowledge ($\overline{AACK}$, $\overline{XACK}$)

Two types of memory acknowledge signals are supplied by the 82C08. They are the Advanced Acknowledge strobe ($\overline{AACK}$) and the Transfer Acknowledge strobe ($\overline{XACK}$). The S programming bit optimizes $\overline{AACK}$ for synchronous operation ("early" $\overline{AACK}$) or asynchronous operation ("late" $\overline{AACK}$). Both the early and late $\overline{AACK}$ strobes are two clocks long for CFS = 0 and three clocks long for CFS = 1.

The $\overline{XACK}$ strobe is asserted when data is valid (for reads) or when data may be removed (for writes) and meets the MULTIBUS requirements. $\overline{XACK}$ is removed asynchronously by the command going inactive.

Since in an asynchronous operation the 82C08 removes read data before late $\overline{AACK}$ or $\overline{XACK}$ is recognized by the CPU, the user must provide for data latching in the system until the CPU reads the data. In synchronous operation data latching is unnecessary, since the 82C08 will not remove data until the CPU has read it.

If the X programming bit is high, the strobe is configured as $\overline{XACK}$, while if the bit is low, the strobe is configured as $\overline{AACK}$.

Data will always be valid a fixed time after the occurrence of the advanced acknowledge. Thus, the advanced acknowledge may also serve as a RAM cycle timing indicator.

General System Considerations

1. The $\overline{RAS}$0, 1, $\overline{CAS}$0, 1, and AO0-8 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. Although the 82C08 has programmable options, in practice there are only a few choices the designer must make. For iAPX 86/186 systems (CFS = 0) the C2 default mode (pin 33 tied low) is the best choice. This permits zero wait states at 8 and 10 MHz with 150 ns DRAMs. The only consideration is the refresh rate, which must be programmed if the CPU is run at less than 8 MHz.

   For iAPX 286 systems (CFS = 1) the designer must choose between configuration C0 (RFS = 0) and C1 (RFS = 1, $\overline{FFS}$ = 0). C0 permits zero wait state, 8 MHz iAPX 286 operation with 120 ns DRAMs. However, for consecutive reads, this performance depends on interleaving between two banks. The C1 configuration trades off 1 wait state performance for the ability to use 150 ns DRAMs. 150 ns DRAMs can be supported by the C0 configuration using 7 MHz iAPX 286.
3. For non-Intel microprocessors, the asynchronous command mode would be the best choice, since Intel status lines are not available. To minimize the synchronization delay, the 82C08 should use a 16 MHz clock. The preferred timing configuration is C0.

Table 7. Memory Acknowledge Summary

	Synchronous	Asynchronous	XACK
Fast Cycle	AACK Optimized for Local 80286 (early)	AACK Optimized for Remote 80286 (late)	Multibus Compatible
Slow Cycle	AACK Optimized for Local 8086/186 (early)	AACK Optimized for Remote 8086/186 (late)	Multibus Compatible

POWER DOWN

During Power Down (PD) mode, the 82C08 will perform refresh cycles to preserve the memory content. Two pins are dedicated to this feature, PDD (Power Down Detect) and PDCLK (Power Down Clock). PDD is used to inform the 82C08 of a system power failure, and will remain active as long as the power is down. It is the system's responsibility to detect power failure and to supply this signal. PDCLK is used to supply the clock during power down for the 82C08 refresh circuits. It is the system's responsibility to supply this clock.

Power Supplies

Power down is achieved by eliminating the clock from all the 82C08 circuits that are not participating in the refresh generation. The 82C08 has two power pins (V_{CC}'s), one supplies power to the output buffers and the other, to 82C08 logic. All the active circuits during power down are connected to the logic V_{CC}, including the active output buffers. Therefore, it is the user's choice to connect only the logic V_{CC} pin to the back-up power supply, or to connect both pins to it. It is recommended, however, to connect both pins to the same power supply in order to simplify and to shorten the power up time.

Extended Refresh at Power Down (PD)

To reduce power dissipation during PD, 82C08 will support the extended refresh cycle of the Intel 51CXXL (e.g. 51C64L). In this mode, the refresh period can be extended up to 64 milliseconds versus 4 milliseconds in non-extended cycles. This is achieved by slowing down the PDCLK frequency.

The user should take into consideration that when supporting extended refresh during PD, the dynamic RAM must be refreshed completely within 4 milliseconds, without active cycles, both before going into and after coming out of extended refresh. The 82C08 has the option of performing burst refresh of all the memory whenever the user cannot guarantee the 4 milliseconds idle interval. This is achieved by performing 3 consecutive burst refresh cycles, activated internally by the 82C08.

The option of refreshing all the memory is enabled in failsafe mode configuration (RFRQ input high at reset). When 82C08 detects power down, (high level at PDD) it examines the RFRQ input. High level at the RFRQ input will cause 3 PD burst refresh cycles to be performed. The user should supply the power and the system clock during the time interval of the 3 PD burst cycles, e.g. 4700 (fast cycle) or 3100 (slow cycle) clock cycles after activating PDD. Low level at RFRQ input enables the 82C08 to enter power down immediately without executing any bursts.

Power Down Procedure

The 82C08 will preserve the memory content during the entire period of the system operation. Upon detection of power down, the 82C08 will save internally its configuration status and the refresh address counter content, execute 3 burst refresh cycles. (If it is programmed to failsafe mode and the RFRQ input level is high), it will switch the internal clock from the system clock (CLK) to the power down clock (PDCLK) and will continue the refresh to the next address location. (See Figure 11.)

When power is up again (PDD input deactivated), the 82C08 will issue internal reset which will not reprogram the device and will not clear the refresh address counter, and therefore, refresh will continue to the next address location. After the internal reset, 82C08 performs 3 PD burst refresh cycles which refresh the whole memory, as at entering extended PD. This is done to give the 82C08 enough time to wake up. Notice, at the time interval of 4700 (fast cycle) or 3100 (slow cycle) clocks after power recovering no memory access will be performed.

82C08 Outputs on Power Down

Four of the 82C08 outputs are not activated during power down, $\overline{\text{AACK}}$, $\overline{\text{CAS0}}$-1 and WE. All these outputs will be forced to a non-active state, $\overline{\text{AACK}}$ and $\overline{\text{CAS0}}$-1 will be forced high and WE will be forced low (External NAND buffer is used to drive the $\overline{\text{WE}}$ DRAM inputs, hence a high level on the DRAM inputs). The other 82C08 outputs, AO0-9 and $\overline{\text{RAS0}}$-1, will switch to perform the memory refresh in a "$\overline{\text{RAS}}$-ONLY REFRESH CYCLE." The $\overline{\text{RAS}}$ outputs internal pull-ups assure high levels on these outputs, as close as possible to V_{CC}, for low DRAM power. The size of the output buffers, in power down, is smaller than the normal size, and therefore, the speed of these buffers is slower. It is done in order to reduce the speed of charging and discharging the outputs and hence reduce spikes on the power lines. It is required especially in power down, since there is only one power supply pin active which drives the output buffers as well as the internal logic.

All the device inputs, beside PDD, PDCLK, and RESET will be ignored during power down.

During power down burst refresh the 82C08 performs up to 256 refreshes. Whereas during standard burst refresh the 82C08 performs up to 128 refreshes. The power down burst refresh feature allows the 82C08 to support extended refreshes of some DRAMs, configured as 512 rows.

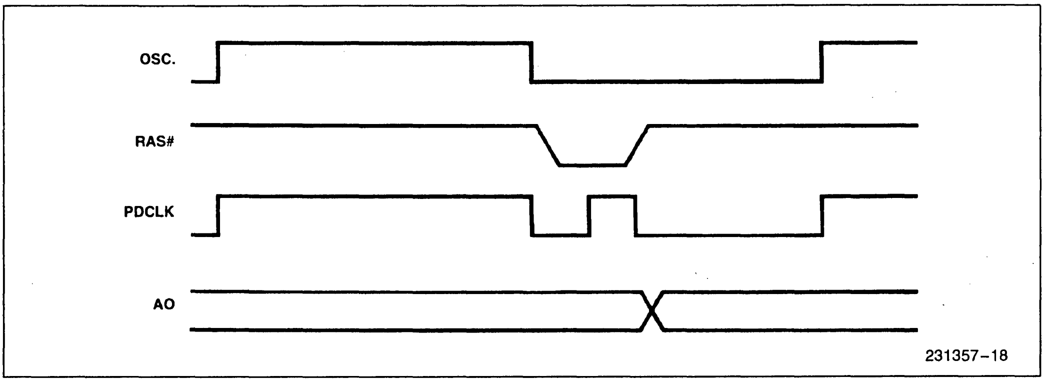


Figure 8

Power Down Detect

As previously mentioned, the PDD input will be supplied by the system to inform the 82C08 of a power failure. It can be asynchronous since the 82C08 synchronizes it internally. The PDD input will be sampled by the 82C08 before the beginning of every memory cycle but only after the termination of programming and initialization period. The user should guarantee V_{CC} and CLK stable during the programming and initialization period (300 clocks after RESET). If the whole memory refresh is required (for extended refresh) then V_{CC} and system clock should be available 4700 (fast cycle) or 3100 (slow cycle) clocks after activating PDD. If it isn't required then 82C08 should wait for present memory cycle completion and synchronization time which will take about 25 system clock cycles.

With PDD going inactive, the 82C08 synchronizes the clock back to the CLK clock, issuing internal reset and will perform 3 PD burst refresh cycles.

NOTE:
The power supplies and the CLK should go up before the PDD is deactivated. All CPU requests will be ignored when PDD is active.

Refresh during Power Down

The 82C08 has two clock pins, CLK is the system clock and PDCLK is the power down clock. PDCLK should be an independent clock which has its own crystal oscillator. When entering power down, the 82C08 will disable the system clock internally and will run with the PDCLK. The system clock will be enabled and the PDCLK will be disabled when power is up. The CLK and PDCLK will be switched internally for the refresh circuits.

During power down, '$\overline{\text{RAS}}$-ONLY REFRESH' will be performed by the 82C08. The time interval between refreshes is 5 PDCLKs and this is fixed for all applications. However, the 82C08 can support the extended refresh (up to 64 ms) by slowing down the PDCLK frequency.

During the power down refresh cycle, $\overline{\text{RAS}}$ will be activated for one PDCLK cycle only. In extended refresh, the PDCLK frequency will be below 50 kHz and this will cause a long duration of the $\overline{\text{RAS}}$ signal which will increase the DRAM's current rapidly. To minimize the $\overline{\text{RAS}}$ low pulse, the two RC networks shown in Figure 9 are designed to insert one very fast (1 μs) cycle whenever $\overline{\text{RAS}}$ is low (see Figure 8). The time constant of RC1 and RC2 should be centered around 300 ns and 100 ns respectively.

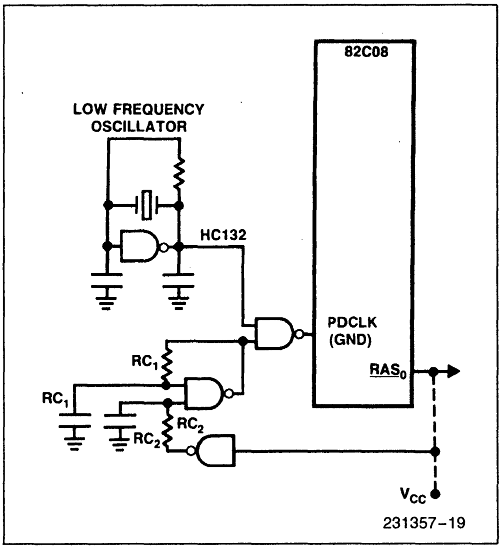


Figure 9. Low Frequency Oscillator

Power Down Synchronization

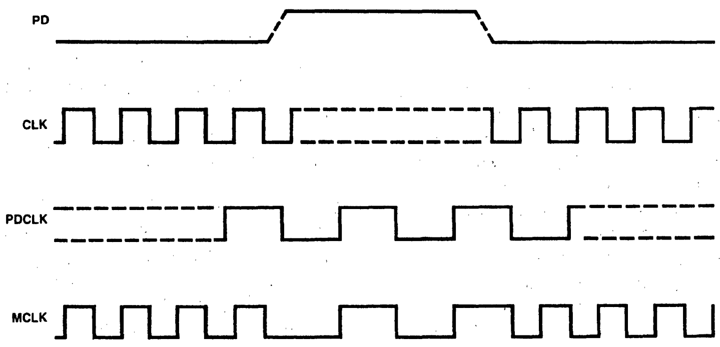
The 82C08 main clock (MCLK) is generated internally, from the system clock (CLK) and the power down clock (PDCLK) (see Figure 10), and is driving the circuits that are active at all times, i.e.: circuits that are active both in power down mode and in normal operation. The system clock (CLK) is driving the circuits that are active in normal operation only, and the PDCLK is driving the circuits that are active in power down only. The operation of the three clocks is as follows:

When entering power down mode, and the whole memory refresh is required, the CLK minimum active time after PDD is activated is 4700 (fast cycle) or 3100 (slow cycle) clocks.

When it isn't required, PDCLK should be active, and CLK should remain active for at least 20 clock cycles + synchronization time. The synchronization time is the ratio of PDCLK and CLK + 1. Therefore, the CLK minimum active time after PD is activated:

20 + [CLK(MHz) / PDCLK(MHz) + 1] clock cycles

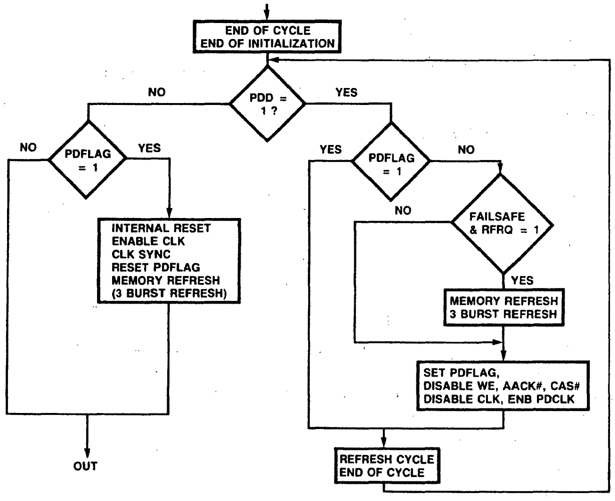
When the power is up again, PDCLK should remain active at least 4 clock cycles after PD is going inactive, to assure completion of refresh cycle and internal synchronization time.



231357-20

Figure 10

POWER DOWN FLOW



231357-21

Figure 11

Differences Between 8208 and 82C08

The differences between the HMOS 8208 and the CHMOS 82C08 represent forward compatible enhancements. The 82C08 can be plugged into an 8208 socket without changes.

LOGICAL DIFFERENCES

1. 82C08 has one new feature:

   Power Down (PD)

2. 82C08 supports CMOS DRAMs with T_{RAC} 100, 150
3. Address Mapping:

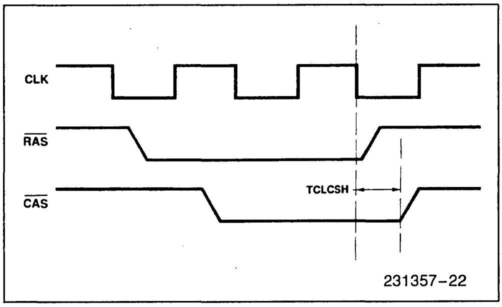
Outputs	9 Most Significant Bits	9 Least Significant Bits
8208 82C08	column address row address	row address column address

4. Slow cycle shortening:
   1). The write cycle is two clocks shorter so consecutive writes will be executed without wait states.
   2) The WE output is two clocks shorter. Therefore, an external latch on the WE output is not necessary.
   3) CAS output is shorter by one clock on the read cycle. This reduces one level of buffers for address/data bus needed in 8208 designs. Read access margins are improved to support non-Intel spec. RAMs.
   4) The address outputs switch from row to column address one clock cycle later in the 82C08 as compared to 8208.
5. Fast cycle shortening:
   1) The write cycle in C0 configuration is shortened by one clock.
   2) For both C0 and C1 synchronous configuration, the CAS signal is shorter by one clock and the activation of RAS is tied to the Φ2 cycle of the 80286. This prevents contention on the data bus.
6. Supports Static Column or Ripplemode DRAMs.

ELECTRICAL DIFFERENCES

1. AC parameters:
   1) $\overline{\text{CAS}}$ delay: In C2 synchronous read cycle, the $\overline{\text{CAS}}$ is deactivated by some delay from clock falling edge (TCLCSH timing) as in the following diagram:

      In C2 write cycles the $\overline{\text{CAS}}$ activation is triggered by the clock falling edge with a delay of 35 ns from the clock. For 8208 the delay is TP/1.8 + 53.



231357-22

   2) 82C08 has an additional timing parameter TARH column address to $\overline{\text{RAS}}$ ↑ hold time.
2. DC parameters: The difference is in the current consumption.

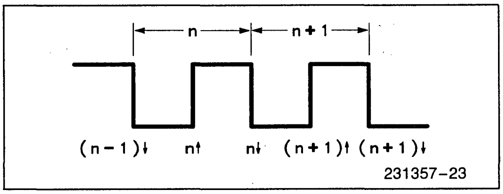
	8208	82C08
I_{CC}	300 mA	30 mA (typical) [10 + 2f] mA (max)
IPD	—	1 mA (max)
I_{SB}	—	2 mA (max)

Configuration Charts

The 82C08 operates in three basic configurations—C0, C1, C2—depending upon the programming of CFS (PD0), $\overline{\text{RFS}}$ (PD2), and $\overline{\text{FFS}}$ (PD7). Table 8 shows these configurations. These modes determine the clock edges for the 82C08's programmable signals, as shown in Table 9. Finally, Table 10 gives the programmable AC parameters of the 82C08 as a function of configuration. The non-programmable parameters are listed under AC Characteristics.

Using the Timing Charts

The notation used to indicate which clock edge triggers an output transition is "n ↑" or "n ↓", where "n" is the number of clock periods that have passed since clock 0, the reference clock, and " ↑ " refers to rising edge and " ↓ " to falling edge. A clock period is defined as the interval from a clock falling edge to the following falling edge. Clock edges are defined as shown below.



231357-23

The clock edges which trigger transitions on each 82C08 output are tabulated in Table 9. "H" refers to the high-going transition, and "L" to low-going transition.

Clock 0 is defined as the clock in which the 82C08 begins a memory cycle, either as a result of a port request which has just arrived, or of a port request which was stored previously but could not be serviced at the time of its arrival because the 82C08 was performing another memory cycle. Clock 0 is identified externally by the leading edge of RAS, which is always triggered on 0 ↓.

Table 8. 82C08 Configurations

Timing Conf.	CFS(PD0)	RFS(PD2)	FFS(PD7)	Wait States*
C_0	iAPX286(1)	FAST RAM(1)	20 MHz(1)	0
C_0	iAPX286(1)	FAST RAM(1)	16 MHz(1)	0
C_1	iAPX286(1)	SLOW RAM(0)	16 MHz(1)	1
C_0	iAPX286(1)	FAST RAM(1)	10 MHz (0)	0
C_0	iAPX286(1)	SLOW RAM(0)	10 MHz (0)	0
C_2	iAPX186(0)	DON'T CARE	DON'T CARE	0

* Using $\overline{EAACK}$ (synchronous mode)

Table 9a. Timing Chart — Synchronous Mode

		$\overline{RAS}$		ADDRESS		$\overline{CAS}$		WE		$\overline{EAACK}$	
Cn	Cycle	L	H	Col	Row*	L	H	H	L	L	H
0	RD,RF	0↓	3↓	0↓	3↓	1↓	3↓			1↓	4↓
	WR	0↓	4↓	0↓	3↓	2↓	4↓	1↓	4↓	1↓	4↓
1	RD,RF	0↓	4↓	0↓	4↓	1↓	5↓			2↓	5↓
	WR	0↓	5↓	0↓	4↓	2↓	5↓	1↓	5↓	2↓	5↓
2	RD,RF	0↓	2↓	0↓	3↓	0↓	2↓			0↓	2↓
	WR	0↓	2↓	0↓	3↓	1↓	3↓	0↓	2↓	0↓	2↓

Table 9b. Timing Chart — Asynchronous Mode

		$\overline{RAS}$		ADDRESS		$\overline{CAS}$		WE		$\overline{LAACK}$		$\overline{XAACK}$	
Cn	Cycle	L	H	Col	Row*	L	H	H	L	L	H	L	H
0	RD,RF	0↓	3↓	0↓	3↓	1↓	4↓			2↓	5↓	3↓	$\overline{RD}$
	WR	0↓	4↓	0↓	3↓	2↓	4↓	1↓	4↓	1↓	4↓	3↓	$\overline{WR}$
1	RD,RF	0↓	4↓	0↓	4↓	1↓	6↓			2↓	5↓	4↓	$\overline{RD}$
	WR	0↓	5↓	0↓	4↓	2↓	5↓	1↓	5↓	1↓	4↓	3↓	$\overline{WR}$
2	RD,RF	0↓	2↓	0↓	3↓	0↓	3↓			1↓	3↓	2↓	$\overline{RD}$
	WR	0↓	2↓	0↓	3↓	1↓	3↓	0↓	2↓	1↑	3↑	2↓	$\overline{WR}$

The only difference between the two tables is the trailing edge of $\overline{CAS}$ for all read cycle configurations. In asynchronous mode, $\overline{CAS}$ trailing edge is one clock later than in synchronous mode.

NOTES FOR INTERPRETING THE TIMING CHART:

1. COLUMN ADDRESS is the time column address becomes valid.
2. The $\overline{CAS}$, $\overline{EAACK}$, $\overline{LAACK}$ and $\overline{XACK}$ outputs are not issued during refresh.
3. $\overline{XACK}$—high is reset asynchronously by command going inactive and not by a clock edge.
4. $\overline{EAACK}$ is used in synchronous mode, $\overline{LAACK}$ and $\overline{XACK}$ in asynchronous mode.
5. ADDRESS-Row is the clock edge where the 82C08 A0 switches from current column address to the next row address.
6. If a cycle is inhibited by PCTL = 1 (Multibus I/F mode) then $\overline{CAS}$ is not activated during write cycle and $\overline{XACK}$ is not activated in either read or write cycles.

*Column addresses switch to row addresses for next memory cycle. The row address buffer is transparent following this clock edge. 'TRAH' specification is guaranteed as per data sheet.

82C08—DRAM Interface Parameter Equations

Several DRAM parameters, but not all, are a direct function of 82C08 timings, and the equations for these parameters are given in the following tables. The following is a list of those DRAM parameters which have NOT been included in the following tables, with an explanation for their exclusion.

WRITE CYCLE

tDS: system-dependent parameter.
tDH: system-dependent parameter.
tDHR: system-dependent parameter.

READ, WRITE REFRESH CYCLES

tRAC: response parameter.
tCAC: response parameter.
tREF: See "Refresh Period Options".
tCRP: must be met only if CAS-only cycles, which do not occur with 82C08, exist.
tRAH: See "A.C. Characteristics"
tRCD: See "A.C. Characteristics"
tASC: See "A.C. Characteristics"
tASR: See "A.C. Characteristics"
tOFF: response parameter.

Table 10. Programmable Timings

Read and Refresh Cycles

Parameter	C2-Slow Cycle	C0-Fast Cycle	C1-Fast Cycle	Notes
tRP	2TCLCL-T27	3TCLCL-T27	3TCLCL-T27	1
tCPN	1.5TCLCL-T34	3TCLCL-T34	2TCLCL-T34	1, 5
tCPN	2.5TCLCL-T34	4TCLCL-T34	3TCLCL-T34	1, 4
tRSH	2TCLCL-T32	2TCLCL-T32	3TCLCL-T32	1
tCSH	3TCLCL-T25	4TCLCL-T25	6TCLCL-T25	1, 5
tCSH	2TCLCL+T34(min)-T25	3TCLCL-T25	5TCLCL-T25	1, 4
tCAH	3TCLCL-T32	2TCLCL-T32	3TCLCL-T32	1
tAR	3TCLCL-T25	3TCLCL-T25	4TCLCL-T25	1
tT	3/30	3/30	3/30	2
tRC	4TCLCL	6TCLCL	7TCLCL	1
tRAS	2TCLCL-T25	3TCLCL-T25	4TCLCL-T25	1
tCAS	3TCLCL-T32	3TCLCL-T32	5TCLCL-T32	1, 5
tCAS	2TCLCL+T34(min)-T32	2TCLCL-T25	4TCLCL-T32	1, 4
tRCS	TCLCL + T32(min)-T35 + TBUF	TCLCL-T35 + TBUF	2TCLCL-T35 + TBUF	1
tRCH	TCLCL + T36(min)-T34 + TBUF	TCLCL-T34 + TBUF	2TCLCL-T34 + TBUF	1

Write Cycles

Parameter	C2-Slow Cycle	C0-Fast Cycle	C1-Fast Cycle	Notes
tRP	2TCLCL-T27	3TCLCL-T27	3TCLCL-T27	1
tCPN	2TCLCL-T34	4TCLCL-T34	4TCLCL-T34	1
tRSH	TCLCL-T32	2TCLCL-T32	3TCLCL-T32	1
tCSH	3TCLCL-T25	4TCLCL-T25	5TCLCL-T25	1
tCAH	2TCLCL-T32	TCLCL-T32	2TCLCL-T32	1
tAR	3TCLCL-T25	3TCLCL-T25	4TCLCL-T25	1
tT	3/30	3/30	3/30	2
tRC	4TCLCL	7TCLCL	8TCLCL	1
tRAS	2TCLCL-T25	4TCLCL-T25	5TCLCL-T25	1
tCAS	2TCLCL-T32 + TBUF	2TCLCL-T32	3TCLCL-T32	1
tWCH	TCLCL-T32 + TBUF	2TCLCL-T32 + TBUF	3TCLCL-T32 + TBUF	1,3
tWCR	2TCLCL-T25 + TBUF	4TCLCL-T25 + TBUF	5TCLCL-T25 + TBUF	1,3
tWP	2TCLCL-T36-TBUF	3TCLCL-T36-TBUF	4TCLCL-T36-TBUF	1
tRWL	2TCLCL-T36-TBUF	3TCLCL-T36-TBUF	4TCLCL-T36-TBUF	1
tCWL	3TCLCL-T36-TBUF	3TCLCL-T36-TBUF	4TCLCL-T36-TBUF	1
tWCS	TCLCL+T36-T31-TBUF	TCLCL−T36-TBUF	TCLCL−T36-TBUF	1

NOTES:
1. Minimum.
2. Value on right is maximum; value on left is minimum.
3. Applies to the eight warm-up cycles during initialization.
4. For synchronous mode only.
5. For asynchronous mode only.

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	0.5W

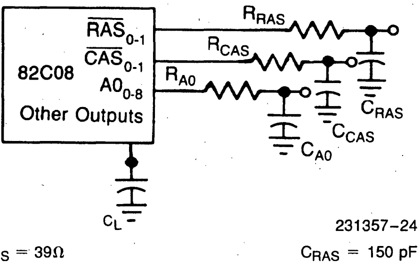
**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.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

D.C. CHARACTERISTICS T_A = 0°C to 70°C; V_{CC} = 5.0V ±10%; V_{SS} = GND

Symbol	Parameter	Min	Max	Units	Comments
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	(Note 1)
V_{OH}	Output High Voltage	2.6		V	(Note 1)
I_{CC}	Supply Current		10 + 2f	mA	(Note 3)
I_{LI}	Input Leakage Current		±10	μA	0V ≤ V_{IN} ≤ V_{CC}
V_{CL}	Clock Input Low Voltage	−0.5	+0.6	V	
V_{CH}	Clock Input High Voltage	3.8	V_{CC} + 0.5	V	
C_{IN}	Input Capacitance		20	pF	fc = 1 MHz(6)
V_{OHPD}	RAS Output High Power Down	V_{CC} − 0.5		V	(Note 2)
I_{PD}	Power Down Supply Current	—	5.0	mA	(Note 5)
I_{SB}	Standby Current	—	2.0	mA	(Note 4)

NOTE:
1. I_{OL} = 5 mA and I_{OH} = −0.32 mA WE: I_{OL} = 8 mA
2. $\overline{RAS}$ Output voltage during power down.
3. Typical value. Where f is freq. in MHz.
for CMOS: V_{IL} max = 0.5V; V_{IH} min = (V_{CC} − 0.5V)
for TTL: I_{CC} will be higher by 30 mA
4. Measured at V_{IL} = 0V and V_{IH} = V_{CC} with no loads connected.
5. IPD = 1 mA at 32 KHz with no loads connected.
6. Sampled, not 100% tested. T_A = 25°C.

A.C. Testing Load Circuit

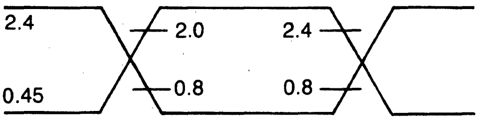


231357-24

R_{RAS} = 39Ω
R_{CAS} = 39Ω
R_{A0} = 22Ω

C_{RAS} = 150 pF
C_{CAS} = 150 pF
C_{A0} = 200 pF
C_L = 50 pF

A.C. Testing Input, Output Waveform



231357-25

A.C. Testing inputs (except clock) are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0" (clock* is driven at 4.0V and 0.45V for Logic "1" and "0" respectively). Timing measurements are made at 2.4V for Logic "1" and 0.8V for Logic "0".
* also PDCLK

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

Measurements made with respect to RAS_{0-1}, CAS_{0-1}, AO_{0-8}, are at +2.4V and 0.8V CLK at 3V, 1V. All other pins are measured at 2.0V and 0.8V. All times are ns unless otherwise indicated. Testing done with specified test load.

Ref	Symbol	Parameter	Min	Max	Units	Notes
CLOCK AND PROGRAMMING						
	tF	Clock Fall Time		12	ns	3
	tR	Clock Rise Time		12	ns	3
1	TCLCL	Clock Period				
		82C08-20	50	250	ns	1
		82C08-16	62.5	250	ns	1
		82C08-10	100	500	ns	2
		82C08-8	125	500	ns	2
2	TCL	Clock Low Time				
		82C08-20	12	230	ns	1
		82C08-16	15	230	ns	1
		82C08-10	44		ns	2
		82C08-8	TCLCL/2-12		ns	2
3	TCH	Clock High Time				
		82C08-20	16	230	ns	1
		82C08-16	20	230	ns	1
		82C08-10	44		ns	2
		82C08-8	TCLCL/3+2		ns	2
4	TRTVCL	Reset to CLK ↓ Setup	40		ns	4
5	TRTH	Reset Pulse Width	4TCLCL		ns	
6	TPGVRTL	PCTL, PDI, RFRQ to RESET ↓ Setup	125		ns	5
7	TRTLPGX	PCTL, RFRQ to RESET ↓ Hold	10		ns	
8	TCLPC	PCLK from CLK ↓ Delay		45	ns	
9	TPDICL	PDI to CLK ↓ Setup	60		ns	
10	TCLPDX	PDI to CLK ↓ Hold	40		ns	6
SYNCHRONOUS μP PORT INTERFACE						
11	TPEVCL	PE to CLK ↓ Setup	30			2
12	TKVCL	$\overline{RD}$, $\overline{WR}$, $\overline{PE}$, PCTL to CLK ↓ Setup	20		ns	1
13	TCLKX	$\overline{RD}$, $\overline{WR}$, $\overline{PE}$, PCTL to CLK ↓ Hold	0		ns	
14	TKVCH	RD, WR, PCTL to CLK ↑ Setup	20		ns	2

A.C. CHARACTERISTICS (Continued)

Ref	Symbol	Parameter	Min	Max	Units	Notes
ASYNCHRONOUS μP PORT INTERFACE						
15	TRWVCL	$\overline{\text{RD}}$, $\overline{\text{WR}}$ to CLK ↓ Setup	20		ns	8.9
16	TRWL	$\overline{\text{RD}}$, $\overline{\text{WR}}$ Pulse Width	2TCLCL+30		ns	
17	TRWLPEV	$\overline{\text{PE}}$ from $\overline{\text{RD}}$, $\overline{\text{WR}}$ ↓ Delay				
		CFS=1		TCLCL-20	ns	1
		CFS=0		TCLCL-30	ns	2
18	TRWLPEX	$\overline{\text{PE}}$ to $\overline{\text{RD}}$, $\overline{\text{WR}}$ ↓ Hold	2TCLCL+30		ns	
19	TRWLPTV	PCTL from $\overline{\text{RD}}$, $\overline{\text{WR}}$ ↓ Delay		TCLCL-30	ns	2
20	TRWLPTX	PCTL to $\overline{\text{RD}}$, $\overline{\text{WR}}$ ↓ Hold	2TCLCL+30		ns	2
21	TRWLPTV	PCTL from $\overline{\text{RD}}$, $\overline{\text{WR}}$ ↓ Delay		2TCLCL-20	ns	1
22	TRWLPTX	PCTL to $\overline{\text{RD}}$, $\overline{\text{WR}}$ ↓ Hold	3TCLCL+30		ns	1
RAM INTERFACE						
23	TAVCL	AL, AH, BS to	35+tASR		ns	2
		CLK ↓ Set-up				
		82C08-20	50+tASR		ns	
		82C08-16	45+tASR		ns	
24	TCLAX	AL, AH, BS to CLK ↓ Hold	0		ns	
25	TCLRSL	RAS ↓ from		25	ns	1
		CLK ↓ Delay		35	ns	2
				60	ns	24
26	TRCD	$\overline{\text{RAS}}$ to $\overline{\text{CAS}}$ Delay				
		CFS=1	TCLCL-25		ns	1, 14
		CFS=0	30		ns	23
		CFS=0	TCLCL/2-30		ns	2, 11, 14
		CFS=0	60		ns	2, 12, 14
27	TCLRSH	$\overline{\text{RAS}}$ ↑ from		25	ns	
		CLK ↓ Delay		60	ns	24

A.C. CHARACTERISTICS (Continued)

Ref	Symbol	Parameter	Min	Max	Units	Notes
RAM INTERFACE (Continued)						
28	TRAH	CFS = 1	18			1, 13, 15
		CFS=0	TCLCL/4-10		ns	2, 11, 15
		CFS=0	18		ns	23
29	TASR	Row A0 $\overline{RAS}$ ↓ Setup				10, 16
30	TASC	Column A0 to $\overline{CAS}$ ↓ Setup				
		CFS=1	2		ns	1, 13, 17, 18
		CFS=0	5		ns	2, 13, 17, 18
		CFS=0	5		ns	23
31	TCAH	Column A0 to $\overline{CAS}$ Hold	(See DRAM Interface Tables)			
32	TCLCSL	$\overline{CAS}$ ↓ from CLK ↓ Delay				
		CFS=0	TCLCL/4+30	TCLCL/1.8+56	ns	2, 26
		CFS=0	50	105	ns	23, 26
		CFS=0	8	35	ns	2, 23, 27
		CFS=1		35	ns	1
34	TCLCSH	$\overline{CAS}$ ↑ from CLK ↓ Delay		50	ns	
			TCLCL/4	$\frac{TCLCL}{3.2}+50$	ns	22
35	TCLWL	WE ↓ from CLK ↓ Delay		35	ns	
36	TCLWH	WE ↑ from CLK ↓ Delay				
		CFS=0	TCLCL/4+30	TCLCL/1.8+53		2
		CFS=1		35	ns	1
		CFS=0	50	100	ns	23
37	TCLTKL	$\overline{XACK}$ ↓ from CLK ↓ Delay		35	ns	
38	TRWLTKH	$\overline{XACK}$ ↑ from $\overline{RD}$ ↑, $\overline{WR}$ ↑ Delay		50	ns	
39	TCLAKL	$\overline{AACK}$ ↓ from $\overline{CLK}$ ↓ Delay		35	ns	
40	TCLAKH	$\overline{AACK}$ ↑ from CLK ↓ Delay		50	ns	
49	TARH	Column Address to $\overline{RAS}$ ↑ Hold Time	2			1

A.C. CHARACTERISTICS (Continued)

Ref	Symbol	Parameter	Min	Max	Units	Notes
REFRESH REQUEST						
41	TRFVCL	RFRQ to CLK ↓ Setup	20		ns	
42	TCLRFX	RFRQ to CLK ↓ Hold	10		ns	
43	TFRFH	Failsafe RFRQ Pulse Width	TCLCL + 30		ns	19
44	TRFXCL	Single RFRQ Inactive to CLK ↓ Setup	20		ns	20
45	TBRFH	Burst RFRQ Pulse Width	2TCLCL + 30		ns	19
46	TPDDVCL	PDD Setup Time	20		ns	24, 25
47	TPDHRFX	RFRQ Valid after PDD Active	4TCLCL + 20			24
48	TRFVPDH	RFRQ Setup Time to PDD Active	20			24

The following RC loading is assumed:

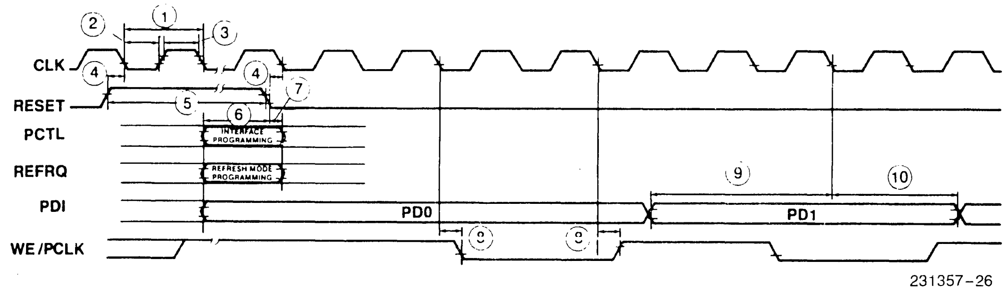
$A0_{0-8}$	R = 22Ω	C = 200 pF
$\overline{RAS}_{0-1}$, CAS_{0-1}	R = 39Ω	C = 150 pF
$\overline{AACK}$, WE/PCLK		C = 50 pF

NOTES:

1. Specification when programmed in the Fast Cycle processor mode (iAPX 286 mode). 82C08-20, -16.
2. Specification when programmed in the Slow Cycle processor mode (iAPX 186 mode). 82C08-10, 82C08-8.
3. tR and tF are referenced from the 3.5V and 1.0V levels.
4. RESET is internally synchronized to CLK. Hence a set-up time is required only to guarantee its recognition at a particular clock edge.
5. The first programming bit (PD0) is also sampled by RESET going low.
6. TCLPDX is guaranteed if programming data is shifted using PCLK.
8. TRWVCL is not required for an asynchronous command except to guarantee its recognition at a particular clock edge.
9. Valid when programmed in either Fast or Slow Cycle mode.
10. tASR is a user specified parameter and its value should be added accordingly to TAVCL.
11. When programmed in Slow Cycle mode and 125 ns ≤ TCLCL < 200 ns.
12. When programmed in Slow Cycle mode and 200 ns ≤ TCLCL.
13. Specification for Test Load conditions.
14. tRCD (actual) = tRCD (specification) +0.06 (ΔC_{RAS}) − 0.06(ΔC_{CAS}) where ΔC = C (test load) − C (actual) in pF. (These are first order approximations.)
15. tRAH (actual) = tRAH (specification) + 0.06 (ΔC_{RAS}) − 0.022 (ΔC_{A0}) where ΔC = C (test load) − C (actual) in pF. (These are first order approximations.)
16. tASR (actual) = tASR (specification) +0.06 (ΔC_{A0}) − 0.025 (ΔC_{RAS}) where ΔC = C (test load) − C (actual) in pF. (These are first order approximations.)
17. tASC (actual) = tASC (specification) +0.06 (ΔC_{A0}) − 0.025 (ΔC_{CAS}) where ΔC (test load) − C (actual) in pF. (These are first order approximations.)
18. tASC is a function of clock frequency and thus varies with changes in frequency. A minimum value is specified.
19. TFRFH and TBRFH pertain to asynchronous operation only.
20. Single RFRQ should be supplied synchronously to avoid burst refresh.
22. CFS = 0, synchronous mode, Read cycle.
23. For 10 MHz Slow Cycle only.
24. Power down mode.
25. PDD is internally synchronized. A setup time is required only to guarantee its recognition at a particular clock edge.
26. Slow Cycle Read only.
27. Slow Cycle Write only.

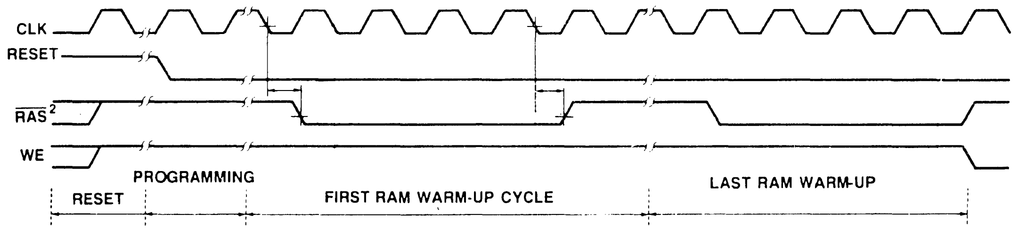
WAVEFORMS

CLOCK AND PROGRAMMING TIMINGS



231357–26

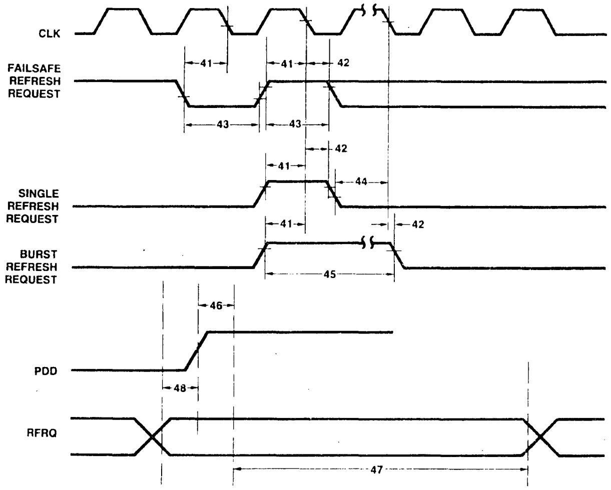
RAM WARM-UP CYCLES



231357–27

NOTE:
The present example assumes a $\overline{\text{RAS}}$ four clocks long.

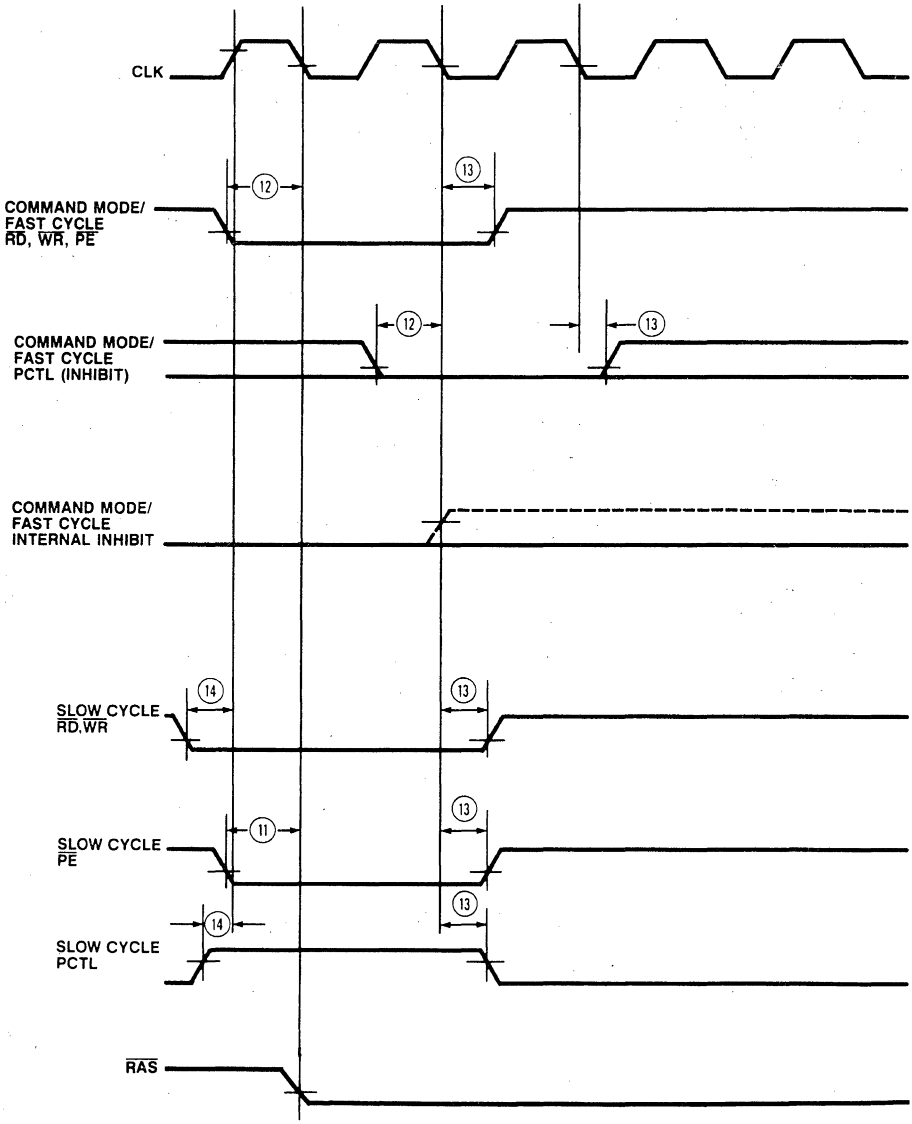
REFRESH REQUEST TIMING



231357–28

WAVEFORMS (Continued)

SYNCHRONOUS PORT INTERFACE

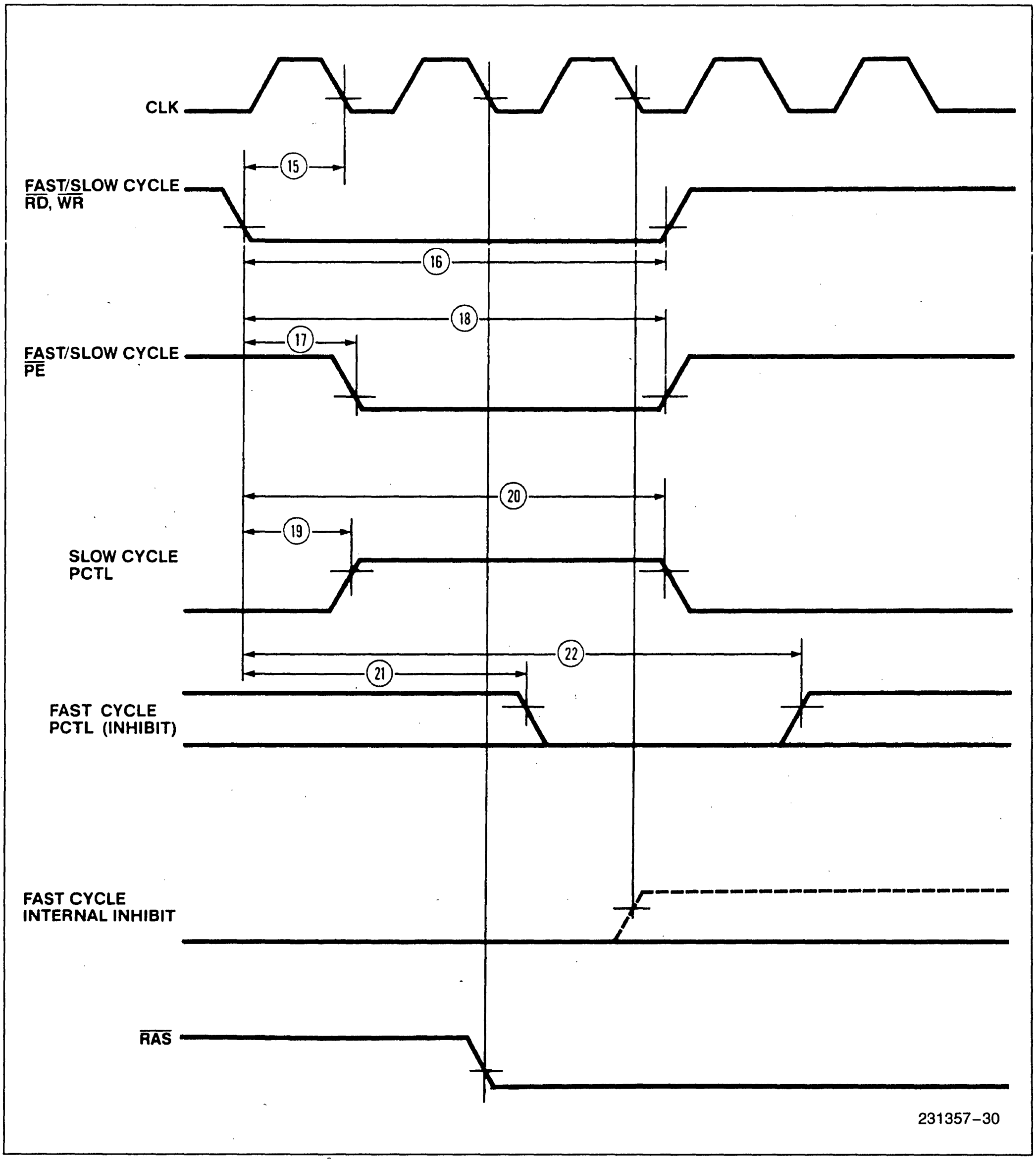


231357–29

NOTE:
Actual transitions are programmable. Refer to Tables 8 and 9.

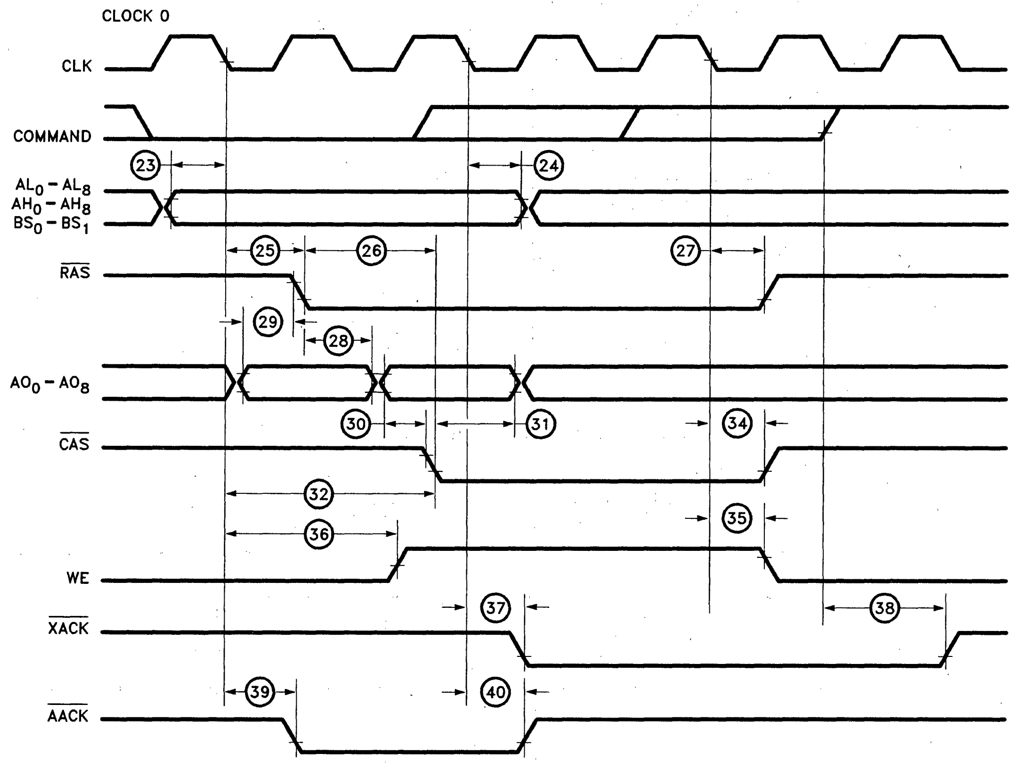
WAVEFORMS (Continued)

ASYNCHRONOUS PORT INTERFACE



WAVEFORMS (Continued)

RAM INTERFACE TIMING



231357-31

NOTE:
Actual transitions are programmable. See Tables 8 and 9.

November 1986

Interfacing the 8207 Dynamic RAM Controller to the iAPX 186

JIM SLEEZER
APPLICATION ENGINEER

Order Number: 230809-001

INTRODUCTION

Most microprocessor based workstation designs today use large amounts of DRAM for program storage. A drawback to DRAMs is the many critical timings that must be met. This control function could easily equal the area of the DRAM array if implemented with discrete logic.

The VLSI 8207 Advanced Dynamic RAM Controller (ADRC) performs complete DRAM timing and control. This includes the normal RAM 8 warm-up cycles, various refresh cycles and frequencies, address multiplexing, and address strobe timing. The 8207's system interface and RAM timing and control are programmable to permit it to be used in most applications.

Integrating all of the above functions (plus a dual port and error correcting interfaces) allows the user to realize significant cost savings over discrete logic. For example, comparing the 8207 to the iSBC012B 512K byte RAM board (where the DRAM control is done entirely with TTL), an 8207 design saved board space 3 in^2 vs 10 in^2); required less power (420 mA vs 1220 mA); and generated less heat. Moreover, design time was reduced, and increased margins were achieved due to less skewing of critical timings. This comparison is based on a single port design and did not include the 8207's RAM warm-up, dual-port and error correcting features. If these features were fully implemented, there would be no change to the 8207 figures, listed above, while the TTL figures would easily double.

This Application Note will illustrate an iAPX design with the 8207 controlling the dynamic RAM array. The reader should be familiar with the 82097 data sheet, the 80186 data sheet, and a RAM data sheet*.

DESIGN GOALS

The main objective of this design is for the 80186 to run with no wait states with a Dynamic RAM array. The design uses one port of the 8207. The dual port and error correcting interfaces of the 8207 are covered in separate Application Notes.

The size of the RAM array is 4 banks of 64K RAMs or 512K bytes. The memory is to be interfaced locally to the 80186.

USING THE 8207

The three areas to be considered when designing in the 8207 are:

- 8207 programming logic
- Microprocessor interface
- RAM array

8207 Programming

The 8207 requires up to two 74LS165 shift registers for programming. This design needs one 8 bit shift register, as shown in Figure 1. The 16 bits in the Program Data Word are set as shown in Figure 2. Refresh is done internally, so the REFRQ input must be tied high. The memory commands are iAPX 86 status, so the timing

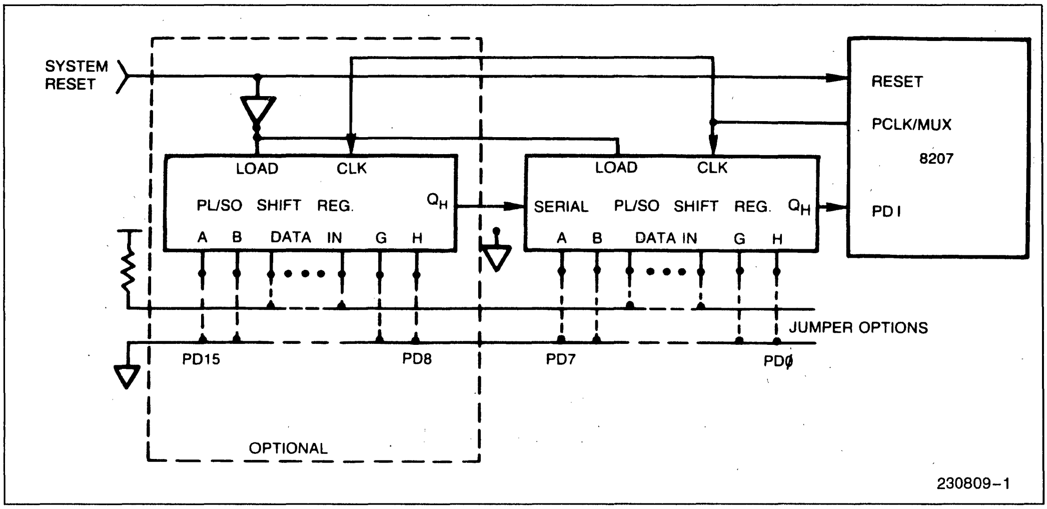
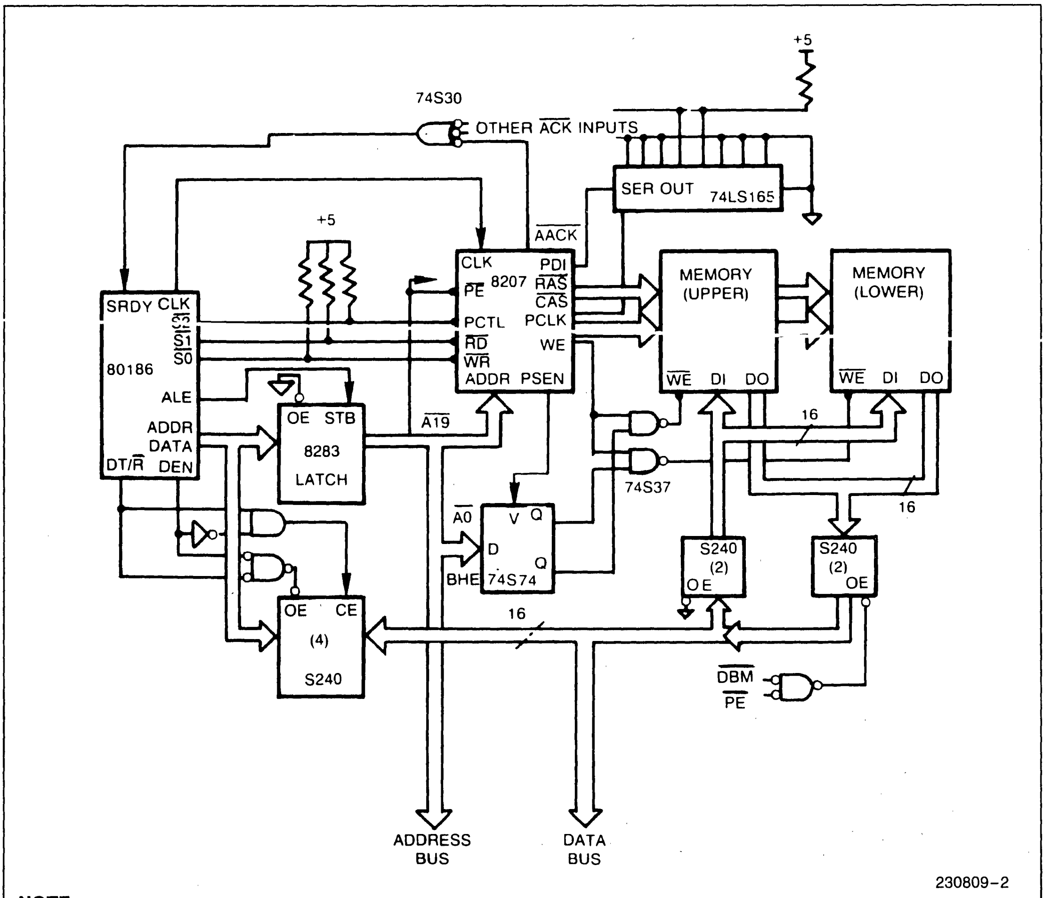


Figure 1. 8207 Programming Shift Registers

*All RAM references in this Application Note are based on Intel's 2164A 64K Dynamic RAM.



NOTE:
The 8207 requires series resistors on all outputs to RAM.

Figure 3. 80186 to 8207, non-ECC, synchronous system single port

of EAACK will always guarantee 2 clocks of address hold time from RAS.

Acknowledge Setup Time

The margin between the 8207 issuing EAACK and the 80186 ready input for no wait states minus delays from clock edges, logic delays, and setup time is calculated as follows.

1 clock − 8207 TCLAKL max − 74S30 t_{PLH} @ 15 pf − 80186 TSRYCL ≥ 0

125 ns − 35 − 22 − 35 = 32 ns

Read Access Margin

The 8207 starts a memory cycle on the falling clock edge between the 80186's T1 and T2. Data must be valid within 2 clocks. Valid data from the RAMs is based upon the CAS access period minus buffer, clock, setup requirements.

2 TCLCL − 8207 TCLCSL @ 150 pF (t34) − DRAM tCAC − 74S240 propagation delay @ 50 pF − additional bus loading delay (250 pF)(1) − 74S240 delay @ 50 pF − 80186 TDVCL ≥ 0

250 ns − 122 − 85 − 7 − 7 − 7 − 20 = 2 ns

NOTE:
(1) 74STTL logic derated by 0.05 ns/pF. 74STTL buffers (240, 37) derated by 0.025 ns/pF.

0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
15							8	7							0

230809-3

Figure 2. Program Data Word

Write Data Setup and Hold Margin

Data from the processor must be valid when WE is issued by the 8207 to meet the RAM specification tDS (2164A = 0 ns), and then held for a minimum of 30 ns.

The PCTLA input must be high when RESET goes inactive.

The differential reset circuit shown in the Data Sheet is necessary only to ensure that memory commands are not received by the 8207 when Port A is changed from synchronous to asynchronous (vice versa for Port B). This design keeps Port A synchronous so no differential reset circuit is needed.

Microprocessor Interface

To achieve no wait states, the 8207 must connect directly to the microprocessor's CLKOUT and status lines. The 8207 Acknowledge (EAACK) must connect to the SRDY input of the 80186.

When the 80186 is reset, it tristates the status lines. The 8207 PCTLA input requires a high to decode the proper memory commands. This is accomplished by using a pull-up resistor or some component that incorporates a pull-up on S2.

The 8207 address inputs are connected directly to the latched/demultiplexed address bus.

RAM Array

The 8207 provides complete control of all RAM timings, warm up cycles, and refresh cycles. All write cycles are "late writes." During write cycles, the data out lines go active. This requires separate data in/out lines in the RAM array.

To operate the 80186 with no wait states, it is necessary to choose sufficiently fast DRAMs. The 150 ns version of the 2164A allows operating the 80186 at 8 MHz, and the 200 ns version up to 7 MHz.

HARDWARE DESIGN

Figure 3 shows a block diagram of the design, and Figure 4 is a timing diagram showing the relationship between the 8207 and the 80186.

8207 Command Setup

Two events must occur for a command to be recognized by the 8207. The 80186 status outputs are sampled by a rising clock edge and Port Enable (PE) is sampled by the next falling edge (refer to the Data Sheet wave forms).

The command timing is determined by the period between the status being issued and the first rising clock edge of the 8207, minus setup and delays.

80186 status valid to 8207 rising clock − status from clock delay − 8207 setup to clock ≥ 0

1 TCLCL − 80186 TCHSV max − 8207 TKVCH min ≥ 0

125 ns − 55 − 20 = 50 ns

PE is a chip select for a valid address range. It can be generated from the address bus or from the 80186's programmable memory selects. This design uses an inverted A19. The timing is determined by the interval between the address becoming valid and the falling clock edge, minus setup and delays.

80186 address valid to 8207 falling clock edge − 80186 address from clock delay − 8283 latch delays − 8207 PE setup ≥ 0

1 TCLCL − 80186 TCLAV max − 8283 IVOV @ 300 pF − 8207 TPEVCL ≥ 0

125 ns − 44 − 22 − 30 − 29 ns

The hold times are 0 ns and are met.

Address Setup

For an 81086 design, the 8207 requires the address to be stable before RAS goes active, and to remain stable for 2 clocks. Unused 8207 address inputs should be tied to V_{CC}.

tASR is a RAM specification. If it is greater than zero, this must be added to the address setup time of the 807. Address setup is the interval between addresses being issued and RAS going active, minus appropriate delays.

80186 address valid to 8207 RAS active − 80186 address from clock delay − bus delays − (8207 setup + RAM t_{ASR}) ≥ 0

TCLCL + 8207 TCLRSL min @ 150 pf(1) − 80186 TCLAV max − 8283 IVOV max @ 300 pf − (8207 TAVCL min + DRAM t_{ASR}) ≥ 0

125 ns + 0 − 44 − 22 − (35 + 0) = 24 ns

The address hold time of 2 clocks + 0 ns is always met, since the addresses are latched by the 8282/3. Even when the processor is in wait states (for refresh),

NOTE:
(1) Not specified—use 0 ns.

TCLCL + TCLCH + 8207 TCLW min(1) + 74S37 delay tPHL min @ 50 pf + additional loading (142 pf) − 81086 TCVCTV − 74S240tPZL − bus delays (250 pf) − 74S240 delay − 2164A tDS ≥ 0

125 + 62.5 + 0 + 6.5 + 3.5 − 70 − 15 − 7 − 7 − 0 = 98.5 ns.

The hold time, tDH, is from WE going low to the 80186 DEN going high plus buffer delays minus WE from clock delays.

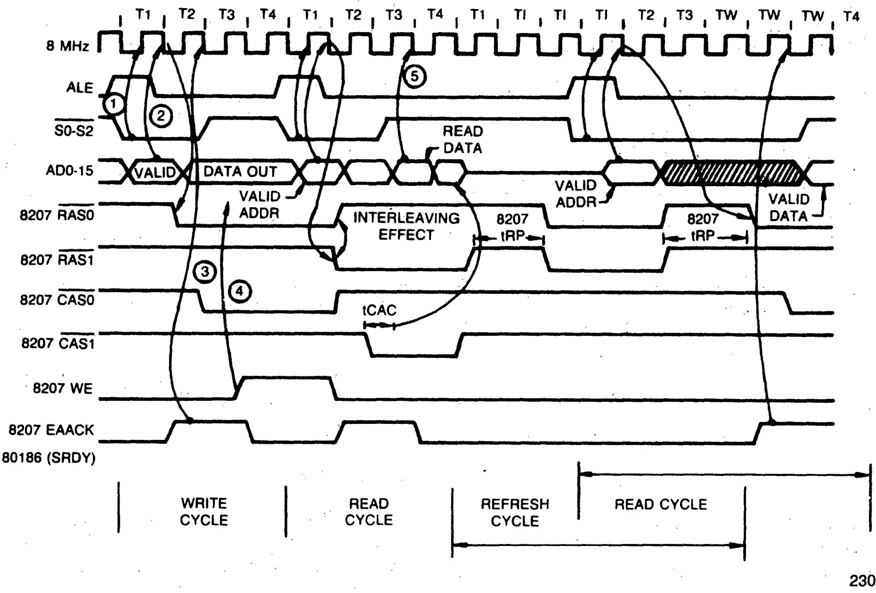
TCLCL − 80186 TCVCTX min + 74S32 tPD(2) min + 74S240 tPHZ (min)(2) + 250 pf bus delays + 74S240 propagation delay min − 8207 TCLW max − 74S37 tPHL @ 50 pf − 142 pf loading delays − DRAM tDH ≥ 0

62.5 ns + 10 + 2 + 3 + 7 + 3.5 − 35 − 3.5 − 30 = 19.5 ns

All margins are actually better by about 10–20 ns. No improvement in timing was allowed for lower capacitive loads when additional buffers are used (i.e. the 80186 address out delay is at 200 pf, but the 8283 latch only loads these lines with about 20 pf).

SUMMARY

The 8207 supports the 80186 microprocessor running with no wait states. The 8207 interfaces easily between the microprocessor and dynamic RAM. There are no difficult timings to be resolved by the designer using external logic.



NOTES:

1. Command Setup Margin
2. PE Setup Margin
3. EAACK Setup Margin
4. Data Setup Margin
5. Read Access Margin

Figure 4. 8207/80186 Timing Relationship

NOTE:

1. Not specified, use 0 ns.
2. Not specified, use one half of typical value.

APPLICATION NOTE

AP-168

November 1986

Interfacing the 8207 Advanced Dynamic RAM Controller to the iAPX 286

JIM SLEEZER
APPLICATION ENGINEER

Order Number: 230862-001

INTRODUCTION

The 80286 high speed microprocessor pushes microprocessor based systems to new performance levels. However, its high speed bus requires special design considerations to utilize that performance. Interfacing the 80286 to a dynamic RAM array require many timings to be analyzed, refresh cycle effects on bus timing examined, minimum and maximum signal widths noted, and the list continues.

The 8207 Advanced Dynamic RAM Controller was specifically designed to solve all interfacing issues for the 80286, provide complete control and timing for the DRAM array, plus achieve optimum system performance. This includes the normal RAM 8 warmup cycles, various refresh cycles and frequencies, address multiplexing, and address strobe timings. The 8207 Dynamic RAM Controller's system interface and RAM timing and control are programmable to permit it to be used in most applications.

Integrating these functions (plus dual port and error correcting interfaces) allows the user to realize significant savings in both engineering design time, PC board space and product cost. For example, in comparing the 8207 to the ISBC012B 512k byte RAM board (where the DRAM timing and control is done entirely with TTL), the 8207 design saved board space (3 in^2 vs 10 in^2); used less power (420 mA vs 1220 mA); reduced the design time; and increased margins due to less skewing of timings. The comparison is based upon a single port 8207 design and does not include its RAM warm-up, dual port, error correcting, and error scrubbing or RAM interleaving features.

This Application Note will detail an 80286 and 8207 design. The reader should have read the 8207 and the 80286 data sheets, a DRAM data sheet*, and have them available for reference.

DESIGN GOALS

The main objective of this design is to run the RAM array without wait states, to maximize the 80286's performance, and to use as little board space as possible. The 80286 will interface synchronously to Port A of the 8207 and the 8207 will control 512k bytes of RAM (4 banks using 64k DRAMs). The dual port and error correcting features of the 8207 are covered in separate Application Notes.

8207 INTERFACE

The 8207 Memory design can be subdivided into three sections:

- Programming the 8207.
- The 82086/8207 interface.
- The Dynamic RAM array.

Programming the 8207

The RAM timing is configured via the 16 bit program word that the 8207 shifts-in when reset. This can require two 74LS165 shift registers to provide complete DRAM configurability. The 8207 defaults to the configuration shown in Table 1 when PDI is connected to ground. This design does not need the flexibility the shift registers would allow since standard 8207/80286 clock frequencies, DRAM speeds and refresh rates are used. Table 1 details the 8207/80286 configuration and Table 10 in the Data Sheet identifies "CO" as the configuration of the 8207 all timings will be referenced to (80286 mode at 16 MHz using fast RAMs = CO).

Table 1. Default Non-ECC Programming, PD1 Pin (57) Tied to Ground

Port A is Synchronous (EAACKA and XACKA)
Port B is Asynchronous (LAACKB and XACKB)
Fast-cycle Processor Interface (10 or 16 MHz)
Fast RAM 100/120 ns RAM
Refresh Interval uses 236 clocks
128 Row refresh in 2 ms; 256 Row refresh in 4 ms
Fast Processor Clock Frequency (16 MHz)
"Most Recently Used" Priority Scheme
4 RAM banks occupied

The 8207 will accept 80286 status inputs when the PCTLA pin is sampled low at reset. This pin is not necessary for an 80286 design (besides programming) and is tied to ground.

Refresh is the final option to be programmed. If the Refresh pin is sampled high at reset, an internal timer

*All RAM references in this Application Note are based upon Intel's CMOS 51C64-12 64k Dynamic RAM. Any DRAM with similar timings will function. Refer to section 4.4.

is enabled, and if low at reset, this timer is disabled. The first method is the easiest to implement, so the RFRQ pin is tied to V_{CC}.

The differential reset circuit shown in the Data Sheet is necessary only to ensure that memory commands are not received by the 8207 when Port A is changed from synchronous to asynchronous (vice versa for Port B). This design keeps Port A synchronous so no differential reset circuit is needed.

RAM Array

The 8207 completely controls all RAM timings, warm-up cycles, and refresh cycles. To determine if a particular RAM will work with the 8207, calculate the margins provided by the 8207 (Table 15, 16—8207 Data Sheet) and ensure they are greater than the RAM requirement. An additional consideration is the access times of the RAMs. The access time of the system is dependent upon the number of data buffers between the 80286 and the DRAMs. To operate the 80286 at zero wait states requires access times of 100–120 ns. Slower RAMs can be used (150 ns) by either adding a wait state (programming the 8207 for "C1") or reducing the clock frequency (to 14.9 MHz approximately and maintaining the CO configuration).

All write cycles are "late writes" and the data out lines of the RAM will go active. This will require separate data in and out lines in the RAM array. Another consideration for the RAM array is the proper layout of the RAM, and impedance matching resistors on the 8207 outputs. Proper layout is covered in Intel's RAM Data Sheets and Application Notes.

Microprocessor Array

To achieve no wait state operation, the 8207's clock input must be connected to the 80286's clock input. The EAACK (early acknowledge) output of the 8207 must connect to the SRDY input of the 82284. The 8207's address inputs connect directly to the 80286 address outputs and the addresses are latched internally. This latch is strobed by an internal signal with the same timing as LEN (which is for dual port 82086 designs). Figure 2 shows the timing relationship between LEN and the 80286.

LEN will fall from high to low, which latches the bus address internally, when a valid command is received. LEN can go high in two clock cycles if the RAM cycle started (RAS going low) at the same time LEN went low. If the 8207 is doing a refresh cycle, the 80286 will be put into wait states until the memory cycle can start. LEN will then go high two clocks after RAS starts, since addresses are no longer needed for the current RAM cycle. Thus the low period of LEN could be much longer than listed in the Data Sheet.

DESIGNING THE HARDWARE

Figure 1 shows a detailed block diagram of the design and Figure 2 shows the timing relationship between the 8207 and the 80286.

The following analysis of six parameters will confirm that the design will work. These six system parameters are generally considered the most important in any microprocessor—Dynamic RAM design.

8207 Command Setup Margin

Two events must occur for the 8207 to start a memory cycle. Either RD or WR active (low) and PE must be low when the 8207 samples these pins on a falling clock edge. If PE is not valid at the same clock edge that samples RD or WR active, the memory cycle will be aborted and no acknowledge will be issued.

The command setup time is based upon the status being valid at the first falling clock edge.

80286 status valid to 8207 falling clock − 80286 status from clock delay − 8207 command setup to clock ≤ 0

TCLCL − 80286 t12 (max) − 8207 TKVCL (min) ≤ 0

62.5 − 40 ns − 20 ns = 2.5 ns

PE is decoded from the address bus and must be set up to the same falling clock edge that recognizes the RD, WR inputs. This margin is determined from the clock edge that issues the address and the clock edge that will recognize RD or WR, minus decoding logic delays.

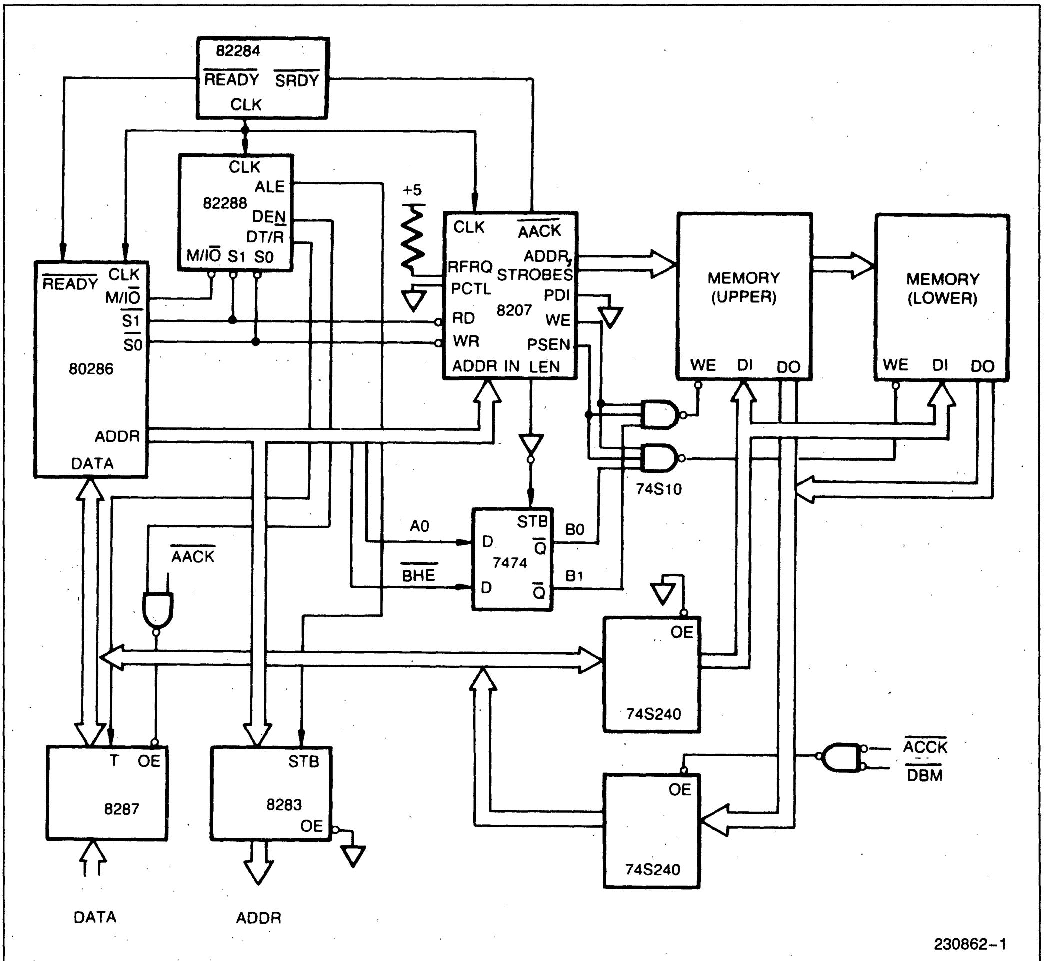
There are 2 clocks between addresses being issued by the 80286 and PE being sampled by the 8207. Then the 80286 address delay from the clock edge and decoding logic delays are subtracted from this interval. This margin must be greater than 0.

2TCLCL − 80286 t13 (max) − 8207 TPEVCL (min) ≤ 0

125 − 60 − 30 = 35 ns

The address decode logic must use no more than 35 ns (and less is better). Figure 3 shows an easy implementation which uses a maximum of 12 ns.

The 8207 requires a zero ns hold time and is always met.



NOTE:
The 8207 requires series resistors on all outputs.

Figure 1. 80286 to 8207, non-ECC, Synchronous System Single Port

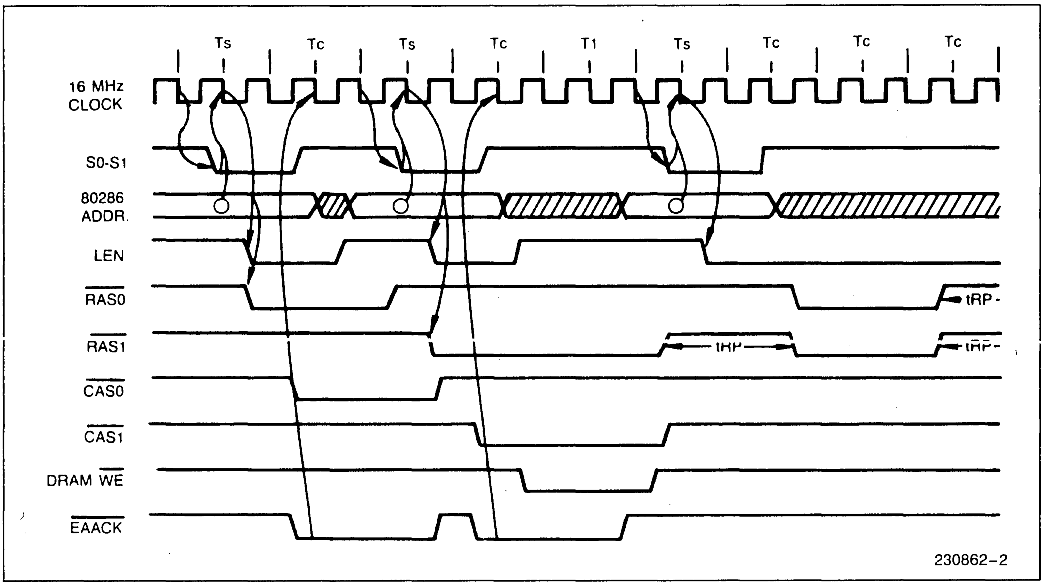


Figure 2. 80286/8207 Timing—"CO"

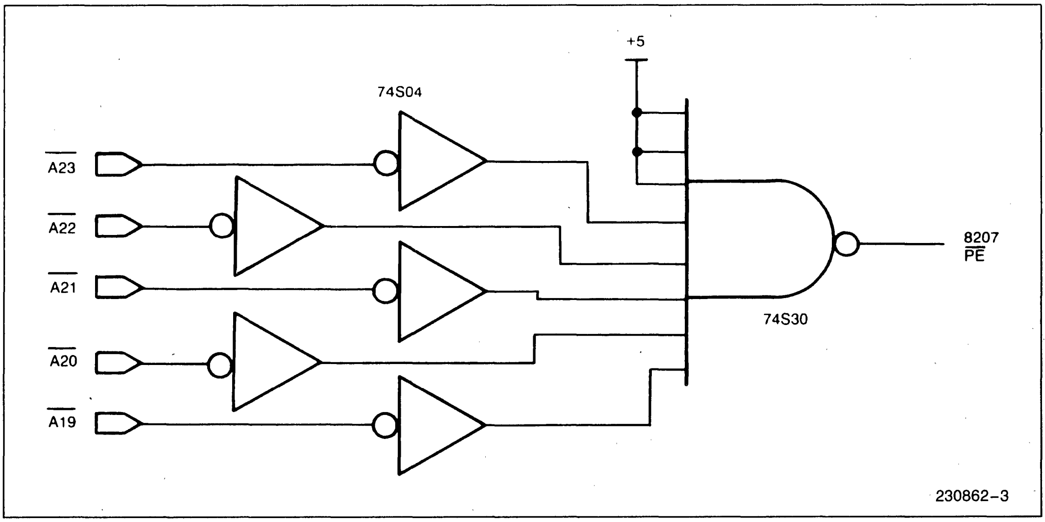


Figure 3. Address Decode Logic

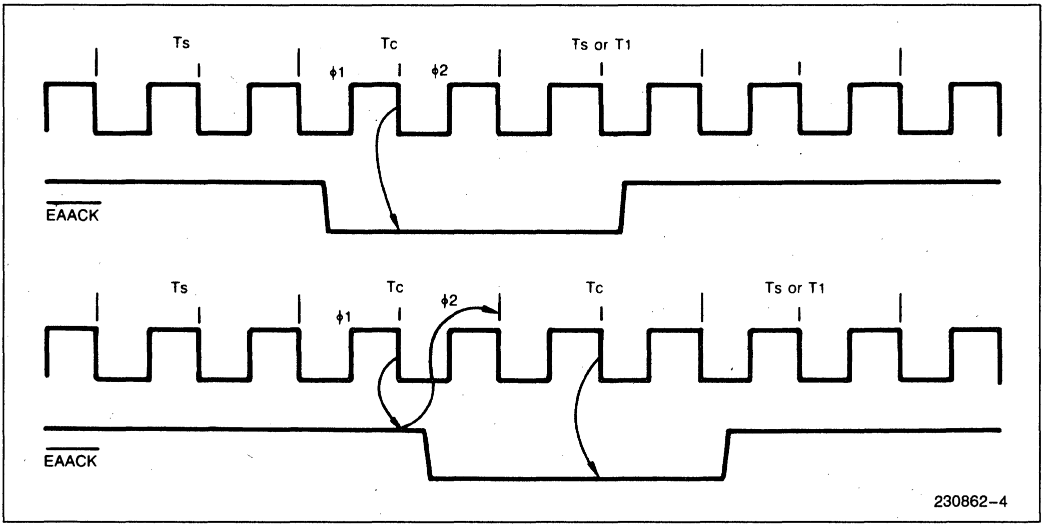


Figure 4. Acknowledge to the 82284

Address Setup Margin

The 8207 must have stable addresses up to two clocks after RAS goes active. This is of no concern to the user, since LEN latches the address internally and will not admit a new address until two clocks after RAS goes active.

Addresses must be stable at least 35 ns (tAVCL) before RAS goes active to allow for propagation delays through the 8207, if a RAM cycle is not delayed by the 8207.

tASR is a RAM specification. If it is greater than zero, tASR must be added to the address setup time of the 8207. Address setup is the interval between addresses being issued, by the 80286, and RAS going active, minus appropriate delays.

The margin is determined from the number of clocks between addresses being issued from the 80286 to RAS going active. Exactly when RAS goes active is unimportant, since here we are only interested in the clock edge.

2TCLCL − 80286 t13 (max) − 8207 TAVCL (min) ≤ 0

125 ns − 60 ns − 35 ns = 30 ns

Acknowledge Setup Margin

The 8207 acknowledge (EAACK) can be issued at any point in the 80286 bus cycle (end of φ1 or φ2 of Ts or Tc). If EAACK is issued at the end of φ2 (Ts or Tc), the 80286 will complete the current bus cycle. If EAACK is issued at the end of φ1 of Tc, the 82284 will not generate READY to the 80286 in time to end the current bus cycle. A new Tc would then be generated and EAACK would now be sampled in time to terminate the bus cycle. EAACK is 3 clocks long in order to meet setup and hold times for either condition.

We need the margin between the 8207 issuing EAACK and the 82284 needing it. Figure 4 shows a worst case example.

TCLCL − 8207 TCLAKL max − 82284 t11 ≤ 0

62.5 ns − 35 ns − 15 ns = 12.5 ns

Read Access Margin

The 8207 will typically start a memory cycle (i.e. RAS goes low) at the end of φ1 of Ts. But if the start of a memory cycle is delayed (by a refresh cycle for instance), then RAS will be delayed. In the first case, this represents 3 clocks and the second case could require 4 clocks to meet the data setup requirements of the 80286. In either case, data must be valid at the end of Tc. The 8207 holds CAS active long enough to ensure valid data is received by the 80286 in either case.

DRAMs specify two access times, RAS access (tRAC) and CAS access (tCAC) Both access periods must be calculated and the one with the least margin used. Also the number of data buffers should be kept to a minimum. Too many buffers would require either faster (more expensive) DRAMs, or a reduction in the performance of the CPU (by adding wait states).

RAS Access Margin

3TCLCL − 8207 TCLRSL max @ 150 pF − DRAM tRAC − 74S240 propagation delay max @ 50 pF − 80286 t8 ≤ 0

187.5 ns − 35 ns − 120 ns − 7 ns − 10 ns = 15.5 ns

CAS Access Margin

2TCLCL − 820-7 TCLCSL max @ 150 pF − DRAM tCAA (or tCAC − 74S240 tplh max @ 50 pF − 80286 t8) ≤ 0

125 ns − 35 ns − 60 ns − 7 ns − 10 ns = 13 ns

By solving each equation for tRAC and tCAC, the speed requirement of the RAM can be determined.

DRAM tRAC = 3 TCLCL − 8207 TCLRSL − 74S240 tplh − 82086 t8 = 135.5 ns

DRAM tCAC = 2 TCLCL − 8207 TCLCSL − 74S240 tplh − 80286 t8 = 73 ns

NOTES:
1. Not specified. Assume no delay for worst case analysis.
2. STTL derated by 0.05 ns/pF.

So any DRAM that has a RAS access period less than 135 ns, a CAS access period less than 73 ns, and meets all requirements in the DRAM Interface Timing (Table 15, 16—8207 Data Sheet), will work.

Write Data Setup and Hold Margin

Write data from the processor must be valid when the 8207 issues WE to meet the DRAM specification tDS and then held to meet the tDH requirement. Some write cycles will be byte writes and the information to determine which byte is decoded from A0 and BHE/. Since the 80286's address bus is pipelined, these two signals can change before the RAM cycle starts, hence they must be latched by LEN. PSEN is used in the WE term to shorten the WE pulse. Its use is not essential.

Data must be set up to the falling edge of WE, since WE occurs after CAS. The 2 clocks between valid write data and WE going active (at the RAM's) minus propagation delays determines the margin.

2 TCLCL − 80286 t14 (max) @ 100 pF − 74S240 tplh + 8207 TCLW (min)^1 + 74S10 tphl @ 192 pF^2 − DRAM tDS = 0

125 ns − 50 ns − 7 ns + 0 ns + 14 ns − 0 ns = 82 ns

The timing of the 8207's acknowledge is such that data will be kept valid by the 80286, for more than two clocks after WE goes active. This easily meets all RAM tDH specifications.

SUMMARY

The 8207 complements the 80286's performance and high integration with its own performance, integration and ease of use. No critical timings or logic design has been left to the designer. The 80286/8207 combination allows users to realize maximum performance from their simpler design.

Support Peripherals

6

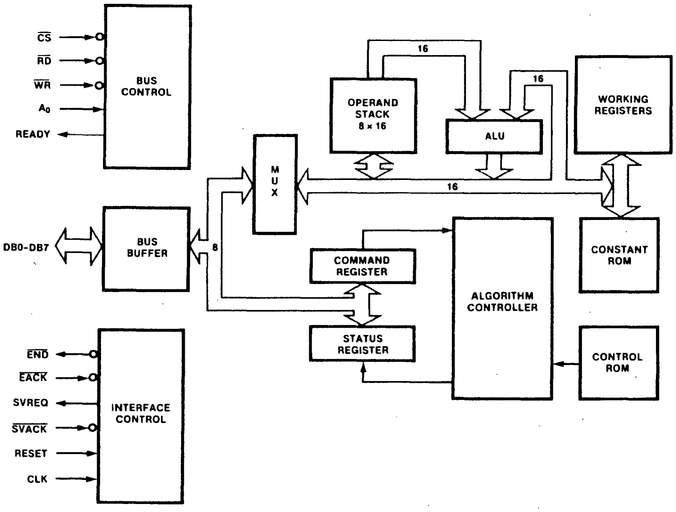
8231A
ARITHMETIC PROCESSING UNIT

- Fixed Point Single and Double Precision (16/32 Bit)
- Floating Point Single Precision (32 Bit)
- Binary Data Formats
- Add, Subtract, Multiply and Divide
- Trignometric and Inverse Trigonometric Functions
- Square Roots, Logarithms, Exponentiation
- Float to Fixed and Fixed to Float Conversions
- Stack Oriented Operand Storage
- Compatible with all Intel and most other Microprocessor Families
- Direct Memory Access or Programmed I/O Data Transfers
- End of Execution Signal
- General Purpose 8-Bit Data Bus Interface
- Standard 24 Pin Package
- +12V and +5V Power Supplies
- Advanced N-Channel Silicon Gate HMOS Technology

The Intel® 8231A Arithmetic Processing Unit (APU) is a monolithic HMOS LSI device that provides high performance fixed and floating point arithmetic and floating point trigonometric operations. It may be used to enhance the mathematical capability of a wide variety of processor-oriented systems. Chebyshev polynomials are used in the implementation of the APU algorithms.

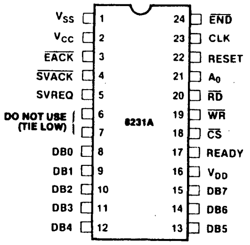
All transfers, including operand, result, status and command information, take place over an 8-bit bidirectional data bus. Operands are pushed onto an internal stack and commands are issued to perform operations on the data and the stack. Results are then available to be retrieved from the stack.

Transfers to and from the APU may be handled by the associated processor using conventional programmed I/O, or may be handled by a direct memory access controller for improved performance. Upon completion of each command, the APU issues an end of execution signal that may be used as an interrupt by the CPU to help coordinate program execution.



231305-1

Figure 1. Block Diagram



231305-2

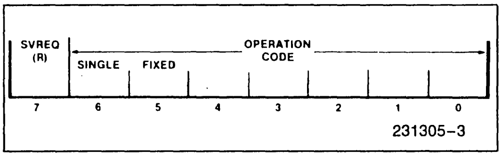
Figure 2. Pin Configuration

Table 1. Pin Description

<table>
<tr><th>Symbol</th><th>Pin No.</th><th>Type</th><th>Name and Function</th></tr>
<tr><td>V_{CC}</td><td>2</td><td></td><td><b>POWER:</b> +5V power supply.</td></tr>
<tr><td>V_{DD}</td><td>16</td><td></td><td><b>POWER:</b> +12V power supply.</td></tr>
<tr><td>V_{SS}</td><td>1</td><td></td><td><b>GROUND.</b></td></tr>
<tr><td>CLK</td><td>23</td><td>I</td><td><b>CLOCK:</b> An external, TTL compatible, timing source is applied to the CLK pin.</td></tr>
<tr><td>RESET</td><td>22</td><td>I</td><td><b>RESET:</b> The active high reset signal provides initialization for the chip. RESET also terminates any operation in progress. RESET clears the status register and places the 8231A into the idle state. Stack contents and command registers are not affected (5 clock cycles).</td></tr>
<tr><td>$\overline{\text{CS}}$</td><td>18</td><td>I</td><td><b>CHIP SELECT:</b> $\overline{\text{CS}}$ is an active low input signal which selects the 8231A and enables communication with the data bus.</td></tr>
<tr><td>A_0</td><td>21</td><td>I</td><td><b>ADDRESS:</b> In conjunction with the $\overline{\text{RD}}$ and WR signals, the A_0 control line establishes the type of communication that is to be performed with the 8231A as shown below:</td></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>A_0</th><th>$\overline{\text{RD}}$</th><th>$\overline{\text{WR}}$</th><th>Function</th></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Enter data byte into stack</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Read data byte from stack</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Enter command</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Read status</td></tr>
</table>
</td></tr>
<tr><td>$\overline{\text{RD}}$</td><td>20</td><td>I</td><td><b>READ:</b> This active low input indicates that data or status is to be read from the 8231A if CS is low.</td></tr>
<tr><td>$\overline{\text{WR}}$</td><td>19</td><td>I</td><td><b>WRITE:</b> This active low input indicates that data or a command is to be written into the 8231A if CS is low.</td></tr>
<tr><td>$\overline{\text{EACK}}$</td><td>3</td><td>I</td><td><b>END OF EXECUTION:</b> This active low input clears the end of execution output signal ($\overline{\text{END}}$. If $\overline{\text{EACK}}$ is tied low, the $\overline{\text{END}}$ output will be a pulse that is one clock period wide.</td></tr>
<tr><td>$\overline{\text{SVACK}}$</td><td>4</td><td>I</td><td><b>SERVICE REQUEST:</b> This active low input clears the service request output (SVREQ).</td></tr>
<tr><td>$\overline{\text{END}}$</td><td>24</td><td>O</td><td><b>END:</b> This active low, open-drain output indicates that execution of the previously entered command is complete. It can be used as an interrupt request and is cleared by $\overline{\text{EACK}}$, RESET or any read or write access to the 8231.</td></tr>
<tr><td>SVREQ</td><td>5</td><td>O</td><td><b>SERVICE REQUEST:</b> This active high output signal indicates that command execution is complete and that post execution service was requested in the previous command byte. It is cleared by $\overline{\text{SVACK}}$, the next command output to the device, or by RESET.</td></tr>
<tr><td>READY</td><td>17</td><td>O</td><td><b>READY:</b> This active high output indicates that the 8231A is able to accept communication with the data bus. When an attempt is made to read data, write data or to enter a new command while the 8231A is executing a command, READY goes low until execution of the current command is complete (See READY Operation, p. 6).</td></tr>
<tr><td>DB0–DB7</td><td>8-15</td><td>I/O</td><td><b>DATA BUS:</b> These eight bidirectional lines provide for transfer of commands, status and data between the 8231A and the CPU. The 8231A can drive the data bus only when CS and RD are low.</td></tr>
</table>

COMMAND STRUCTURE

Each command entered into the 8231A consists of a single 8-bit byte having the format illustrated below:



231305-3

Bits 0–4 select the operation to be performed as shown in the table. Bits 5–6 select the data format appropriate to the selected operation. If bit 5 is a 1, a fixed point data format is specified. If bit 5 is 0, floating point format is specified. Bit 6 selects the precision of the data to be operated upon by fixed point commands only (if bit 5 = 0, bit 6 must be 0). If bit 6 is a 1, single-precision (16-bit) operands are assumed. If bit 6 is a 0, double-precision (32-bit) operands are indicated. Results are undefined for all illegal combinations of bits in the command byte. Bit 7 indicates whether a service request is to be issued after the command is executed. If bit 7 is a 1, the service request output (SVREQ) will go high at the conclusion of the command and will remain high until reset by a low level on the service acknowledge pin ($\overline{\text{SVACK}}$) or until completion of execution of the succeeding command where service request (bit 7) is 0. Each command issued to the 8231A requests post execution service based upon the state of bit 7 in the command byte. When bit 7 is a 0, SVREQ remains low.

Table 2. 32-Bit Floating Point Instructions

Instruction	Description	Hex[1] Code	Stack Contents[2] After Execution A B C D	Status Flags[4] Affected
ACOS	Inverse Cosine of A	0 6	R U U U	S, Z, E
ASIN	Inverse Sine of A	0 5	R U U U	S, Z, E
ATAN	Inverse Tangent of A	0 7	R B U U	S, Z
CHSF	Sign Change of A	1 5	R B C D	S, Z
COS	Cosine of A (radians)	0 3	R B U U	S, Z
EXP	e^A Function	0 A	R B U U	S, Z, E
FADD	Add A and B	1 0	R C D U	S, Z, E
FDIV	Divide B by A	1 3	R C D U	S, Z, E
FLTD	32-Bit Integer to Floating Point Conversion	1 C	R B C U	S, Z
FLTS	16-Bit Integer to Floating Point Conversion	1 D	R B C U	S, Z
FMUL	Multiply A and B	1 2	R C D U	S, Z, E
FSUB	Subtract A from B	1 1	R C D U	S, Z, E
LOG	Common Logarithm (base 10) of A	0 8	R B U U	S, Z, E
LN	Natural Logarithm of A	0 9	R B U U	S, Z, E
POPF	Stack Pop	1 8	B C D A	S, Z
PTOF	Stack Push	1 7	A A B C	S, Z
PUPI	Push π onto Stack	1 A	R A B C	S, Z
PWR	B^A Power Function	0 B	R C U U	S, Z, E
SIN	Sine of A (radians)	0 2	R B U U	S, Z
SQRT	Square Root of A	0 1	R B C U	S, Z, E
TAN	Tangent of A (radians)	0 4	R B U U	S, Z, E
XCHF	Exchange A and B	1 9	B A C D	S, Z

Table 3. 32-Bit Integer Instructions

Instruction	Description	Hex[1] Code	Stack Contents[2] After Execution A B C D	Status Flags[4] Affected
CHSD	Sign Change of A	3 4	R B C D	S, Z, O
DADD	Add A and B	2 C	R C D A	S, Z, C, E
DDIV	Divide B by A	2 F	R C D U	S, Z, E
DMUL	Multiply A and B (R = lower 32-bits)	2 E	R C D U	S, Z, O
DMUU	Multiply A and B (R = upper 32-bits)	3 6	R C D U	S, Z, O
DSUB	Subtract A from B	2 D	R C D A	S, Z, C, O
FIXD	Floating Point to Integer Conversion	1 E	R B C U	S, Z, O
POPD	Stack Pop	3 8	B C D A	S, Z
PTOD	Stack Push	3 7	A A B C	S, Z
XCHD	Exchange A and B	3 9	B A C D	S, Z

Table 4. 16-Bit Integer Instructions

Instruction	Description	Hex[1] Code	Stack Contents[3] After Execution								Status Flags[4] Affected
			A_U	A_L	B_U	B_L	C_U	C_L	D_U	D_L	
CHSS	Change Sign of A_U	7 4	R	A_L	B_U	B_L	C_U	C_L	D_U	D_L	S, Z, O
FIXS	Floating Point to Integer Conversion	1 F	R	B_U	B_L	C_U	C_L	U	U	U	S, Z, O
POPS	Stack Pop	7 8	A_L	B_U	B_L	C_U	C_L	D_U	D_L	A_U	S, Z
PTOS	Stack Push	7 7	A_U	A_U	A_L	B_U	B_L	C_U	C_L	D_U	S, Z
SADD	Add A_U and A_L	6 C	R	B_U	B_L	C_U	C_L	D_U	D_L	A_U	S, Z, C, E
SDIV	Divide A_L by A_U	6 F	R	B_U	B_L	C_U	C_L	D_U	D_L	U	S, Z, E
SMUL	Multiply A_L by A_U (R = lower 16-bits)	6 E	R	B_U	B_L	C_U	C_L	D_U	D_L	U	S, Z, E
SMUU	Multiply A_L by A_U (R = upper 16-bits)	7 6	R	B_U	B_L	C_U	C_L	D_U	D_L	U	S, Z, E
SSUB	Subtract A_U from A_L	6 D	R	B_U	B_L	C_U	C_L	D_U	D_L	A_U	S, Z, C, E
XCHS	Exchange A_U and A_L	7 9	A_L	A_L	B_U	B_L	C_U	C_L	D_U	D_L	S, Z
NOP	No Operation	0 0	A_U	A_L	B_U	B_L	C_U	C_L	D_U	D_L	

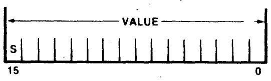
NOTES:

1. In the hex code column, SVREQ is a 0.
2. The stack initially is composed of four 32-bit numbers (A, B, C, D). A is equivalent to Top Of Stack (TOS) and B is Next On Stack (NOS). Upon completion of a command the stack is composed of: the result (R); undefined (U); or the initial contents (A, B, C, or D).
3. The stack initially is composed of eight 16-bit numbers (A_U, A_L, B_U, B_L, C_U, C_L, D_U, D_L). A_U is the TOS and A_L is NOS. Upon completion of a command the stack is composed of: the result (R); undefined (U); or the initial contents (A_U, A_L, B_U, B_L . . .).
4. Nomenclature: Sign (S); Zero (Z); Overflow (O); Carry (C); Error Code Field (E).

DATA FORMATS

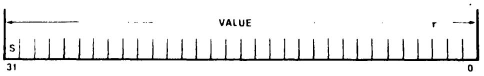
The 8231A arithmetic processing unit handles operands in both fixed point and floating point formats. Fixed point operands may be represented in either single (16-bit operands) or double precision (32-bit operands), and are always represented as binary, two's complement values.

Single Precision Fixed Point Format



231305-4

Double Precision Fixed Point Format



231305-5

The sign (positive or negative) of the operand is located in the most significant bit (MSB). Positive values are represented by a sign bit of zero (S = 0). Negative values are represented by the two's complement of the corresponding positive value with a sign bit equal to 1 (S = 1). The range of values that may be accommodated by each of these formats is −32,768 to +32,767 for single precision and −2,147,483,648 to +2,147,483,647 for double precision.

Floating point binary values are represented in a format that permits arithmetic to be performed in a fashion analogous to operations with decimal values expressed in scientific notation.

$$(5.83 \times 10^2)(8.16 \times 10^1) = (4.75728 \times 10^4)$$

In the decimal system, data may be expressed as values between 0 and 10 times 10 raised to a power that effectively shifts the implied decimal point right or left the number of places necessary to express the result in conventional form (e.g., 47,572.8). The value-portion of the data is called the mantissa. The exponent may be either negative or positive.

The concept of floating point notation has both a gain and a loss associated with it. The gain is the ability to represent the significant digits of data with values spanning a large dynamic range limited only by the capacity of the exponent field. For example, in decimal notation in the exponent field is two digits wide, and the mantissa is five digits, a range of values (positive or negative) from 1.0000×10^{-99} to $9.9999 \times 10^{+99}$ can be accommodated. The loss is that only the significant digits of the value can be represented. Thus there is no distinction in this representation between the values 123451 and 123452, for example since each would be expressed as: 1.2345×10^5. The sixth digit has been discarded. In most applications where the dynamic range of values to be represented in large, the loss of significance, and hence accuracy of results, is a minor consideration. For greater precision a fixed point format could be chosen, although with a loss of potential dynamic range.

The 8231A is a binary arithmetic processor and requires that floating point data be represented by a fractional mantissa value between 0.5 and 1 multiplied by 2 raised to an appropriate power. This is expressed as follows:

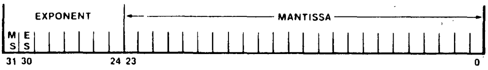
$$\text{value} = \text{mantissa} \times 2^{\text{exponent}}$$

For example, the value 100.5 expressed in this form is $0.1100\ 1001 \times 2^7$. The decimal equivalent of this value may be computed by summing the components (powers of two) of the mantissa and then multiplying by the exponent as shown below:

$$\begin{aligned} \text{value} &= (2^{-1} + 2^{-2} + 2^{-5} + 2^{-8}) \times 2^7 \\ &= 0.5 + 0.25 + 0.03125 + 0.00390625) \times 128 \\ &= 0.78515625 \times 128 \\ &= 100.5 \end{aligned}$$

FLOATING POINT FORMAT

The format for floating point values in the 8231A is given below. The mantissa is expressed as a 24-bit (fractional) value; the exponent is expressed as a two's complement 7-bit value having a range of −64 to +63. The most significant bit is the sign of the mantissa (0 = positive, 1 = negative), for a total of 32 bits. The binary point is assumed to be the left of the most significant mantissa bit (bit 23). All floating point data values must be normalized. Bit 23 must be equal to 1, except for the value zero, which is represented by all zeros.



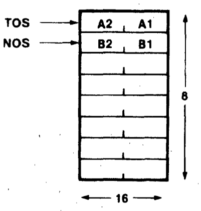
231305-6

The range of values that can be represented in this format is $\pm(2.7 \times 10^{-20}$ to $9.2 \times 10^{18})$ and zero.

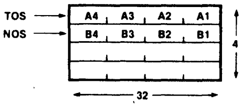
FUNCTIONAL DESCRIPTION

STACK CONTROL

The user interface to the 8231A includes access to an 8 level 16-bit wide data stack. Since single precision fixed point operands are 16-bits in length, eight such values may be maintained in the stack. When using double precision fixed point or floating point formats four values may be stored. The stack in these two configurations can be visualized as shown below:



231305-7



231305-8

Data are written onto the stack, eight bits at a time, in the order shown (A1, A2, A3, . . .). Data are removed from the stack in reverse byte order (A4, A3, A2 . . .). Data should be entered onto the stack in multiples of the number of bytes appropriate to the chosen data format.

DATA ENTRY

Data entry is accomplished by bringing the chip select ($\overline{\text{CS}}$), the command/data line (A_0), and $\overline{\text{WR}}$ low, as shown in the timing diagram. The entry of each new data word "pushes down" the previously entered data and places the new byte on the top of stack (TOS). Data on the bottom of the stack prior to a stack entry are lost.

DATA REMOVAL

Data are removed from the stack in the 8231A by bringing chip select ($\overline{\text{CS}}$), command/data (A_0), and $\overline{\text{RD}}$ low as shown in the timing diagram. The removal of each data world redefines TOS so that the next successive byte to be removed becomes TOS. Data removed from the stack rotates to the bottom of the stack.

COMMAND ENTRY

After the appropriate number of bytes of data have been entered onto the stack, a command may be issued to perform an operation on that data. Commands which require two operands for execution (e.g., add) operate on the TOS and NOS values. Single operand commands operate only on the TOS.

Commands are issued to the 8231A by bringing the chip select ($\overline{\text{CS}}$) line low, command data (A_0) line high, and $\overline{\text{WR}}$ line low as indicated by the timing diagram. After a command is issued, the CPU can continue execution of its program concurrently with the 8231A command execution.

COMMAND COMPLETION

The 8231A signals the completion of each command execution by lowering the End Execution line ($\overline{\text{END}}$). Simultaneously, the busy bit in the status register is cleared and the Service Request bit of the command register is checked. If it is a "1" the service request output level (SVREQ) is raised. $\overline{\text{END}}$ is cleared on receipt of an active low End Acknowledge ($\overline{\text{EACK}}$) pulse. Similarly, the service request line is cleared by recognition of an active low Service Acknowledge ($\overline{\text{SVACK}}$) pulse.

READY OPERATION

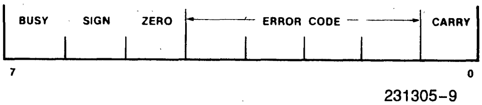
An active high ready (READY) is provided. This line is high in its quiescent state and is pulled low by the 8231A under the following conditions:

1. A previouly initiated operation is in progress (device busy) and Command Entry has been attempted. In this case, the READY line will be pulled low and remain low until completion of the current command execution. It will then go high, permitting entry of the new command.
2. A previously initiated operation is in progress and stack access has been attempted. In this case, the READY line will be pulled low, will remain in that state until execution is complete, and will then be raised to permit completion of the stack access.
3. The 8231A is not busy, and data removal has been requested. READY will be pulled low for the length of time necessary to transfer the byte from the top of stack to the interface latch, and will then go high, indicating availability of the data.
4. The 8231A is not busy, and a data entry has been requested. READY will be pulled low for the length of time required to ascertain if the preceding data byte, if any, has been written to the stack. If so READY will immediately go high. If not, READY will remain low until the interface latch is free and will then go high.
5. When a status read has been requested, READY will be pulled low for the length of time necessary to transfer the status to the interface latch, and will then be raised to permit completion of the status read. Status may be read whether or not the 8231A is busy.

When READY goes low, the APU expects the bus control signals present at the time to remain stable until READY goes high.

DEVICE STATUS

Device status is provided by means of an internal status register whose format is shown below:



231305-9

Busy: Indicates that 8231A is currently executing a command (1 = Busy)

Sign: Indicates that the value on the top of stack is negative (1 = Negative)

Zero: Indicates that the value on the top of stack is zero (1 = Value is zero)

Error Code: This field contains an indication of the validity of the result of the last operation. The error codes are:

0000—No error

1000—Divide by zero

0100—Square root or log of negative number

1100—Argument of inverse sine, cosine, or e^x too large

XX10—Underflow

XX01—Overflow

Carry: Previous operation resulted in carry or borrow from most significant bit. (1 = Carry/Borrow, 0 = No Carry/No Borrow).

If the BUSY bit in the status register is a one, the other status bits are not defined; if zero, indicating not busy the operation is complete and the other status bits are defined as given above.

READ STATUS

The 8231A status register can be read by the CPU at any time (whether an operation is in progress or not) by bringing the chip select ($\overline{CS}$) low, the command/data line (A_0) high, and lowering $\overline{RD}$. The status register is then gated onto the data bus and may be input by the CPU.

EXECUTION TIMES

Timing for execution of the 8231A command set is contained below. All times are given in terms of clock cycles. Where substantial variation of execution times is possible, the minimum and maximum values are quoted; otherwise, typical values are given. Variations are data dependent.

Total execution times may require allowances for operand transfer into the APU, command execution, and result retrieval from the APU. Except for command execution, these times will be heavily influenced by the nature of the data, the control interface used, the speed of memory, the CPU used, the priority allotted to DMA and interrupt operations, the size and number of operands to be transferred, and the use of chained calculations, etc.

DERIVED FUNCTION DISCUSSION

Computer approximations of transcendental functions are often based on some form of polynomial equation, such as:

$$F(X) = A_0 + A_1X + A_2X^2 + A_3X^3 + A_4X^4 \ldots \quad (1\text{-}1)$$

Table 5. Command Execution Times

Command Mnemonic	Clock Cycles	Command Mnemonic	Clock Cycles	Command Mnemonic	Clock Cycles	Command Mnemonic	Clock Cycles
SADD	17	FADD	54-368	LN	4298-6956	POPF	12
SSUB	30	FSUB	70-370	EXP	3794-4878	XCHS	18
SMUL	84-94	FMUL	146-168	PWR	8290-12032	XCHD	26
SMUU	80-98						
SDIV	84-94	FDIV	154-184	NOP	4	XCHF	26
DADD	21	SORT	800	CHSS	23	PUPI	16
DSUB	38	SIN	4464	CHSD	27		
DMUL	194-210	COS	4118	CHSF	18		
DMUU	182-218						
DDIV	208	TAN	5754	PTOS	16		
FIXS	92-216	ASIN	7668	PTOD	20		
FIXD	100-346	ACOS	7734	PTOF	20		
FLTS	98-186	ATAN	6006	POPS	10		
FLTD	98-378	LOG	4474-7132	POPD	12		

The primary shortcoming of an approximation is this form is that it typically exhibits very large errors when the magnitude of $|X|$ is large, although the errors are small when $|X|$ is small. With polynomials in this form, the error distribution is markedly uneven over any arbitrary interval.

A set of approximating functions exists that not only minimizes the maximum error but also provides an even distribution of errors within the selected data representation interval. These are known as Chebyshev Polynomials and are based upon cosine functions. These functions are defined as follows:

$$T_n(X) = \text{Cos}\, n\theta; \text{ where } n = 0, 1, 2 \ldots \qquad (1\text{-}2)$$

$$\theta = \text{Cos}^{-1} X$$

The various terms of the Chebyshev series can be computed as shown below:

$$T_0(X) = \text{Cos}\,(0 \times \theta) = \text{Cos}\,(0) = 1 \qquad (1\text{-}4)$$

$$T_1(X) = \text{Cos}\,(\text{Cos}^{-1}X) = X \qquad (1\text{-}5)$$

$$T_2(X) = \text{Cos}\,2\theta = 2\text{Cos}^2\theta - 1 = 2\text{Cos}^2\,(\text{Cos}^{-1}X) - 1 \qquad (1\text{-}6)$$

$$= 2X^2 - 1$$

In general, the next term in the Chebyshev series can be recursively derived from the previous term as follows:

$$T_n(X) = 2X\,[T_n - 1(X)] - T_n - 2(X); n \geq 2 \qquad (1\text{-}7)$$

Common logarithms are computed by multiplication of the natural logarithm by the conversion factor 0.43429448 and the error function is therefore the same as that for natural logarithm. The power function is realized by combination of natural log and exponential functions according to the equation:

$$X^Y = e^{yLnx}.$$

The error for the power function is a combination of that for the logarithm and exponential functions.

Each of the derived functions is an approximation of the true function. Thus the result of a derived function will have an error. The absolute error is the difference between the function's result and the true result. A more useful measure of the function's error is relative error (absolute error/true result). This gives a measurement of the significant digits of algorithm accuracy. For the derived functions except LN, LOG, and PWR the relative error is typically 4×10^{-7}. For PWR the relative error is the summation of the EXP and LN errors, 7×10^{-7}. For LN and LOG, the absolute error is 2×10^{-7}.

APPLICATION INFORMATION

The diagram in Figure 4 shows the interface connections for the APU with operand transfers handled by an 8237 DMA controller, and CPU coordination handled by an Interrupt Controller. The APU interrupts the CPU to indicate that a command has been completed. When the performance enhancements provided by the DMA and Interrupt operations are not required, the APU interface can be simplified as shown in Figure 3. The 8231A APU is designed with a general purpose 8-bit data bus and interface control so that it can be conveniently used with any general 8-bit processor.

In many systems it will be convenient to use the microcomputer system clock to drive the APU clock input. In the case of 8080A systems it would be the ϕ2TTL signal. Its cycle time will usually fall in the range of 250 ns to 1000 ns, depending on the system speed.

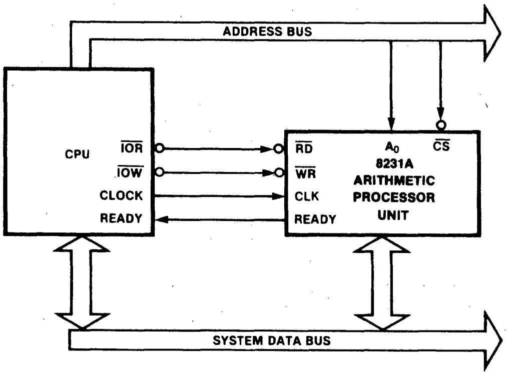


Figure 3. Minimum Configuration Example

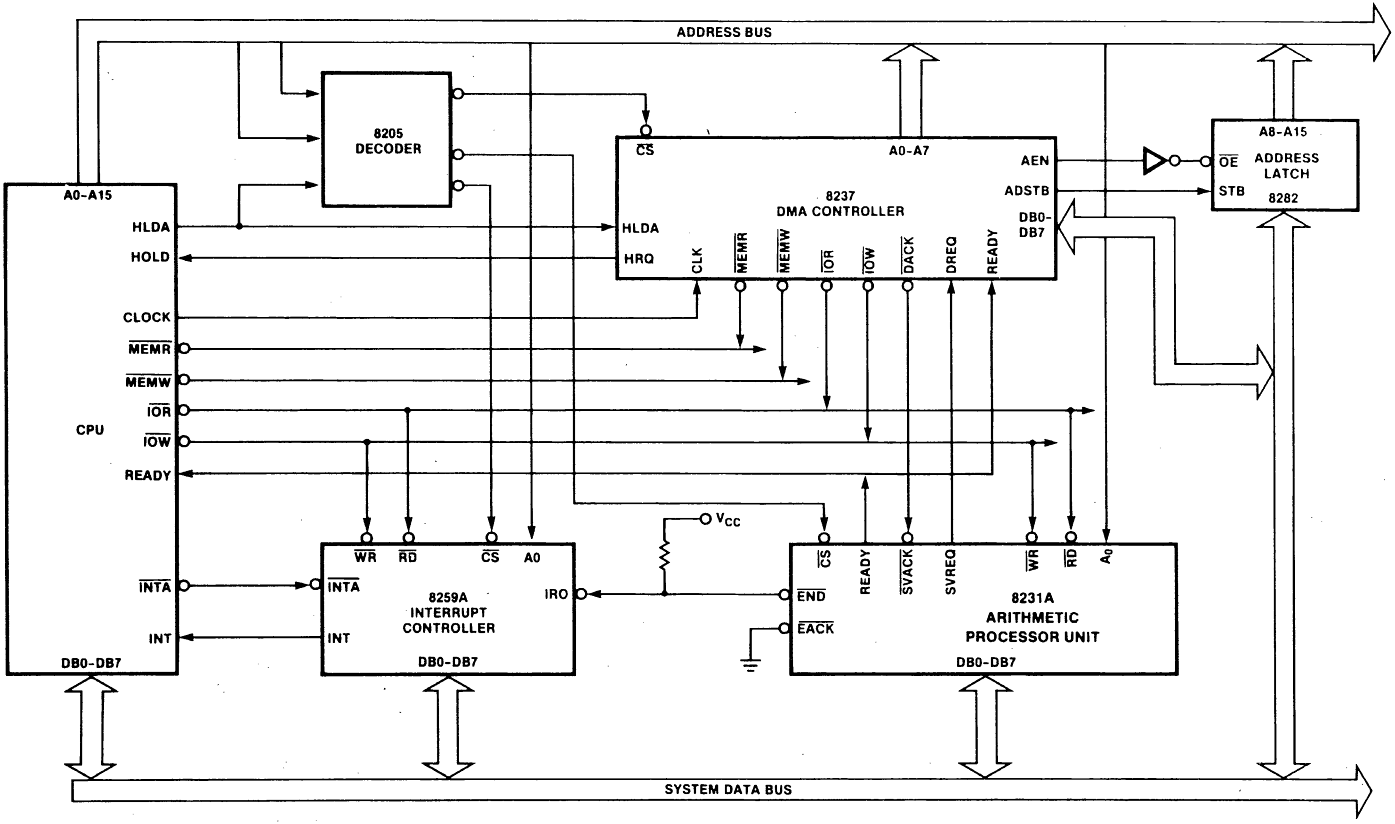


Figure 4. High Performance Configuration Example

ABSOLUTE MAXIMUM RATINGS*

Storage Temperature .... −65°C to +150°C

Ambient Temperature Under Bias ...... 0°C to 70°C

V_{DD} with Respect to V_{SS} .... −0.5V to +15.0V

V_{CC} with Respect to V_{SS} .... −0.5V to +7.0V

All Signal Voltages with Respect to V_{SS} .... −0.5V to +7.0V

Power Dissipation 2.0W

**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.*

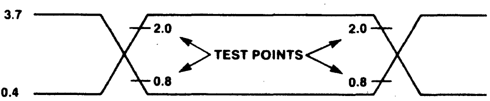
D.C. AND OPERATING CHARACTERISTICS

T_A = 0°C to 70°C, V_{SS} = 0V, V_{CC} = +5V ±10%, V_{DD} = +12V ±10%

Parameters	Description	Min	Typ	Max	Units	Test Conditions
V_{OH}	Output HIGH Voltage	3.7			V	$I_{OH} = -200\ \mu A$
V_{OL}	Output LOW Voltage			0.4	V	$I_{OL} = 3.2$ mA
V_{IH}	Input HIGH Voltage	2.0		V_{CC}	V	
V_{IL}	Input LOW Voltage	−0.5		0.8	V	
I_{IL}	Input Load Current			±10	μA	$V_{SS} \le V_{IN} \le V_{CC}$
I_{OFL}	Data Bus Leakage			±10	μA	$V_{SS} + 0.45 \le V_{OUT} \le V_{CC}$
I_{CC}	V_{CC} Supply Current		50	95	mA	
I_{DD}	V_{DD} Supply Current		50	95	mA	
C_O	Output Capacitance		8		pF	fc = 1.0 MHz, Inputs = 0V(1)
C_I	Input Capacitance		5		pF	
C_{IO}	I/O Capacitance		10		pF	

NOTE:
1. Sampled, not 100% tested.

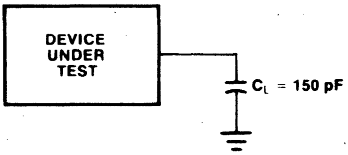
A.C. TESTING INPUT, OUTPUT WAVEFORM



231305–12

A.C. Testing: Inputs are driven at 3.7V for a logical "1" and 0.4V 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



231305–13

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{SS} = 0V, V_{CC} = +5V ±10%, V_{DD} = +12V ±10%

READ OPERATION

Symbol	Parameter		8231A-8		8231A		Units
			Min	Max	Min	Max	
t_{AR}	A_0, $\overline{CS}$ Setup to $\overline{RD}$		0		0		ns
t_{RA}	A_0, $\overline{CS}$ Hold from $\overline{RD}$		0		0		ns
t_{RY}	READY ↓ from $\overline{RD}$ ↓ Delay (Note 2)			150		100	ns
t_{YR}	Ready ↑ to $\overline{RD}$ ↑		0		0		ns
t_{RRR}	READY Pulse Width (Note 3)	Data	3.5 t_{CY} + 50		3.5 t_{CY} + 50		ns
		Status	1.5 t_{CY} + 50		1.5 t_{CY} + 50		ns
t_{RDE}	Data Bus Enable from $\overline{RD}$ ↓		50		50		ns
t_{DRY}	Data Valid to READY ↑		0		0		ns
t_{DF}	Data Float after $\overline{RD}$ ↑		50	200	50	100	ns

WRITE OPERATION

Symbol	Parameter		8231A-8		8231A		Units
			Min.	Max.	Min.	Max.	
t_{AW}	A_0, $\overline{CS}$ Setup to $\overline{WR}$		0		0		ns
t_{WA}	A_0, $\overline{CS}$ Hold after $\overline{WR}$		60		25		ns
t_{WY}	READY ↓ from $\overline{WR}$ ↓ Delay (Note 2)			150		100	ns
t_{YW}	READY ↑ to $\overline{WR}$ ↑		0		0		ns
t_{RRW}	READY Pulse Width (Note 4)			50		50	ns
t_{WI}	Write Inactive Time (Note 4)	Command	4 t_{CY}		4 t_{CY}		ns
		Data	5 t_{CY}		5 t_{CY}		ns
t_{DW}	Data Setup to $\overline{WR}$		150		100		ns
t_{WD}	Data Hold after $\overline{WR}$		20		20		ns

OTHER TIMINGS

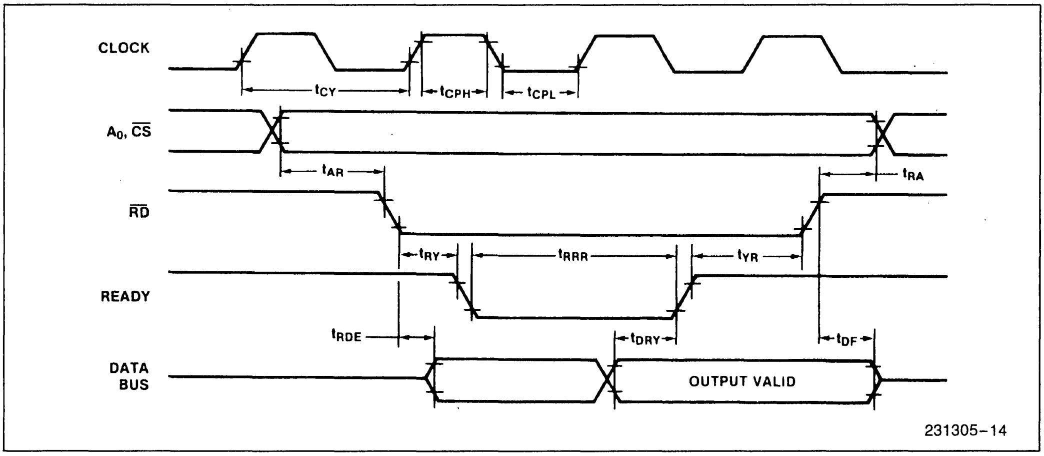
Symbol	Parameter	8231A-8		8231A		Units
		Min	Max	Min	Max	
t_{CY}	Clock Period	480	5000	250	2500	ns
t_{CPH}	Clock Pulse High Width	200		100		ns
t_{CPL}	Clock Pulse Low Width	240		120		ns
t_{EE}	$\overline{\text{END}}$ Pulse Width (Note 5)	400		200		ns
t_{EAE}	$\overline{\text{EACK}}$ ↓ to $\overline{\text{END}}$ ↑ Delay		200		150	ns
t_{AA}	$\overline{\text{EACK}}$ Pulse Width	100		50		ns
t_{SA}	$\overline{\text{SVACK}}$ ↓ to SVREQ ↓ Delay		300		150	ns
t_{SS}	$\overline{\text{SVACK}}$ Pulse Width	100		50		ns

NOTES:

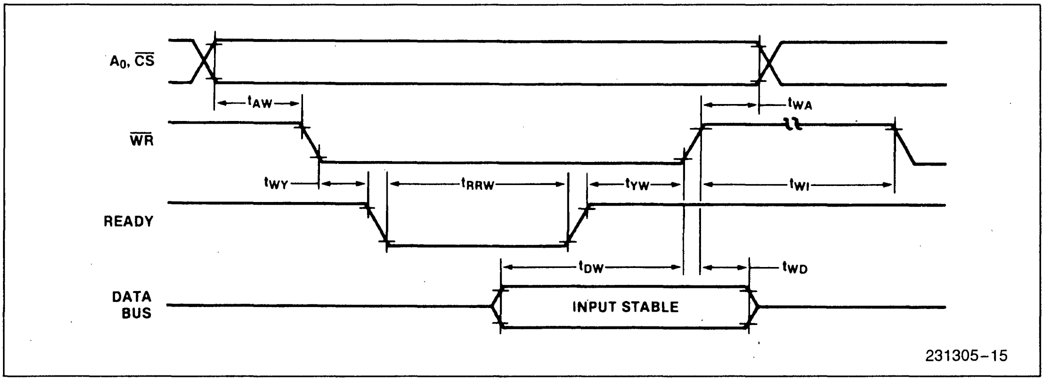
1. Typical values are for $T_A = 25°C$, nominal supply voltages processing parameters.
2. READY is pulled low for both command and data operations.
3. Minimum values shown assume no previously entered command is being executed for the data access. If a previously entered command is being executed, READY low pulse width is the time to complete execution plus the time shown. Status may be read at any time without exceeding the time shown.
4. READY low pulse width is less than 50 ns when writing into the data port or the control port as long as the duty cycle requirement (t_{WI}) is observed and no previous command is being executed. t_{WI} may be safely violated as long as the extended t_{RRW} that results is observed. If a previously entered command is being executed, READY low pulse width is the time to complete execution plus the time shown. These timings refer specifically to the 8231A.
5. $\overline{\text{END}}$ low pulse width is specified for $\overline{\text{EACK}}$ tied to VSS. Otherwise t_{EAE} applies.

WAVEFORMS

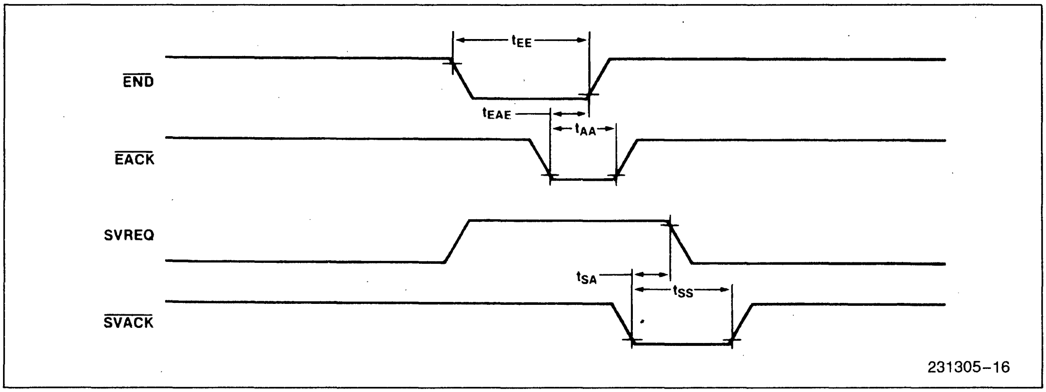
READ OPERATION



WRITE OPERATION



INTERRUPT OPERATION

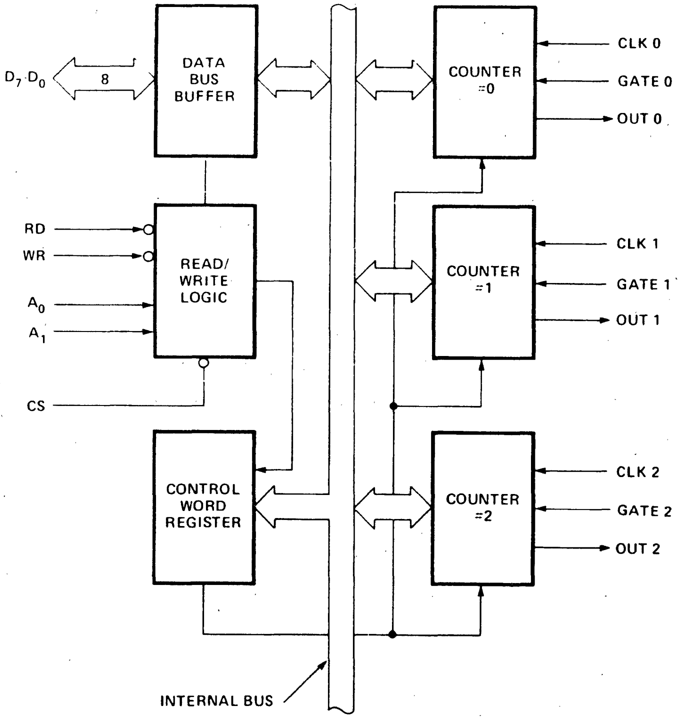


8253/8253-5
PROGRAMMABLE INTERVAL TIMER

- MCS-85™ Compatible 8253-5
- 3 Independent 16-Bit Counters
- DC to 2.6 MHz
- Programmable Counter Modes
- Count Binary or BCD
- Single +5V Supply
- Available in EXPRESS
  — Standard Temperature Range
  — Extended Temperature Range

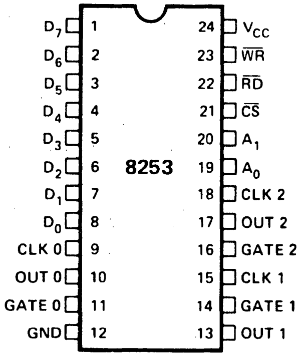
The Intel® 8253 is a programmable counter/timer device designed for use as an Intel microcomputer peripheral. It uses NMOS technology with a single +5V supply and is packaged in a 24-pin plastic DIP.

It is organized as 3 independent 16-bit counters, each with a count rate of up to 2.6 MHz. All modes of operation are software programmable.



231306–1

Figure 1. Block Diagram



231306–2

Figure 2. Pin Configuration

FUNCTIONAL DESCRIPTION

General

The 8253 is programmable interval timer/counter specifically designed for use with the Intel™ Microcomputer systems. Its function is that of a general purpose, multi-timing element that can be treated as an array of I/O ports in the system software.

The 8253 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control. Instead of setting up timing loops in systems software, the programmer configures the 8253 to match his requirements, initializes one of the counters of the 8253 with the desired quantity, then upon command the 8253 will count out the delay and interrupt the CPU when it has completed its tasks. It is easy to see that the software overhead is minimal and that multiple delays can easily be maintained by assignment of priority levels.

Other counter/timer functions that are non-delay in nature but also common to most microcomputers can be implemented with the 8253.

- Programmable Rate Generator
- Event Counter
- Binary Rate Multiplier
- Real Time Clock
- Digital One-Shot
- Complex Motor Controller

Data Bus Buffer

The 3-state, bi-directional, 8-bit buffer is used to interface the 8253 to the system data bus. Data is transmitted or received by the buffer upon execution of INput or OUTput CPU instructions. The Data Bus Buffer has three basic functions.

1. Programming the MODES of the 8253.
2. Loading the count registers.
3. Reading the count values.

Read/Write Logic

The Read/Write Logic accepts inputs from the system bus and in turn generates control signals for overall device operation. It is enabled or disabled by CS so that no operation can occur to change the function unless the device has been selected by the system logic.

$\overline{RD}$ (Read)

A "low" on this input informs the 8253 that the CPU is inputting data in the form of a counters value.

$\overline{WR}$ (Write)

A "low" on this input informs the 8253 that the CPU is outputting data in the form of mode information or loading counters.

A0, A1

These inputs are normally connected to the address bus. Their function is to select one of the three counters to be operated on and to address the control word register for mode selection.

$\overline{CS}$ (Chip Select)

A "low" on this input enables the 8253. No reading or writing will occur unless the device is selected. The $\overline{CS}$ input has no effect upon the actual operation of the counters.

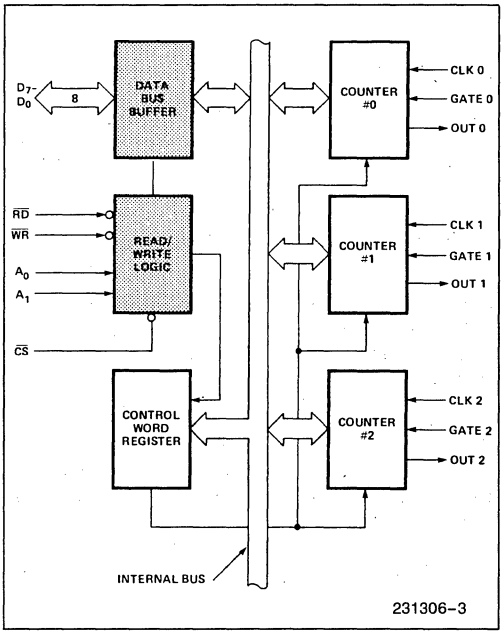


Figure 3. Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

$\overline{\text{CS}}$	$\overline{\text{RD}}$	$\overline{\text{WR}}$	A_1	A_0	
0	1	0	0	0	Load Counter No. 0
0	1	0	0	1	Load Counter No. 1
0	1	0	1	0	Load Counter No. 2
0	1	0	1	1	Write Mode Word
0	0	1	0	0	Read Counter No. 0
0	0	1	0	1	Read Counter No. 1
0	0	1	1	0	Read Counter No. 2
0	0	1	1	1	No-Operation 3-State
1	X	X	X	X	Disable 3-State
0	1	1	X	X	No-Operation 3-State

Control Word Register

The Control Word Register is selected when A0, A1 are 11. It then accepts information from the data bus buffer and stores it in a register. The information stored in this register controls the operation MODE of each counter, selection of binary or BCD counting and the loading of each count register.

The Control Word Register can only be written into; no read operation of its contents is available.

Counter #0, Counter #1, Counter #2

These three functional blocks are identical in operation so only a single counter will be described. Each Counter consists of a single, 16-bit, pre-settable, DOWN counter. The counter can operate in either binary or BCD and its input, gate and output are configured by the selection of MODES stored in the Control Word Register.

The counters are fully independent and each can have separate MODE configuration and counting operation, binary or BCD. Also, there are special features in the control word that handle the loading of the count value so that software overhead can be minimized for these functions.

The reading of the contents of each counter is available to the programmer with simple READ operations for event counting applications and special commands and logic are included in the 8253 so that the contents of each counter can be read "on the fly" without having to inhibit the clock input.

8253 SYSTEM INTERFACE

The 8253 is a component of the Intel™ Microcomputer systems and interfaces in the same manner as all other peripherals of the family. It is treated by the systems software as an array of peripheral I/O ports; three are counters and the fourth is a control register for MODE programming.

Basically, the select inputs A0, A1 connect to the A0, A1 address bus signals of the CPU. The $\overline{\text{CS}}$ can be derived directly from the address bus using a linear select method. Or it can be connected to the output of a decoder, such as an Intel 8205 for larger systems.

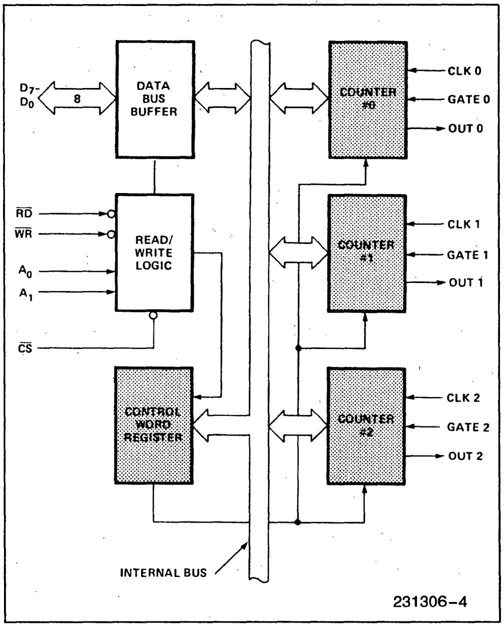


Figure 4. Block Diagram Showing Control Word Register and Counter Functions

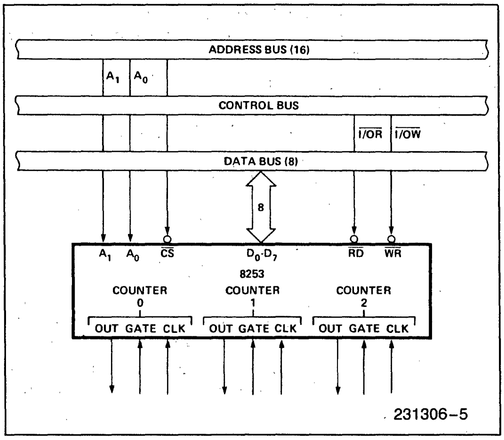


Figure 5. 8253 System Interface

OPERATIONAL DESCRIPTION

General

The complete functional definition of the 8253 is programmed by the systems software. A set of control words *must* be sent out by the CPU to initialize each counter of the 8253 with the desired MODE and quantity information. Prior to initialization, the MODE, count, and output of all counters is undefined. These control words program the MODE, Loading sequence and selection of binary or BCD counting.

Once programmed, the 8253 is ready to perform whatever timing tasks it is assigned to accomplish.

The actual counting operation of each counter is completely independent and additional logic is provided on-chip so that the usual problems associated with efficient monitoring and management of external, asynchronous events or rates to the microcomputer system have been eliminated.

Programming the 8253

All of the MODES for each counter are programmed by the systems software by simple I/O operations.

Each counter of the 8253 is individually programmed by writing a control word into the Control Word Register. (A0, A1 = 11)

Control Word Format

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

Definition Of Control

SC—SELECT COUNTER:

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Illegal

RL—READ/LOAD:

RL1	RL0	
0	0	Counter Latching operation (see READ/WRITE Procedure Section).
1	0	Read/Load most significant byte only.
0	1	Read/Load least significant byte only.
1	1	Read/Load least significant byte first, then most significant byte.

M—MODE:

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

BCD:

0	Binary Counter 16-Bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

Counter Loading

The count register is not loaded until the count value is written (one or two bytes, depending on the mode selected by the RL bits), followed by a rising edge and a falling edge of the clock. Any read of the counter prior to that falling clock edge may yield invalid data.

MODE DEFINITION

MODE 0: Interrupt on Terminal Count. The output will be initially low after the mode set operation. After the count is loaded into the selected count register, the output will remain low and the counter will count. When terminal count is reached, the output will go high and remain high until the selected count register is reloaded with the mode or a new count is loaded. The counter continues to decrement after terminal count has been reached.

Rewriting a counter register during counting results in the following:

(1) Write 1st byte stops the current counting.
(2) Write 2nd byte starts the new count.

MODE 1: Programmable One-Shot. The output will go low on the count following the rising edge of the gate input.

The output will go high on the terminal count. If a new count value is loaded while the output is low it will not affect the duration of the one-shot pulse until the succeeding trigger. The current count can be read at any time without affecting the one-shot pulse.

The one-shot is retriggerable, hence the output will remain low for the full count after any rising edge of the gate input.

MODE 2: Rate Generator. Divide by N counter. The output will be low for one period of the input clock. The period from one output pulse to the next equals the number of input counts in the count register. If the count register is reloaded between output pulses the present period will not be affected, but the subsequent period will reflect the new value.

The gate input, when low, will force the output high. When the gate input goes high, the counter will start from the initial count. Thus, the gate input can be used to synchronize the counter.

When this mode is set, the output will remain high until after the count register is loaded. The output then can also be synchronized by software.

MODE 3: Square Wave Rate Generator. Similar to MODE 2 except that the output will remain high until one half the count has been completed (or even numbers) and go low for the other half of the count. This is accomplished by decrementing the counter by two on the falling edge of each clock pulse. When the counter reaches terminal count, the state of the output is changed and the counter is reloaded with the full count and the whole process is repeated.

If the count is odd and the output is high, the first clock pulse (after the count is loaded) decrements the count by 1. Subsequent clock pulses decrement the clock by 2. After timeout, the output goes low and the full count is reloaded. The first clock pulse (following the reload) decrements the counter by 3. Subsequent clock pulses decrement the count by 2 until timeout. Then the whole process is repeated. In this way, if the count is odd, the output will be high for (N + 1)/2 counts and low for (N − 1)/2 counts.

In Modes 2 and 3, if a CLK source other than the system clock is used, GATE should be pulsed immediately following $\overline{\text{WR}}$ of a new count value.

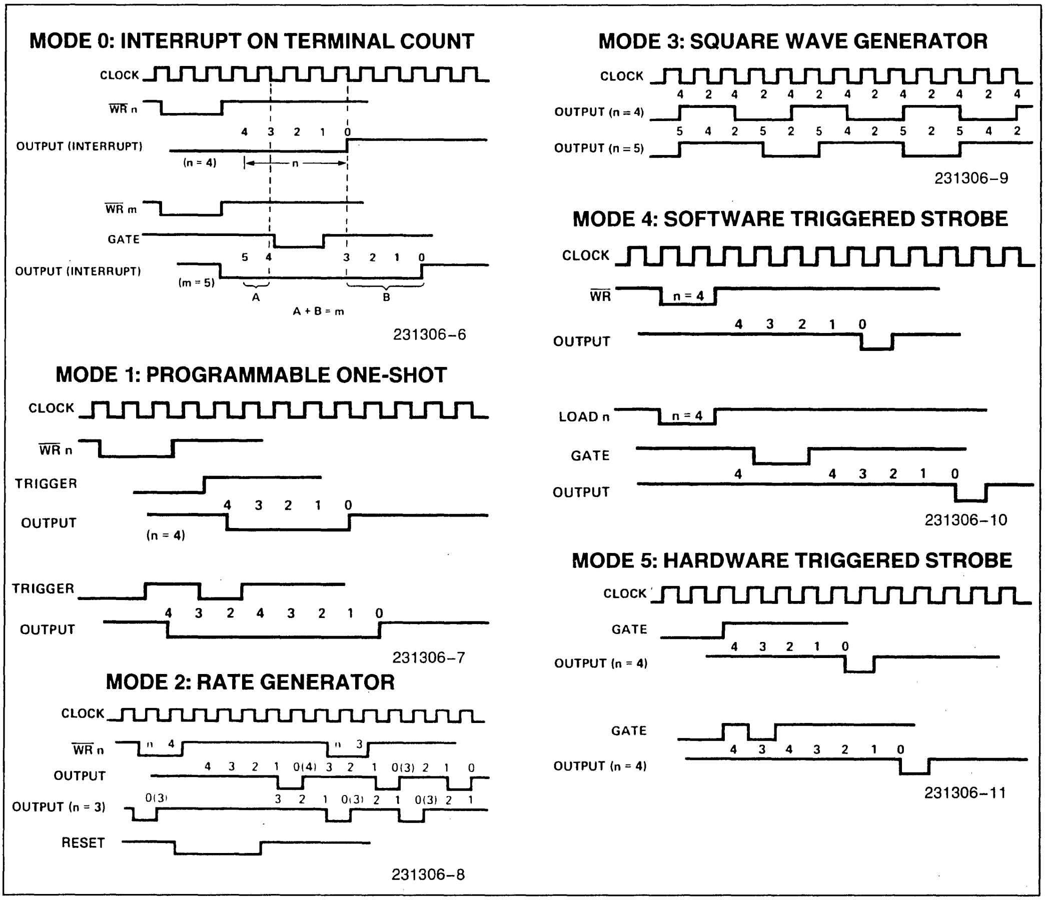
MODE 4: Software Triggered Strobe. After the mode is set, the output will be high. When the count is loaded, the counter will begin counting. On terminal count, the output will go low for one input clock period, then will go high again.

If the count register is reloaded during counting, the new count will be loaded on the next CLK pulse. The count will be inhibited while the GATE input is low.

MODE 5: Hardware Triggered Strobe. The counter will start counting after the rising edge of the trigger input and will go low for one clock period when the terminal count is reached. The counter is retriggerable. The output will not go low until the full count after the rising edge of any trigger.

Signal Status Modes	Low Or Going Low	Rising	High
0	Disables counting	—	Enables counting
1	—	1) Initiates counting 2) Resets output after next clock	—
2	1) Disables counting 2) Sets output immediately high	1) Reloads counter 2) Initiates counting	Enables counting
3	1) Disables counting 2) Sets output immediately high	1) Reloads counter 2) Initiates counting	Enables counting
4	Disables counting	—	Enables counting
5	—	Initiates counting	—

Figure 6. Gate Pin Operations Summary



MODE 0: INTERRUPT ON TERMINAL COUNT

231306-6

MODE 1: PROGRAMMABLE ONE-SHOT

231306-7

MODE 2: RATE GENERATOR

231306-8

MODE 3: SQUARE WAVE GENERATOR

231306-9

MODE 4: SOFTWARE TRIGGERED STROBE

231306-10

MODE 5: HARDWARE TRIGGERED STROBE

231306-11

Figure 7. 8253 Timing Diagrams

8253 READ/WRITE PROCEDURE

Write Operations

The systems software must program each counter of the 8253 with the mode and quantity desired. The programmer must write out to the 8253 a MODE control word and the programmed number of count register bytes (1 or 2) prior to actually using the selected counter.

The actual order of the programming is quite flexible. Writing out of the MODE control word can be in any sequence of counter selection, e.g., counter #0 does not have to be first or counter #2 last. Each counter's MODE control word register has a separate address so that its loading is completely sequence independent. (SC0, SC1).

The loading of the Count Register with the actual count value, however, must be done in exactly the sequence programmed in the MODE control word (RL0, RL1). This loading of the counter's count register is still sequence independent like the MODE control word loading, but when a selected count register is to be loaded it *must* be loaded with the number of bytes programmed in the MODE control word (RL0, RL1). The one or two bytes to be loaded in the count register do not have to follow the associated MODE control word. They can be programmed at any time following the MODE control word loading as long as the correct number of bytes is loaded in order.

All counters are down counters. Thus, the value loaded into the count register will actually be decremented. Loading all zeros into a count register will result in the maximum count (2^{16} for Binary or 10^4 for BCD). In MODE 0 the new count will not restart until the load has been completed. It will accept one of two bytes depending on how the MODE control words (RL0, RL1) are programmed. Then proceed with the restart operation.

	MODE Control Word Counter n
LSB	Counter Register byte Counter n
MSB	Counter Register byte Counter n

NOTE:
Format shown is a simple example of loading the 8253 and does not imply that it is the only format that can be used.

Figure 8. Programming Format

			A1	**A0**
No. 1		MODE Control Word Counter 0	1	1
No. 2		MODE Control Word Counter 1	1	1
No. 3		MODE Control Word Counter 2	1	1
No. 4	LSB	Count Register Byte Counter 1	0	1
No. 5	MSB	Count Register Byte Counter 1	0	1
No. 6	LSB	Count Register Byte Counter 2	1	0
No. 7	MSB	Count Register Byte Counter 2	1	0
No. 8	LSB	Count Register Byte Counter 0	0	0
No. 9	MSB	Count Register Byte Counter 0	0	0

NOTE:
The exclusive addresses of each counter's count register make the task of programming the 8253 a very simple matter, and maximum effective use of the device will result if this feature is fully initilized.

Figure 9. Alternate Programming Formats

Read Operations

In most counter applications it becomes necessary to read the value of the count in progress and make a computational decision based on this quantity. Event counters are probably the most common application that uses this function. The 8253 contains logic that will allow the programmer to easily read the contents of any of the three counters without disturbing the actual count in progress.

There are two methods that the programmer can use to read the value of the counters. The first method involves the use of simple I/O read operations of the selected counter. By controlling the A0, A1 inputs to the 8253 the programmer can select the counter to be read (remember that no read operation of the mode register is allowed A0, A1-11). The only requirement with this method is that in order to assure a stable count reading the actual operation of the selected counter *must be inhibited* either by controlling the Gate input or by external logic that inhibits the clock input. The contents of the counter selected will be available as follows:

First I/O Read contains the least significant byte (LSB).

Second I/O Read contains the most significant byte (MSB).

Due to the internal logic of the 8253 it is absolutely necessary to complete the entire reading procedure. If two bytes are programmed to be read, then two bytes *must* be read before any loading WR command can be sent to the same counter.

Read Operation Chart

A1	A0	RD	
0	0	0	Read Counter No. 0
0	1	0	Read Counter No. 1
1	0	0	Read Counter No. 2
1	1	0	Illegal

Reading While Counting

In order for the programmer to read the contents of any counter without effecting or disturbing the counting operation the 8253 has special internal logic that can be accessed using simple $\overline{WR}$ commands to the MODE register. Basically, when the programmer wishes to read the contents of a selected counter "on the fly" he loads the MODE register with a special code which latches the present count value into a storage register so that its contents contain an accurate, stable quantity. The programmer then issues a normal read command to the selected counter and the contents of the latched register is available.

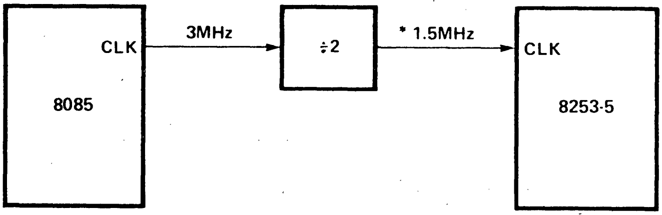
MODE Register for Latching Count

A0, A1 = 11

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	0	0	X	X	X	X

SC1, SC0— specify counter to be latched.
D5, D4 — 00 designates counter latching operation.
X — don't care.

The same limitation applies to this mode of reading the counter as the previous method. That is, it is mandatory to complete the entire read operation as programmed. This command has no effect on the counter's mode.



*If an 8085 clock output is to drive an 8253-5 clock input, it must be reduced to 2 MHz or less.

Figure 10. MCS-85™ Clock Interface*

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	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_A = 0°C to 70°C, V_{CC} = 5V ±10%*

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2.2	V_{CC} + .5V	V	
V_{OL}	Output Low Voltage		0.45	V	(Note 1)
V_{OH}	Output High Voltage	2.4		V	(Note 2)
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to 0.45V
I_{CC}	V_{CC} Supply Current		140	mA	

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} = 5.0V ±10%, GND = 0V*

Bus Parameters(3)

READ CYCLE

Symbol	Parameter	8253		8253-5		Unit
		Min	Max	Min	Max	
t_{AR}	Address Stable before $\overline{READ}$	50		30		ns
t_{RA}	Address Hold Time for $\overline{READ}$	5		5		ns
t_{RR}	$\overline{READ}$ Pulse Width	400		300		ns
t_{RD}	Data Delay from $\overline{READ}$(4)		300		200	ns
t_{DF}	$\overline{READ}$ to Data Floating	25	125	25	100	ns
t_{RV}	Recovery Time between $\overline{READ}$ and Any Other Control Signal	1		1		μs

A.C. CHARACTERISTICS (Continued)

WRITE CYCLE

Symbol	Parameter	8253		8253-5		Unit
		Min	Max	Min	Max	
t_{AW}	Address Stable before $\overline{\text{WRITE}}$	50		30		ns
t_{WA}	Address Hold Time for $\overline{\text{WRITE}}$	30		30		ns
t_{WW}	$\overline{\text{WRITE}}$ Pulse Width	400		300		ns
t_{DW}	Data Set Up Time for $\overline{\text{WRITE}}$	300		250		ns
t_{WD}	Data Hold Time for $\overline{\text{WRITE}}$	40		30		ns
t_{RV}	Recovery Time between $\overline{\text{WRITE}}$ and Any Other Control Signal	1		1		μs

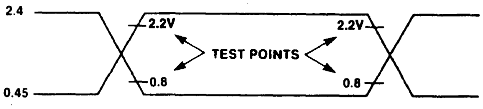
CLOCK AND GATE TIMING

Symbol	Parameter	8253		8253-5		Unit
		Min	Max	Min	Max	
t_{CLK}	Clock Period	380	dc	380	dc	ns
t_{PWH}	High Pulse Width	230		230		ns
t_{PWL}	Low Pulse Width	150		150		ns
t_{GW}	Gate Width High	150		150		ns
t_{GL}	Gate Width Low	100		100		ns
t_{GS}	Gate Set Up Time to CLK ↑	100		100		ns
t_{GH}	Gate Hold Time after CLK ↑	50		50		ns
t_{OD}	Output Delay from CLK ↓ (4)		400		400	ns
t_{ODG}	Output Delay from Gate ↓ (4)		300		300	ns

NOTES:
1. $I_{OL} = 2.2$ mA.
2. $I_{OH} = -400$ μA.
3. AC timings measured at V_{OH} 2.2, $V_{OL} = 0.8$.
4. $C_L = 150$ pF.

*For Extended Temperature EXPRESS, use M8253 electrical parameters.

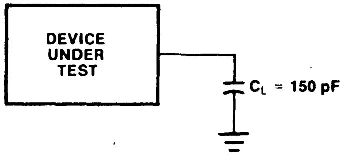
A.C. TESTING INPUT, OUTPUT WAVEFORM



231306-13

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 2.2V for a Logic "1" and 0.8V for a Logic "0".

A.C. TESTING LOAD CIRCUIT

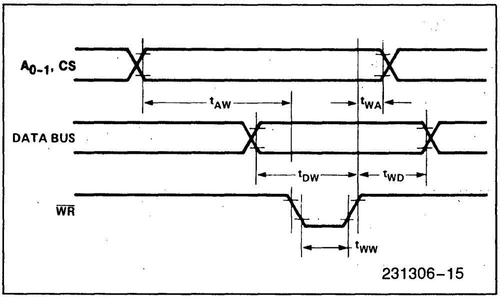


231306-14

C_L Includes Jig Capacitance

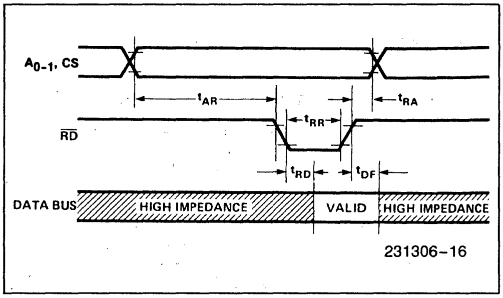
WAVEFORMS

WRITE TIMING



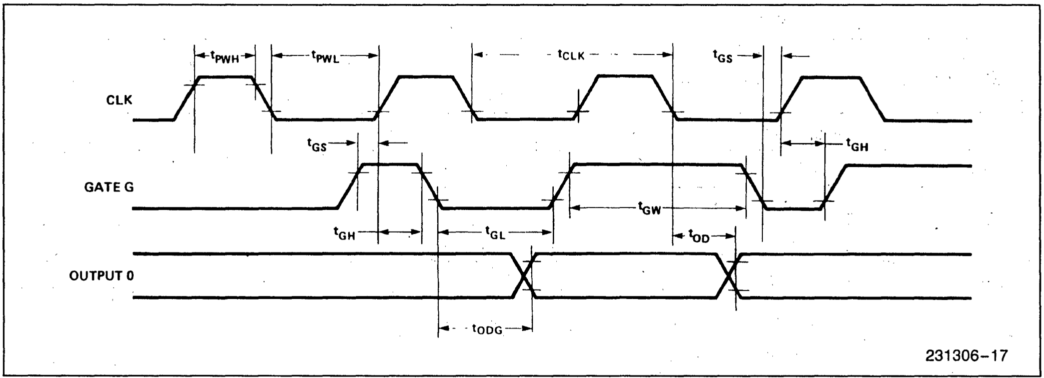
231306-15

READ TIMING



231306-16

CLOCK AND GATE TIMING



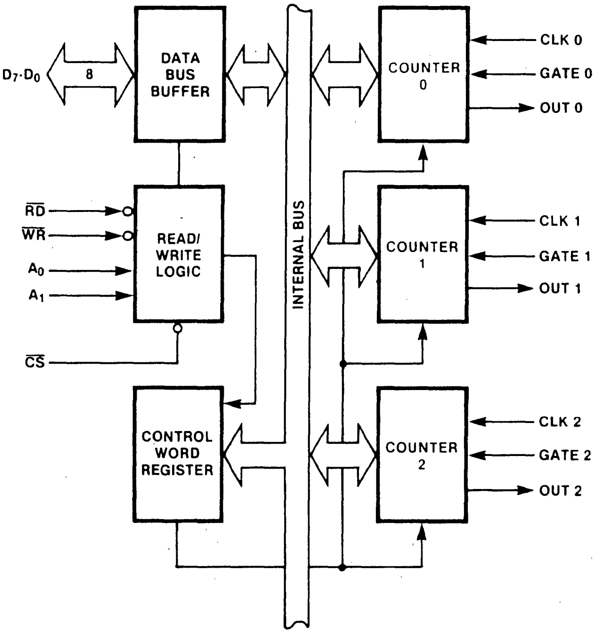
231306-17

8254
PROGRAMMABLE INTERVAL TIMER

- **Compatible with All Intel and Most Other Microprocessors**
- **Handles Inputs from DC to 10 MHz**
  - **— 5 MHz 8254-5**
  - **— 8 MHz 8254**
  - **—10 MHz 8254-2**
- **Status Read-Back Command**
- **Six Programmable Counter Modes**
- **Three Independent 16-Bit Counters**
- **Binary or BCD Counting**
- **Single +5V Supply**
- **Available in EXPRESS**
  - **—Standard Temperature Range**

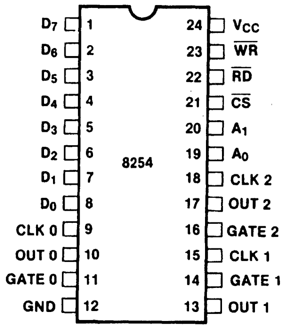
The Intel® 8254 is a counter/timer device designed to solve the common timing control problems in microcomputer system design. It provides three independent 16-bit counters, each capable of handling clock inputs up to 10 MHz. All modes are software programmable. The 8254 is a superset of the 8253.

The 8254 uses HMOS technology and comes in a 24-pin plastic or CERDIP package.



231164–1

Figure 1. 8254 Block Diagram



231164–2

Figure 2. Pin Configuration

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
D_7–D_0	1–8	I/O	**DATA:** Bi-directional three state data bus lines, connected to system data bus.
CLK 0	9	I	**CLOCK 0:** Clock input of Counter 0.
OUT 0	10	O	**OUTPUT 0:** Output of Counter 0.
GATE 0	11	I	**GATE 0:** Gate input of Counter 0.
GND	12		**GROUND:** Power supply connection.
V_{CC}	24		**POWER:** +5V power supply connection.
$\overline{WR}$	23	I	**WRITE CONTROL:** This input is low during CPU write operations.
$\overline{RD}$	22	I	**READ CONTROL:** This input is low during CPU read operations.
$\overline{CS}$	21	I	**CHIP SELECT:** A low on this input enables the 8254 to respond to $\overline{RD}$ and $\overline{WR}$ signals. $\overline{RD}$ and $\overline{WR}$ are ignored otherwise.
A_1, A_0	20–19	I	**ADDRESS:** Used to select one of the three Counters or the Control Word Register for read or write operations. Normally connected to the system address bus. **A_1 | A_0 | Selects** 0 | 0 | Counter 0 0 | 1 | Counter 1 1 | 0 | Counter 2 1 | 1 | Control Word Register
CLK 2	18	I	**CLOCK 2:** Clock input of Counter 2.
OUT 2	17	O	**OUT 2:** Output of Counter 2.
GATE 2	16	I	**GATE 2:** Gate input of Counter 2.
CLK 1	15	I	**CLOCK 1:** Clock input of Counter 1.
GATE 1	14	I	**GATE 1:** Gate input of Counter 1.
OUT 1	13	O	**OUT 1:** Output of Counter 1.

FUNCTIONAL DESCRIPTION

General

The 8254 is a programmable interval timer/counter designed for use with Intel microcomputer systems. It is a general purpose, multi-timing element that can be treated as an array of I/O ports in the system software.

The 8254 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control. Instead of setting up timing loops in software, the programmer configures the 8254 to match his requirements and programs one of the counters for the desired delay. After the desired delay, the 8254 will interrupt the CPU. Software overhead is minimal and variable length delays can easily be accommodated.

Some of the other counter/timer functions common to microcomputers which can be implemented with the 8254 are:

- Real time clock
- Event-counter
- Digital one-shot
- Programmable rate generator
- Square wave generator
- Binary rate multiplier
- Complex waveform generator
- Complex motor controller

Block Diagram

DATA BUS BUFFER

This 3-state, bi-directional, 8-bit buffer is used to interface the 8254 to the system bus (see Figure 3).

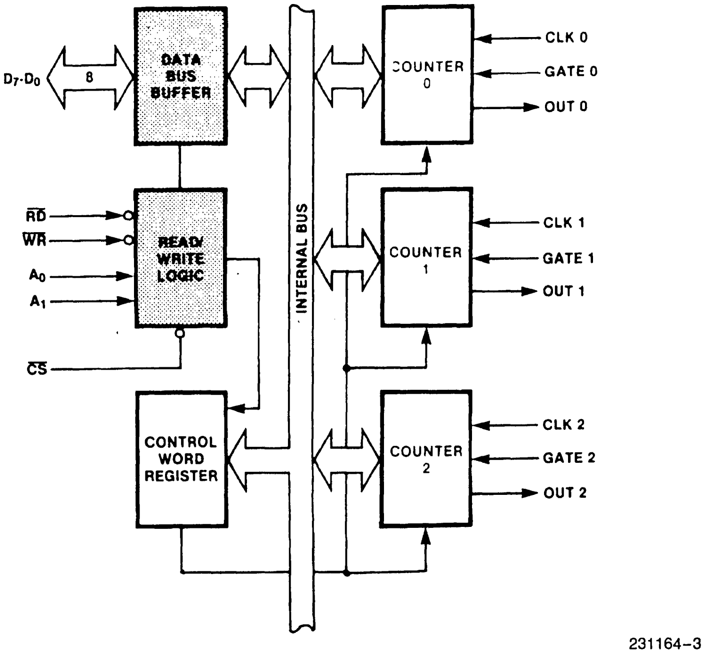


Figure 3. Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

READ/WRITE LOGIC

The Read/Write Logic accepts inputs from the system bus and generates control signals for the other functional blocks of the 8254. A_1 and A_0 select one of the three counters or the Control Word Register to be read from/written into. A "low" on the $\overline{RD}$ input tells the 8254 that the CPU is reading one of the counters. A "low" on the $\overline{WR}$ input tells the 8254 that the CPU is writing either a Control Word or an initial count. Both $\overline{RD}$ and $\overline{WR}$ are qualified by $\overline{CS}$; $\overline{RD}$ and $\overline{WR}$ are ignored unless the 8254 has been selected by holding $\overline{CS}$ low.

CONTROL WORD REGISTER

The Control Word Register (see Figure 4) is selected by the Read/Write Logic when $A_1,A_0 = 11$. If the CPU then does a write operation to the 8254, the data is stored in the Control Word Register and is interpreted as a Control Word used to define the operation of the Counters.

The Control Word Register can only be written to; status information is available with the Read-Back Command.

COUNTER 0, COUNTER 1, COUNTER 2

These three functional blocks are identical in operation, so only a single Counter will be described. The internal block diagram of a single counter is shown in Figure 5.

The Counters are fully independent. Each Counter may operate in a different Mode.

The Control Word Register is shown in the figure; it is not part of the Counter itself, but its contents determine how the Counter operates.

The status register, shown in Figure 5, when latched, contains the current contents of the Control Word Register and status of the output and null count flag. (See detailed explanation of the Read-Back command.)

The actual counter is labelled CE (for "Counting Element"). It is a 16-bit presettable synchronous down counter.

OL_M and OL_L are two 8-bit latches. OL stands for "Output Latch"; the subscripts M and L stand for "Most significant byte" and "Least significant byte"

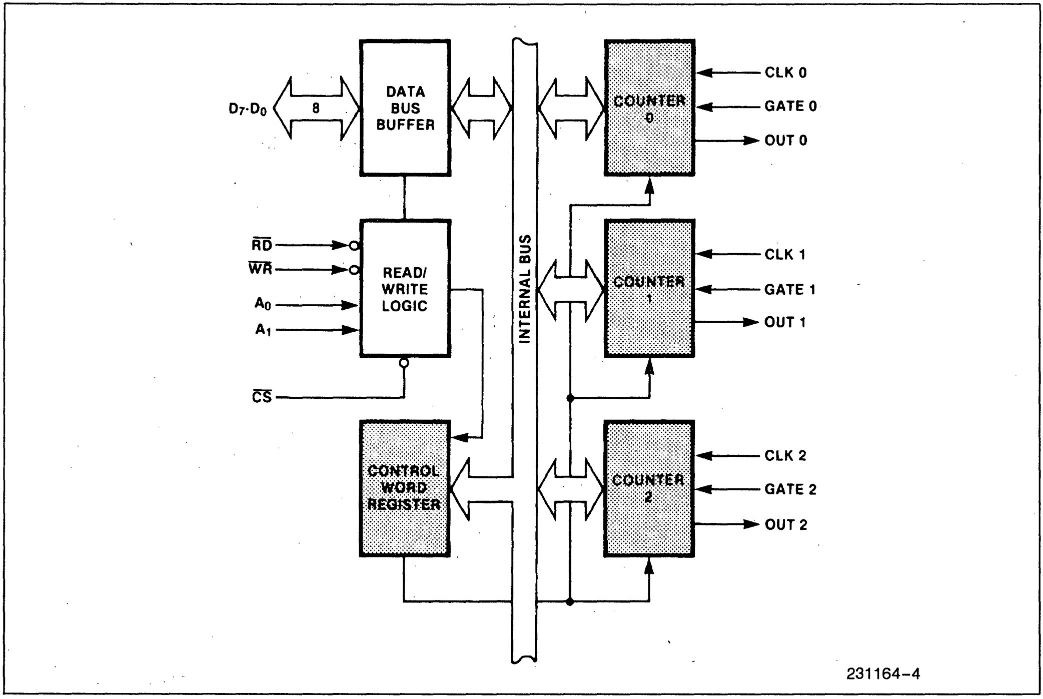


Figure 4. Block Diagram Showing Control Word Register and Counter Functions

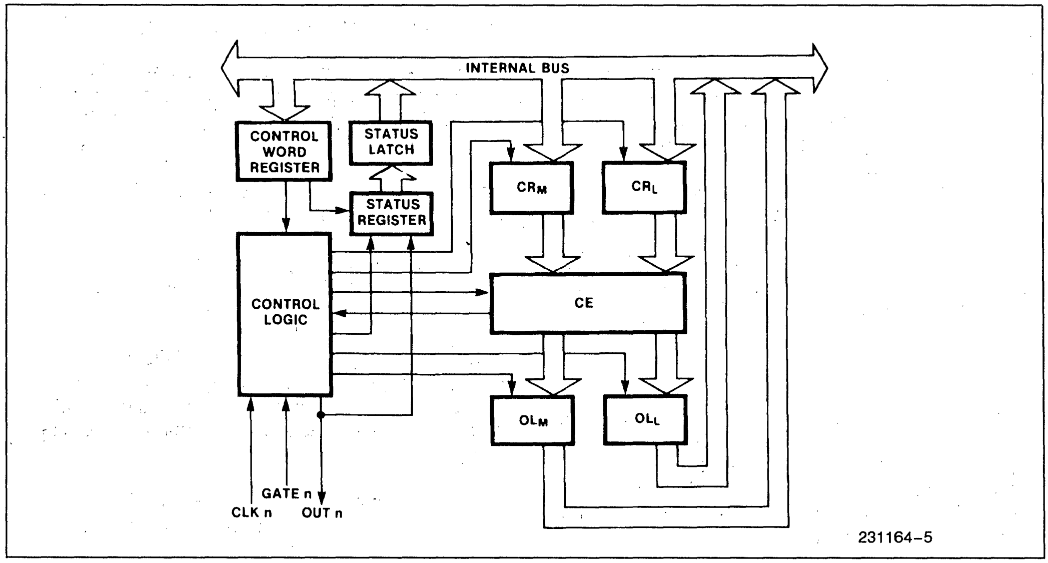


Figure 5. Internal Block Diagram of a Counter

respectively. Both are normally referred to as one unit and called just OL. These latches normally "follow" the CE, but if a suitable Counter Latch Command is sent to the 8254, the latches "latch" the present count until read by the CPU and then return to "following" the CE. One latch at a time is enabled by the counter's Control Logic to drive the internal bus. This is how the 16-bit Counter communicates over the 8-bit internal bus. Note that the CE itself cannot be read; whenever you read the count, it is the OL that is being read.

Similarly, there are two 8-bit registers called CR_M and CR_L (for "Count Register"). Both are normally referred to as one unit and called just CR. When a new count is written to the Counter, the count is stored in the CR and later transferred to the CE. The Control Logic allows one register at a time to be loaded from the internal bus. Both bytes are transferred to the CE simultaneously. CR_M and CR_L are cleared when the Counter is programmed. In this way, if the Counter has been programmed for one byte counts (either most significant byte only or least significant byte only) the other byte will be zero. Note that the CE cannot be written into; whenever a count is written, it is written into the CR.

The Control Logic is also shown in the diagram. CLK n, GATE n, and OUT n are all connected to the outside world through the Control Logic.

8254 SYSTEM INTERFACE

The 8254 is a component of the Intel Microcomputer Systems and interfaces in the same manner as all other peripherals of the family. It is treated by the system's software as an array of peripheral I/O ports; three are counters and the fourth is a control register for MODE programming.

Basically, the select inputs A_0,A_1 connect to the A_0, A_1 address bus signals of the CPU. The CS can be derived directly from the address bus using a linear select method. Or it can be connected to the output of a decoder, such as an Intel 8205 for larger systems.

OPERATIONAL DESCRIPTION

General

After power-up, the state of the 8254 is undefined. The Mode, count value, and output of all Counters are undefined.

How each Counter operates is determined when it is programmed. Each Counter must be programmed before it can be used. Unused counters need not be programmed.

Programming the 8254

Counters are programmed by writing a Control Word and then an initial count.

The Control Words are written into the Control Word Register, which is selected when A_1,A_0 = 11. The Control Word itself specifies which Counter is being programmed.

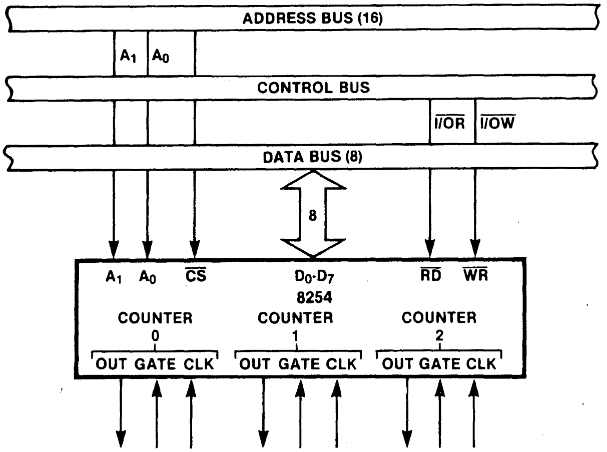


Figure 6. 8254 System Interface

Control Word Format

A_1,A_0 = 11 $\overline{CS}$ = 0 $\overline{RD}$ = 1 $\overline{WR}$ = 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC—Select Counter

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Read-Back Command (see Read Operations)

RW—Read/Write

RW1	RW0	
0	0	Counter Latch Command (see Read Operations)
0	1	Read/Write least significant byte only
1	0	Read/Write most significant byte only
1	1	Read/Write least significant byte first, then most significant byte

M—Mode

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

BCD

0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

NOTE:
Don't care bits (X) should be 0 to insure compatibility with future Intel products.

Figure 7. Control Word Format

By contrast, initial counts are written into the Counters, not the Control Word Register. The A_1,A_0 inputs are used to select the Counter to be written into. The format of the initial count is determined by the Control Word used.

Write Operations

The programming procedure for the 8254 is very flexible. Only two conventions need to be remembered:

1) For each Counter, the Control Word must be written before the initial count is written.

2) The initial count must follow the count format specified in the Control Word (least significant byte only, most significant byte only, or least significant byte and then most significant byte).

Since the Control Word Register and the three Counters have separate addresses (selected by the A_1,A_0 inputs), and each Control Word specifies the Counter it applies to (SC0,SC1 bits), no special instruction sequence is required. Any programming sequence that follows the conventions in Figure 7 is acceptable.

A new initial count may be written to a Counter at any time without affecting the Counter's programmed Mode in any way. Counting will be affected as described in the Mode definitions. The new count must follow the programmed count format.

If a Counter is 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 that same Counter. Otherwise, the Counter will be loaded with an incorrect count.

	A_1	A_0
Control Word—Counter 0	1	1
LSB of count—Counter 0	0	0
MSB of count—Counter 0	0	0
Control Word—Counter 1	1	1
LSB of count—Counter 1	0	1
MSB of count—Counter 1	0	1
Control Word—Counter 2	1	1
LSB of count—Counter 2	1	0
MSB of count—Counter 2	1	0

	A_1	A_0
Control Word—Counter 2	1	1
Control Word—Counter 1	1	1
Control Word—Counter 0	1	1
LSB of count—Counter 2	1	0
MSB of count—Counter 2	1	0
LSB of count—Counter 1	0	1
MSB of count—Counter 1	0	1
LSB of count—Counter 0	0	0
MSB of count—Counter 0	0	0

	A_1	A_0
Control Word—Counter 0	1	1
Control Word—Counter 1	1	1
Control Word—Counter 2	1	1
LSB of count—Counter 2	1	0
LSB of count—Counter 1	0	1
LSB of count—Counter 0	0	0
MSB of count—Counter 0	0	0
MSB of count—Counter 1	0	1
MSB of count—Counter 2	1	0

	A_1	A_0
Control Word—Counter 1	1	1
Control Word—Counter 0	1	1
LSB of count—Counter 1	0	1
Control Word—Counter 2	1	1
LSB of count—Counter 0	0	0
MSB of count—Counter 1	0	1
LSB of count—Counter 2	1	0
MSB of count—Counter 0	0	0
MSB of count—Counter 2	1	0

NOTE:
In all four examples, all Counters are programmed to read/write two-byte counts. These are only four of many possible programming sequences.

Figure 8. A Few Possible Programming Sequences

Read Operations

It is often desirable to read the value of a Counter without disturbing the count in progress. This is easily done in the 8254.

There are three possible methods for reading the counters: a simple read operation, the Counter Latch Command, and the Read-Back Command. Each is explained below. The first method is to perform a simple read operation. To read the Counter, which is selected with the A1, A0 inputs, the CLK input of the selected Counter must be inhibited by using either the GATE input or external logic. Otherwise, the count may be in the process of changing when it is read, giving an undefined result.

COUNTER LATCH COMMAND

The second method uses the "Counter Latch Command". Like a Control Word, this command is written to the Control Word Register, which is selected when A_1,A_0 = 11. Also like a Control Word, the SC0, SC1 bits select one of the three Counters, but two other bits, D5 and D4, distinguish this command from a Control Word.

A_1,A_0 = 11; CS = 0; RD = 1; WR = 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	0	0	X	X	X	X

SC1,SC0—specify counter to be latched

SC1	SC0	Counter
0	0	0
0	1	1
1	0	2
1	1	Read-Back Command

D5,D4—00 designates Counter Latch Command

X—don't care

NOTE:
Don't care bits (X) should be 0 to insure compatibility with future Intel products.

Figure 9. Counter Latching Command Format

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 CPU (or until the Counter 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 Counter's OL holds its count until it is read. Counter Latch Commands do not affect the programmed Mode of the Counter in any way.

If a Counter is latched and then, some time later, latched again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued.

With either method, the count must be read according to the programmed format; specifically, if the Counter is programmed for two byte counts, two bytes must be read. The two bytes do not have to be read one right after the other; read or write or programming operations of other Counters may be inserted between them.

Another feature of the 8254 is that reads and writes of the same Counter may be interleaved; for example, if the Counter 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 Counter 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 Counter. Otherwise, an incorrect count will be read.

READ-BACK COMMAND

The third method uses the Read-Back Command. This command allows the user to check the count value, programmed Mode, and current states of the OUT pin and Null Count flag of the selected counter(s).

The command is written into the Control Word Register and has the format shown in Figure 10. The command applies to the counters selected by setting their corresponding bits D3, D2, D1 = 1.

A0, A1 = 11 $\overline{\text{CS}}$ = 0 $\overline{\text{RD}}$ = 1 $\overline{\text{WR}}$ = 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	$\overline{\text{COUNT}}$	$\overline{\text{STATUS}}$	CNT 2	CNT 1	CNT 0	0

D_5: 0 = Latch count of selected counter(s)
D_4: 0 = Latch status of selected counters(s)
D_3: 1 = Select Counter 2
D_2: 1 = Select Counter 1
D_1: 1 = Select Counter 0
D_0: Reserved for future expansion; Must be 0

Figure 10. Read-Back Command Format

The read-back command may be used to latch multiple counter output latches (OL) by setting the $\overline{\text{COUNT}}$ bit D5 = 0 and selecting the desired counter(s). This single command is functionally equivalent to several counter latch commands, one for each counter latched. Each counter's latched count is held until it is read (or the counter is reprogrammed). The counter is automatically unlatched when read, but other counters remain latched until they are read. If multiple count read-back commands are issued to the same counter without reading the count, all but the first are ignored; i.e., the count which will be read is the count at the time the first read-back command was issued.

The read-back command may also be used to latch status information of selected counter(s) by setting $\overline{\text{STATUS}}$ bit D4 = 0. Status must be latched to be read; status of a counter is accessed by a read from that counter.

The counter status format is shown in Figure 11. Bits D5 through D0 contain the counter's programmed Mode exactly as written in the last Mode Control Word. OUTPUT bit D7 contains the current state of the OUT pin. This allows the user to monitor the counter's output via software, possibly eliminating some hardware from a system.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Output	Null Count	RW1	RW0	M2	M1	M0	BCD

D_7 1 = OUT Pin is 1
0 = OUT Pin is 0

D_6 1 = Null Count
0 = Count available for reading

D_5–D_0 Counter programmed mode (see Figure 7)

Figure 11. Status Byte

NULL COUNT bit D6 indicates when the last count written to the counter register (CR) has been loaded into the counting element (CE). The exact time this happens depends on the Mode of the counter and is described in the Mode Definitions, but until the count is loaded into the counting element (CE), it can't be read from the counter. If the count is latched or read before this time, the count value will not reflect the new count just written. The operation of Null Count is shown in Figure 12.

This Action	Causes
A. Write to the control word register;[1]	Null Count = 1
B. Write to the count register (CR);[2]	Null Count = 1
C. New Count is loaded into CE (CR → CE);	Null Count = 0

NOTE:

1. Only the counter specified by the control word will have its Null Count set to 1. Null count bits of other counters are unaffected.
2. If the counter is programmed for two-byte counts (least significant byte then most significant byte) Null Count goes to 1 when the second byte is written.

Figure 12. Null Count Operation

If multiple status latch operations of the counter(s) are performed without reading the status, all but the first are ignored; i.e., the status that will be read is the status of the counter at the time the first status read-back command was issued.

Both count and status of the selected counter(s) may be latched simultaneously by setting both $\overline{\text{COUNT}}$ and $\overline{\text{STATUS}}$ bits D5,D4 = 0. This is functionally the same as issuing two separate read-back commands at once, and the above discussions apply here also. Specifically, if multiple count and/or status read-back commands are issued to the same counter(s) without any intervening reads, all but the first are ignored. This is illustrated in Figure 13.

If both count and status of a counter are latched, the first read operation of that counter will return latched status, regardless of which was latched first. The next one or two reads (depending on whether the counter is programmed for one or two type counts) return latched count. Subsequent reads return unlatched count.

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	
0	1	0	0	0	Write into Counter 0
0	1	0	0	1	Write into Counter 1
0	1	0	1	0	Write into Counter 2
0	1	0	1	1	Write Control Word
0	0	1	0	0	Read from Counter 0
0	0	1	0	1	Read from Counter 1
0	0	1	1	0	Read from Counter 2
0	0	1	1	1	No-Operation (3-State)
1	X	X	X	X	No-Operation (3-State)
0	1	1	X	X	No-Operation (3-State)

Figure 14. Read/Write Operations Summary

Command								Description	Result
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
1	1	0	0	0	0	1	0	Read back count and status of Counter 0	Count and status latched for Counter 0
1	1	1	0	0	1	0	0	Read back status of Counter 1	Status latched for Counter 1
1	1	1	0	1	1	0	0	Read back status of Counters 2, 1	Status latched for Counter 2, but not Counter 1
1	1	0	1	1	0	0	0	Read back count of Counter 2	Count latched for Counter 2
1	1	0	0	0	1	0	0	Read back count and status of Counter 1	Count latched for Counter 1, but not status
1	1	1	0	0	0	1	0	Read back status of Counter 1	Command ignored, status already latched for Counter 1

Figure 13. Read-Back Command Example

Mode Definitions

The following are defined for use in describing the operation of the 8254.

CLK Pulse: a rising edge, then a falling edge, in that order, of a Counter's CLK input.

Trigger: a rising edge of a Counter's GATE input.

Counter loading: the transfer of a count from the CR to the CE (refer to the "Functional Description")

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.

GATE = 1 enables counting; GATE = 0 disables counting. GATE has no effect on OUT.

After the Control Word and initial count are written to a Counter, 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 Counter, 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 (no clock 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 = 0, it will still 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.

MODE 1: HARDWARE RETRIGGERABLE ONE-SHOT

OUT will be initially high. OUT will go low on the CLK pulse following a trigger to begin the one-shot pulse, and will remain low until the Counter reaches zero. OUT will then go high and remain high until the CLK pulse after the next trigger.

After writing the Control Word and initial count, the Counter is armed. A trigger results in loading the Counter and setting OUT low on the next CLK pulse, thus starting the one-shot pulse. An initial count of N will result in a one-shot pulse N CLK cycles in duration. The one-shot is retriggerable, hence OUT will remain low for N CLK pulses after any trigger. The one-shot pulse can be repeated without rewriting the same count into the counter. GATE has no effect on OUT.

If a new count is written to the Counter during a one-shot pulse, the current one-shot is not affected unless the counter is retriggered. In that case, the Counter is loaded with the new count and the one-shot pulse continues until the new count expires.

MODE 2: RATE GENERATOR

This Mode functions like 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. OUT then goes high again, the Counter reloads the initial count and the process is repeated. Mode 2 is periodic; the same sequence is repeated indefinitely. For an initial count of N, the sequence repeats every N CLK cycles.

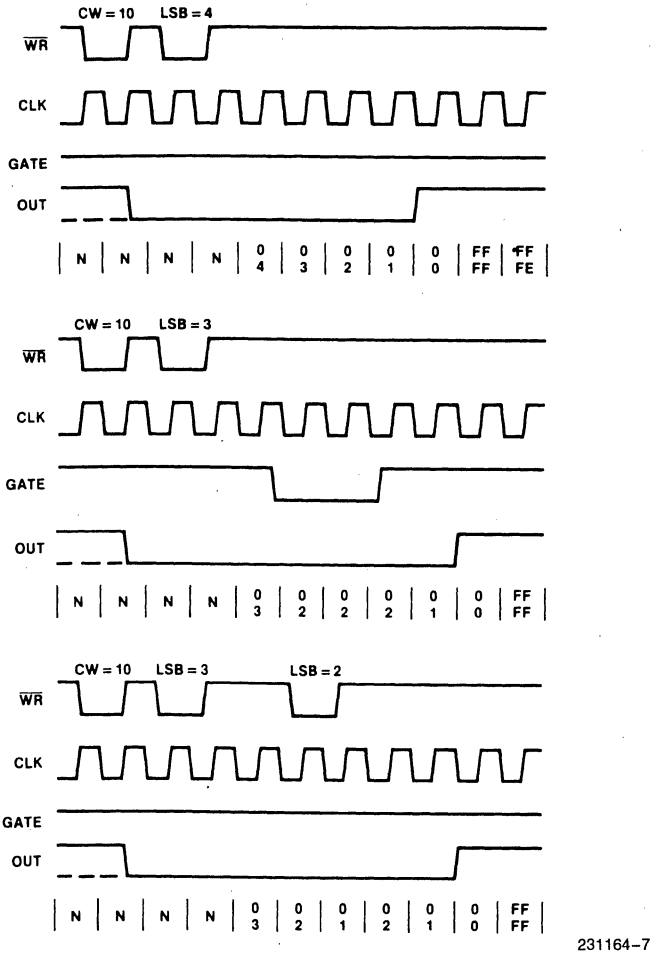
GATE = 1 enables counting; GATE = 0 disables counting. If GATE goes low during an output pulse, OUT is set high immediately. A trigger reloads the Counter with the initial count on the next CLK pulse; OUT goes low N CLK pulses after the trigger. Thus the GATE input can be used to synchronize the Counter.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. OUT goes low N CLK Pulses after the initial count is written. This allows the Counter to be synchronized by software also.

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 period, the Counter 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 counting cycle. In mode 2, a COUNT of 1 is illegal.

MODE 3: SQUARE WAVE MODE

Mode 3 is typically used for Baud rate generation. Mode 3 is similar to Mode 2 except for the duty cycle of OUT. OUT will initially be high. When half the


CW = 10
LSB = 4
WR
CLK
GATE
OUT
N | N | N | N | 0 4 | 0 3 | 0 2 | 0 1 | 0 0 | FF FF | FF FE
CW = 10
LSB = 3
WR
CLK
GATE
OUT
N | N | N | N | 0 3 | 0 2 | 0 2 | 0 2 | 0 1 | 0 0 | FF FF
CW = 10
LSB = 3
LSB = 2
WR
CLK
GATE
OUT
N | N | N | N | 0 3 | 0 2 | 0 1 | 0 2 | 0 1 | 0 0 | FF FF
231164-7

NOTE:
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 15. Mode 0

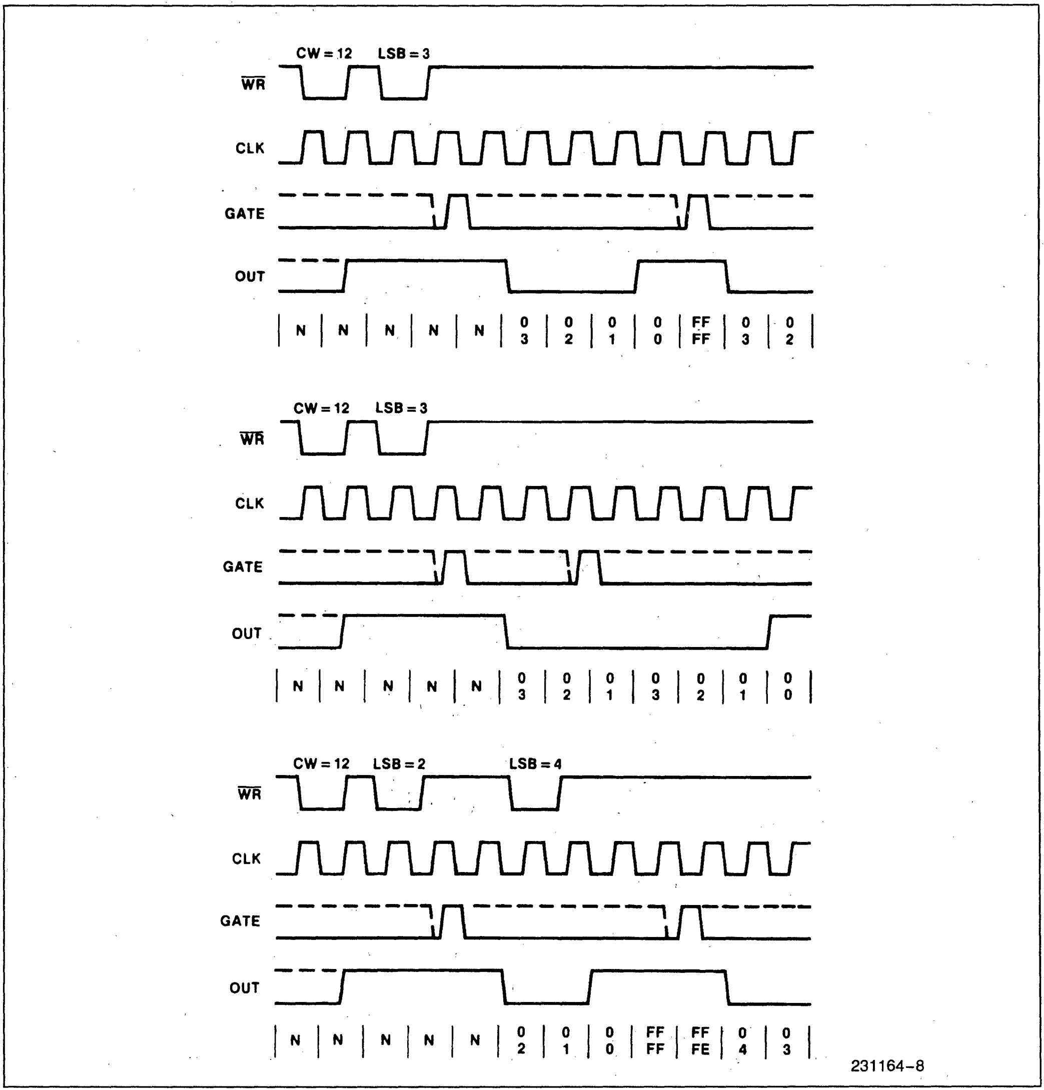

CW = 12 LSB = 3
WR
CLK
GATE
OUT
N N N N N 0 3 0 2 0 1 0 0 FF FF 0 3 0 2
CW = 12 LSB = 3
WR
CLK
GATE
OUT
N N N N N 0 3 0 2 0 1 0 3 0 2 0 1 0 0
CW = 12 LSB = 2 LSB = 4
WR
CLK
GATE
OUT
N N N N N 0 2 0 1 0 0 FF FF FF FE 0 4 0 3
231164-8

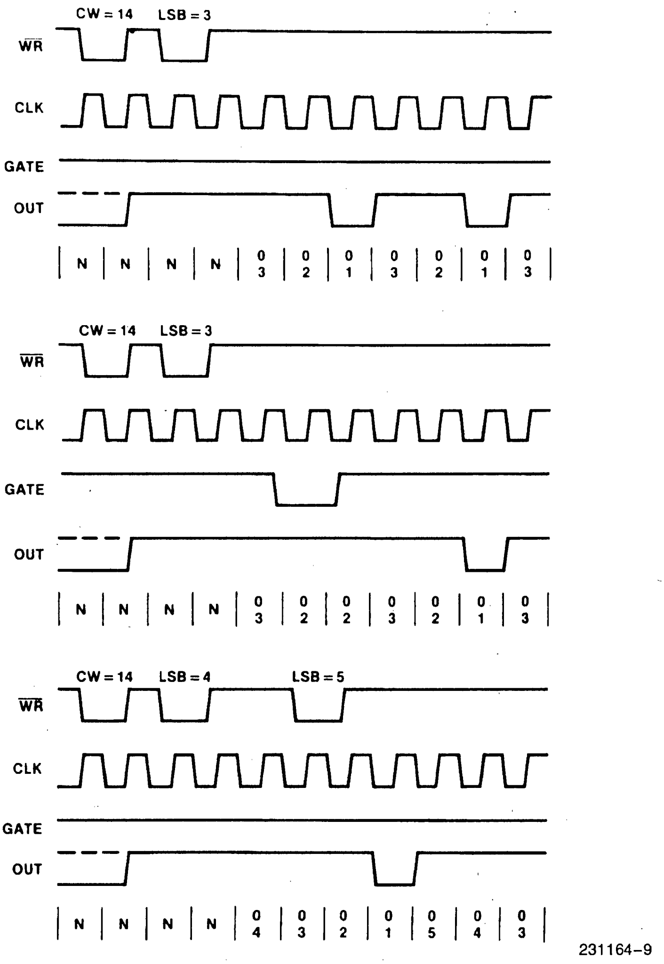
Figure 16. Mode 1

initial count has expired, OUT goes low for the remainder of the count. Mode 3 is periodic; the sequence above is repeated indefinitely. An initial count of N results in a square wave with a period of N CLK cycles.

GATE = 1 enables counting; GATE = 0 disables counting. If GATE goes low while OUT is low, OUT is set high immediately; no CLK pulse is required. A trigger reloads the Counter with the initial count on the next CLK pulse. Thus the GATE input can be used to synchronize the Counter.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This allows the Counter to be synchronized by software also.

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 Counter will be loaded with the new count on the next CLK pulse and counting will continue from the


CW = 14
LSB = 3
WR
CLK
GATE
OUT
N N N N 0 3 0 2 0 1 0 3 0 2 0 1 0 3
CW = 14
LSB = 3
WR
CLK
GATE
OUT
N N N N 0 3 0 2 0 2 0 3 0 2 0 1 0 3
CW = 14
LSB = 4
LSB = 5
WR
CLK
GATE
OUT
N N N N 0 4 0 3 0 2 0 1 0 5 0 4 0 3
231164-9

NOTE:
A GATE transition should not occur one clock prior to terminal count.

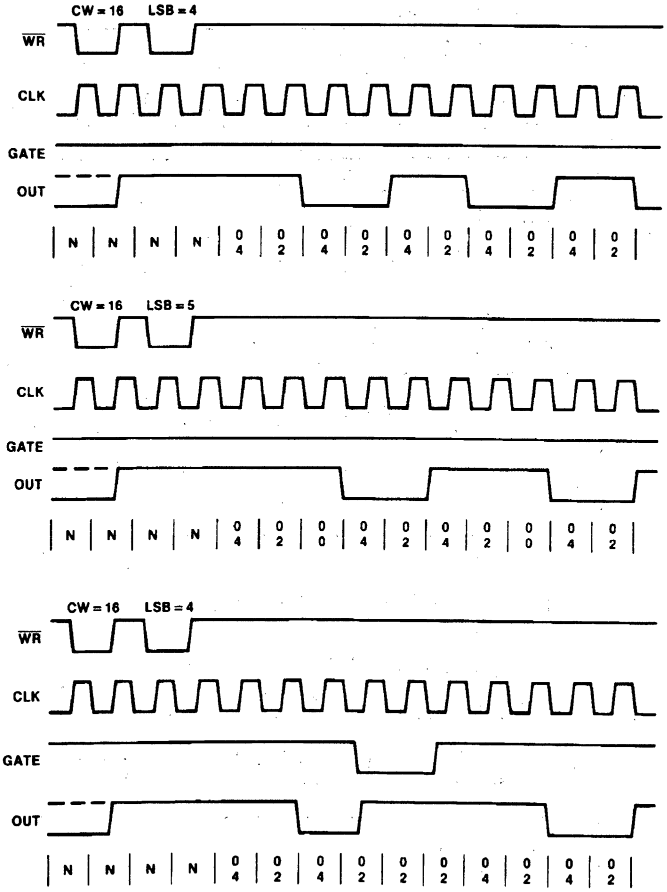
Figure 17. Mode 2

new count. Otherwise, the new count will be loaded at the end of the current half-cycle.

Mode 3 is implemented as follows:

Even counts: OUT is initially high. The initial count is loaded on one CLK pulse and then is decremented by two on succeeding CLK pulses. When the count expires OUT changes value and the Counter is reloaded with the initial count. The above process is repeated indefinitely.

Odd counts: OUT is initially high. The initial count minus one (an even number) is loaded on one CLK pulse and then is decremented by two on succeeding CLK pulses. One CLK pulse *after* the count expires, OUT goes low and the Counter is reloaded with the initial count minus one. Succeeding CLK pulses decrement the count by two. When the count expires, OUT goes high again and the Counter 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.


CW = 16 LSB = 4
WR
CLK
GATE
OUT
N N N N 0 4 0 2 0 4 0 2 0 4 0 2 0 4 0 2 0 4 0 2
CW = 16 LSB = 5
WR
CLK
GATE
OUT
N N N N 0 4 0 2 0 0 0 4 0 2 0 4 0 2 0 0 0 4 0 2
CW = 16 LSB = 4
WR
CLK
GATE
OUT
N N N N 0 4 0 2 0 4 0 2 0 2 0 2 0 4 0 2 0 4 0 2

231164–10

NOTE:
A GATE transition should not occur one clock prior to terminal count.

Figure 18. Mode 3

MODE 4: SOFTWARE TRIGGERED STROBE

OUT will be initially high. When the initial count expires, OUT will go low for one CLK pulse and then go high again. The counting sequence is "triggered" by writing the initial count.

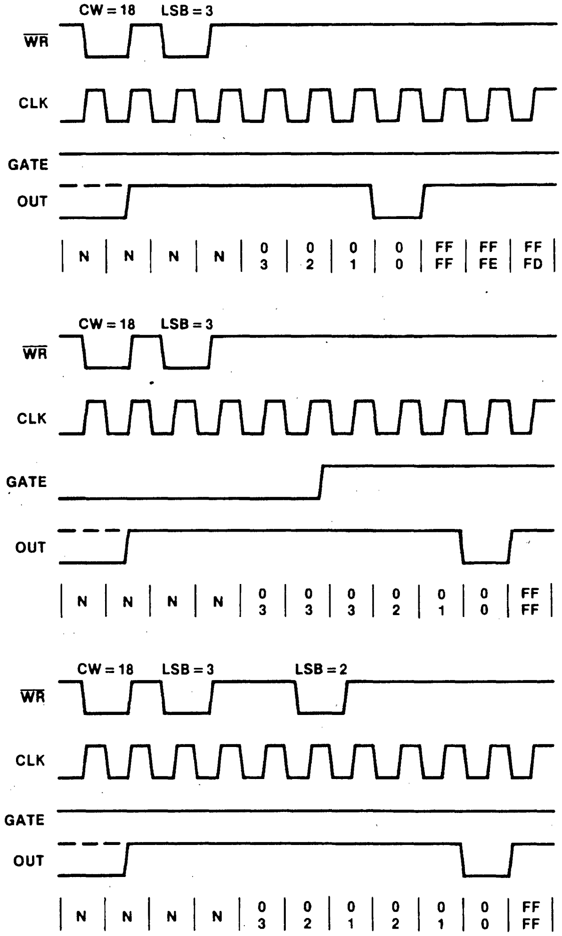
GATE = 1 enables counting; GATE = 0 disables counting. GATE has no effect on OUT.

After writing a Control Word and initial count, the Counter 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 the initial count is written.

If a new count is written during counting, 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 has no effect on counting.
2) Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the sequence to be "retriggered" by software. OUT strobes low N + 1 CLK pulses after the new count of N is written.


CW = 18
LSB = 3
WR
CLK
GATE
OUT
N N N N 0 3 0 2 0 1 0 0 FF FF FF FE FF FD
CW = 18
LSB = 3
WR
CLK
GATE
OUT
N N N N 0 3 0 3 0 3 0 2 0 1 0 0 FF FF
CW = 18
LSB = 3
LSB = 2
WR
CLK
GATE
OUT
N N N N 0 3 0 2 0 1 0 2 0 1 0 0 FF FF

231164-11

Figure 19. Mode 4

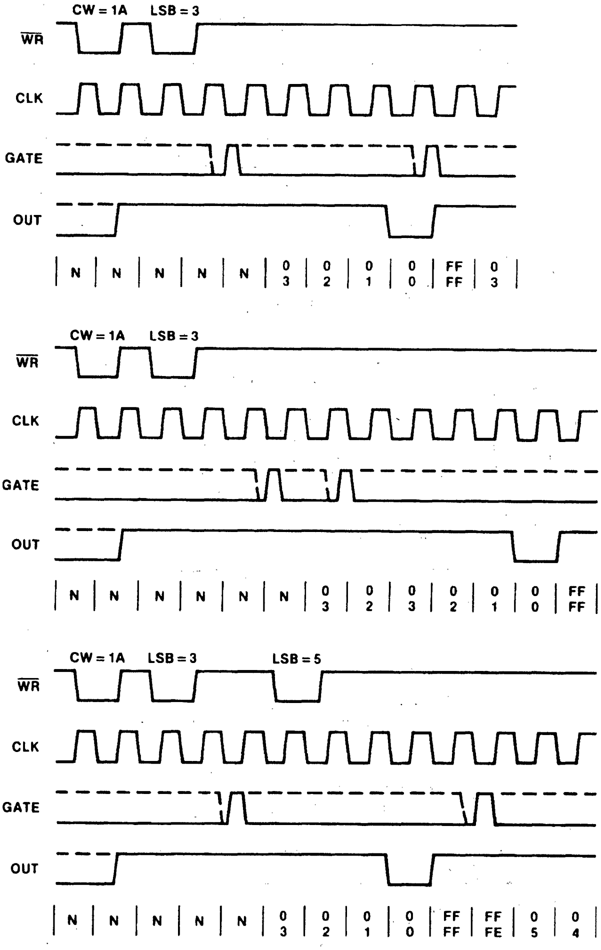
MODE 5: HARDWARE TRIGGERED STROBE (RETRIGGERABLE)

OUT will initially be high. Counting is triggered by a rising edge of GATE. When the initial count has expired, OUT will go low for one CLK pulse and then go high again.

After writing the Control Word and initial count, the counter will not be loaded until the CLK pulse after a trigger. 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 a trigger.

A trigger results in the Counter being loaded with the initial count on the next CLK pulse. The counting sequence is retriggerable. OUT will not strobe low for N + 1 CLK pulses after any trigger. GATE has no effect on OUT.

If a 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 Counter will be loaded with the new count on the next CLK pulse and counting will continue from there.


CW = 1A
LSB = 3
WR
CLK
GATE
OUT
N N N N N 0 3 0 2 0 1 0 0 FF FF 0 3
CW = 1A
LSB = 3
WR
CLK
GATE
OUT
N N N N N N 0 3 0 2 0 3 0 2 0 1 0 0 FF FF
CW = 1A
LSB = 3
LSB = 5
WR
CLK
GATE
OUT
N N N N N 0 3 0 2 0 1 0 0 FF FF FF FE 0 5 0 4

231164-12

Figure 20. Mode 5

Signal Status Modes	Low Or Going Low	Rising	High
0	Disables Counting	— —	Enables Counting
1	— —	1) Initiates Counting 2) Resets Output after Next Clock	— —
2	1) Disables Counting 2) Sets Output Immediately High	Initiates Counting	Enables Counting
3	1) Disables Counting 2) Sets Output Immediately High	Initiates Counting	Enables Counting
4	Disables Counting	— —	Enables Counting
5	— —	Initiates Counting	— —

Figure 21. Gate Pin Operations Summary

Mode	Min Count	Max Count
0	1	0
1	1	0
2	2	0
3	2	0
4	1	0
5	1	0

NOTE:
0 is equivalent to 2^{16} for binary counting and 10^4 for BCD counting.

Figure 22. Minimum and Maximum Initial Counts

Operation Common to All Modes

PROGRAMMING

When a Control Word is written to a Counter, all Control Logic is immediately reset and OUT goes to a known initial state; no CLK pulses are required for this.

GATE

The GATE input is always sampled on the rising edge of CLK. In Modes 0, 2, 3, and 4 the GATE input is level sensitive, and the logic level is sampled on the rising edge of CLK. 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 Counter. This flip-flop is then sampled on the next rising edge of CLK; the flip-flop is reset immediately after it is sampled. In this way, a trigger will be detected no matter when it occurs—a high logic level does not have to be maintained until the next rising edge of CLK. Note that in Modes 2 and 3, the GATE input is both edge- and level-sensitive. In Modes 2 and 3, if a CLK source other than the system clock is used, GATE should be pulsed immediately following $\overline{\text{WR}}$ of a new count value.

COUNTER

New counts are loaded and Counters are decremented on the falling edge of CLK.

The largest possible initial count is 0; this is equivalent to 2^{16} for binary counting and 10^4 for BCD counting.

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.

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} = 2.0 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −400 μA
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to 0.45V
I_{CC}	V_{CC} Supply Current		170	mA	
C_{IN}	Input Capacitance		10	pF	f_c = 1 MHz
$C_{I/0}$	I/O Capacitance		20	pF	Unmeasured pins returned to V_{SS}(4)

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

Bus Parameters(1)

READ CYCLE

Symbol	Parameter	8254-5		8254		8254-2		Unit
		Min	Max	Min	Max	Min	Max	
t_{AR}	Address Stable Before $\overline{RD}$ ↓	45		45		30		ns
t_{SR}	$\overline{CS}$ Stable Before $\overline{RD}$ ↓	0		0		0		ns
t_{RA}	Address Hold Time After $\overline{RD}$ ↑	0		0		0		ns
t_{RR}	$\overline{RD}$ Pulse Width	150		150		95		ns
t_{RD}	Data Delay from $\overline{RD}$ ↓		120		120		85	ns
t_{AD}	Data Delay from Address		220		220		185	ns
t_{DF}	$\overline{RD}$ ↑ to Data Floating	5	90	5	90	5	65	ns
t_{RV}	Command Recovery Time	200		200		165		ns

NOTE:
1. AC timings measured at V_{OH} = 2.0V, V_{OL} = 0.8V.

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

WRITE CYCLE

Symbol	Parameter	8254-5		8254		8254-2		Unit
		Min	Max	Min	Max	Min	Max	
t_{AW}	Address Stable Before $\overline{\text{WR}}$ ↓	0		0		0		ns
t_{SW}	$\overline{\text{CS}}$ Stable Before $\overline{\text{WR}}$ ↓	0		0		0		ns
$t_{\overline{WA}}$	Address Hold Time After $\overline{\text{WR}}$ ↓	0		0		0		ns
t_{WW}	$\overline{\text{WR}}$ Pulse Width	150		150		95		ns
t_{DW}	Data Setup Time Before $\overline{\text{WR}}$ ↑	120		120		95		ns
t_{WD}	Data Hold Time After $\overline{\text{WR}}$ ↑	0		0		0		ns
t_{RV}	Command Recovery Time	200		200		165		ns

CLOCK AND GATE

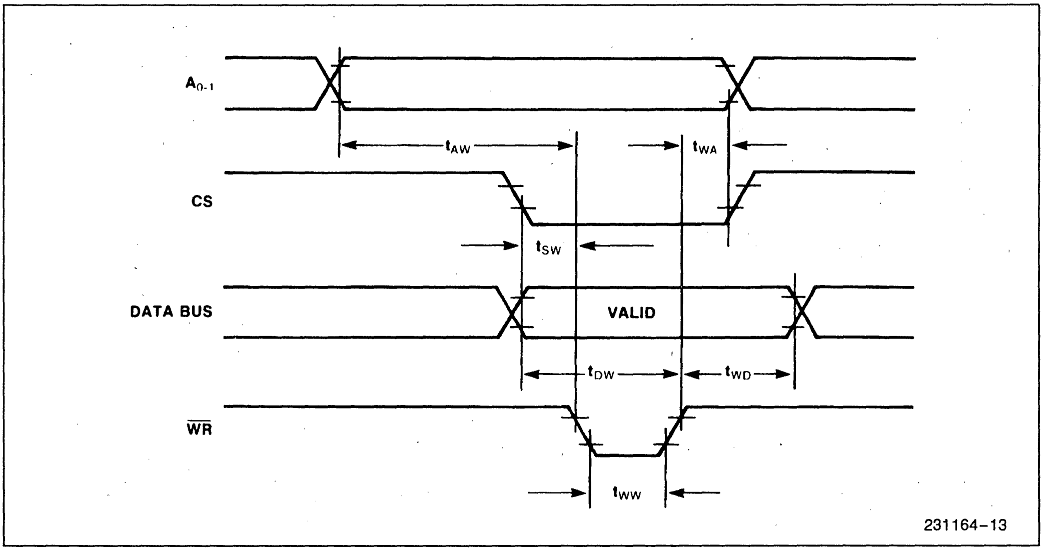
Symbol	Parameter	8254-5		8254		8254-2		Unit
		Min	Max	Min	Max	Min	Max	
t_{CLK}	Clock Period	200	DC	125	DC	100	DC	ns
t_{PWH}	High Pulse Width	60(3)		60(3)		30(3)		ns
t_{PWL}	Low Pulse Width	60(3)		60(3)		50(3)		ns
t_R	Clock Rise Time		25		25		25	ns
t_F	Clock Fall Time		25		25		25	ns
t_{GW}	Gate Width High	50		50		50		ns
t_{GL}	Gate Width Low	50		50		50		ns
t_{GS}	Gate Setup Time to CLK ↑	50		50		40		ns
t_{GH}	Gate Setup Time After CLK ↑	50(2)		50(2)		50(2)		ns
t_{OD}	Output Delay from CLK ↓		150		150		100	ns
t_{ODG}	Output Delay from Gate ↓		120		120		100	ns
t_{WC}	CLK Delay for Loading ↓	0	55	0	55	0	55	ns
t_{WG}	Gate Delay for Sampling	−5	50	−5	50	−5	40	ns
t_{WO}	OUT Delay from Mode Write		260		260		240	ns
t_{CL}	CLK Set Up for Count Latch	−40	45	−40	45	−40	40	ns

NOTES:

2. In Modes 1 and 5 triggers are sampled on each rising clock edge. A second trigger within 120 ns (70 ns for the 8254-2) of the rising clock edge may not be detected.
3. Low-going glitches that violate t_{PWH}, t_{PWL} may cause errors requiring counter reprogramming.
4. Sampled, not 100% tested. T_A = 25°C.
5. If CLK present at TWC min then Count equals N+2 CLK pulses, TWC max equals Count N+1 CLK pulse. TWC min to TWC max, count will be either N+1 or N+2 CLK pulses.
6. In Modes 1 and 5, if GATE is present when writing a new Count value, at TWG min Counter will not be triggered, at TWG max Counter will be triggered.
7. If CLK present when writing a Counter Latch or ReadBack Command, at TCL min CLK will be reflected in count value latched, at TCL max CLK will not be reflected in the count value latched.

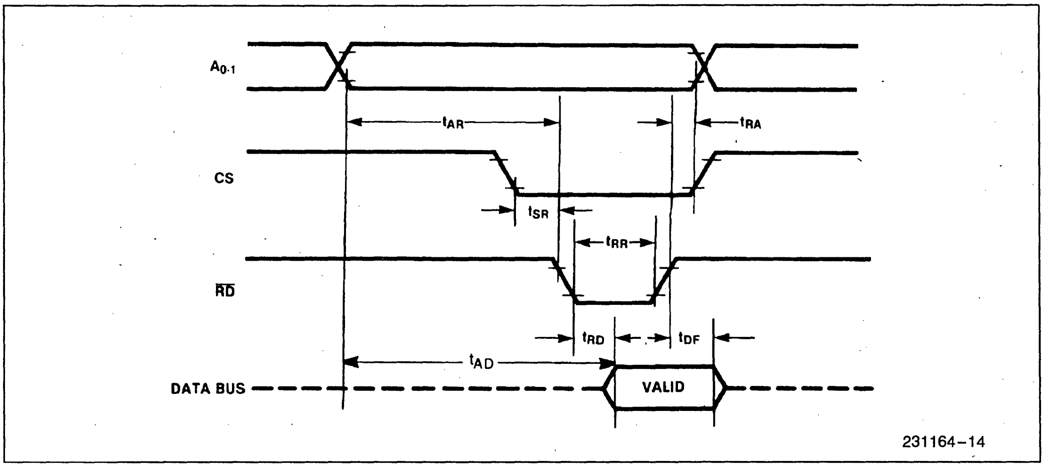
WAVEFORMS

WRITE



231164-13

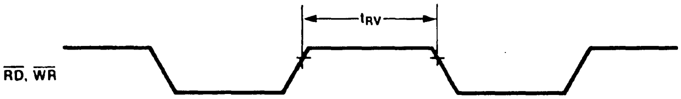
READ



231164-14

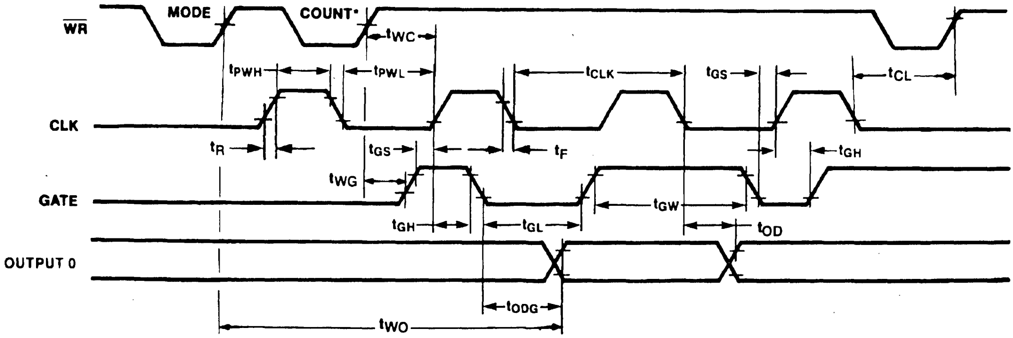
WAVEFORMS (Continued)

RECOVERY



231164-15

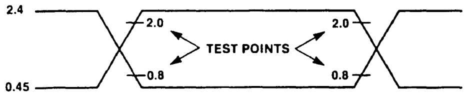
CLOCK AND GATE



231164-16

*Last byte of count being written.

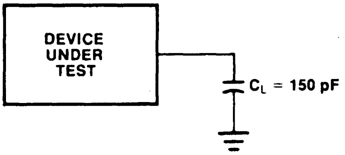
A.C. TESTING INPUT, OUTPUT WAVEFORM



231164-17

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 2.0V for a Logic "1" and 0.8V for a Logic "0".

A.C. TESTING LOAD CIRCUIT



231164-18

C_L = 150 pF
C_L Includes Jig Capacitance

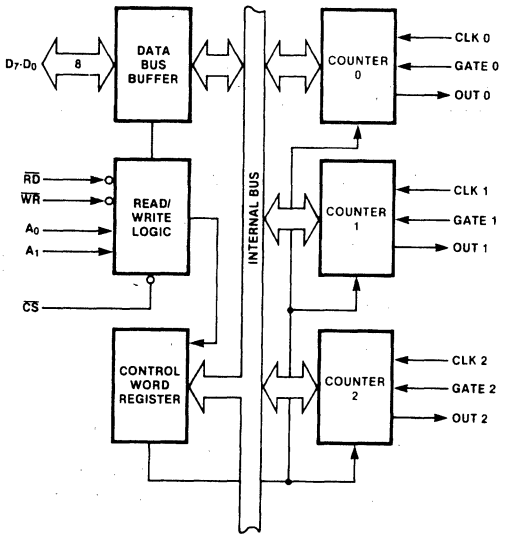
82C54
CHMOS PROGRAMMABLE INTERVAL TIMER

- **Compatible with all Intel and most other microprocessors**
- **High Speed, "Zero Wait State" Operation with 8 MHz 8086/88 and 80186/188**
- **Handles Inputs from DC to 8 MHz**
  — 10 MHz for 82C54-2
- **Available in EXPRESS**
  — Standard Temperature Range
  — Extended Temperature Range
- **Three independent 16-bit counters**
- **Low Power CHMOS**
  — I_{CC} = 10 mA @ 8 MHz Count frequency
- **Completely TTL Compatible**
- **Six Programmable Counter Modes**
- **Binary or BCD counting**
- **Status Read Back Command**
- **Available in 24-Pin DIP and 28-Pin PLCC**

The Intel 82C54 is a high-performance, CHMOS version of the industry standard 8254 counter/timer which is designed to solve the timing control problems common in microcomputer system design. It provides three independent 16-bit counters, each capable of handling clock inputs up to 10 MHz. All modes are software programmable. The 82C54 is pin compatible with the HMOS 8254, and is a superset of the 8253.

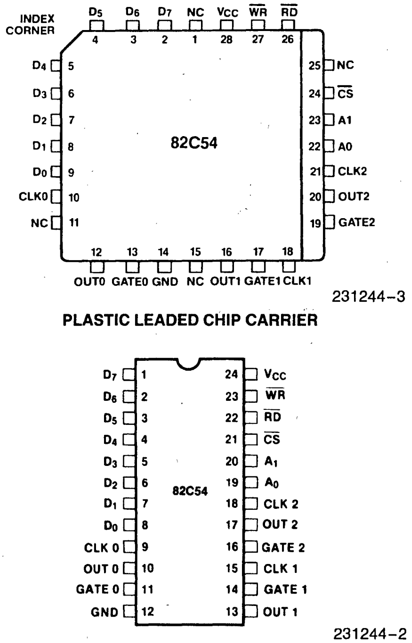
Six programmable timer modes allow the 82C54 to be used as an event counter, elapsed time indicator, programmable one-shot, and in many other applications.

The 82C54 is fabricated on Intel's advanced CHMOS III technology which provides low power consumption with performance equal to or greater than the equivalent HMOS product. The 82C54 is available in 24-pin DIP and 28-pin plastic leaded chip carrier (PLCC) packages.



231244-1

Figure 1. 82C54 Block Diagram



231244-3

PLASTIC LEADED CHIP CARRIER

231244-2

Diagrams are for pin reference only.
Package sizes are not to scale.

Figure 2. 82C54 Pinout

October 1987
Order Number: 231244-004

Table 1. Pin Description

<table>
<tr><th rowspan="2">Symbol</th><th colspan="2">Pin Number</th><th rowspan="2">Type</th><th colspan="3" rowspan="2">Function</th></tr>
<tr><th>DIP</th><th>PLCC</th></tr>
<tr><td>D_7-D_0</td><td>1-8</td><td>2-9</td><td>I/O</td><td colspan="3">Data: Bidirectional tri-state data bus lines, connected to system data bus.</td></tr>
<tr><td>CLK 0</td><td>9</td><td>10</td><td>I</td><td colspan="3">Clock 0: Clock input of Counter 0.</td></tr>
<tr><td>OUT 0</td><td>10</td><td>12</td><td>O</td><td colspan="3">Output 0: Output of Counter 0.</td></tr>
<tr><td>GATE 0</td><td>11</td><td>13</td><td>I</td><td colspan="3">Gate 0: Gate input of Counter 0.</td></tr>
<tr><td>GND</td><td>12</td><td>14</td><td></td><td colspan="3">Ground: Power supply connection.</td></tr>
<tr><td>OUT 1</td><td>13</td><td>16</td><td>O</td><td colspan="3">Out 1: Output of Counter 1.</td></tr>
<tr><td>GATE 1</td><td>14</td><td>17</td><td>I</td><td colspan="3">Gate 1: Gate input of Counter 1.</td></tr>
<tr><td>CLK 1</td><td>15</td><td>18</td><td>I</td><td colspan="3">Clock 1: Clock input of Counter 1.</td></tr>
<tr><td>GATE 2</td><td>16</td><td>19</td><td>I</td><td colspan="3">Gate 2: Gate input of Counter 2.</td></tr>
<tr><td>OUT 2</td><td>17</td><td>20</td><td>O</td><td colspan="3">Out 2: Output of Counter 2.</td></tr>
<tr><td>CLK 2</td><td>18</td><td>21</td><td>I</td><td colspan="3">Clock 2: Clock input of Counter 2.</td></tr>
<tr><td rowspan="6">A_1, A_0</td><td rowspan="6">20-19</td><td rowspan="6">23-22</td><td rowspan="6">I</td><td colspan="3">Address: Used to select one of the three Counters or the Control Word Register for read or write operations. Normally connected to the system address bus.</td></tr>
<tr><th>A_1</th><th>A_0</th><th>Selects</th></tr>
<tr><td>0</td><td>0</td><td>Counter 0</td></tr>
<tr><td>0</td><td>1</td><td>Counter 1</td></tr>
<tr><td>1</td><td>0</td><td>Counter 2</td></tr>
<tr><td>1</td><td>1</td><td>Control Word Register</td></tr>
<tr><td>$\overline{CS}$</td><td>21</td><td>24</td><td>I</td><td colspan="3">Chip Select: A low on this input enables the 82C54 to respond to $\overline{RD}$ and $\overline{WR}$ signals. $\overline{RD}$ and $\overline{WR}$ are ignored otherwise.</td></tr>
<tr><td>$\overline{RD}$</td><td>22</td><td>26</td><td>I</td><td colspan="3">Read Control: This input is low during CPU read operations.</td></tr>
<tr><td>$\overline{WR}$</td><td>23</td><td>27</td><td>I</td><td colspan="3">Write Control: This input is low during CPU write operations.</td></tr>
<tr><td>V_{CC}</td><td>24</td><td>28</td><td></td><td colspan="3">Power: +5V power supply connection.</td></tr>
<tr><td>NC</td><td></td><td>1, 11, 15, 25</td><td></td><td colspan="3">No Connect</td></tr>
</table>

FUNCTIONAL DESCRIPTION

General

The 82C54 is a programmable interval timer/counter designed for use with Intel microcomputer systems. It is a general purpose, multi-timing element that can be treated as an array of I/O ports in the system software.

The 82C54 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control. Instead of setting up timing loops in software, the programmer configures the 82C54 to match his requirements and programs one of the counters for the desired delay. After the desired delay, the 82C54 will interrupt the CPU. Software overhead is minimal and variable length delays can easily be accommodated.

Some of the other counter/timer functions common to microcomputers which can be implemented with the 82C54 are:

- Real time clock
- Even counter
- Digital one-shot
- Programmable rate generator
- Square wave generator
- Binary rate multiplier
- Complex waveform generator
- Complex motor controller

Block Diagram

DATA BUS BUFFER

This 3-state, bi-directional, 8-bit buffer is used to interface the 82C54 to the system bus (see Figure 3).

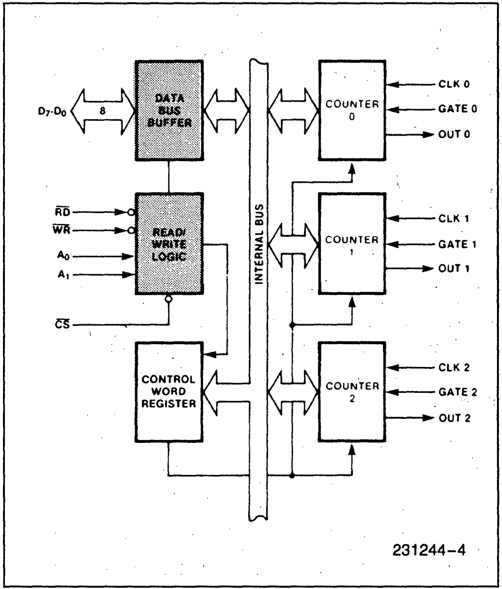


Figure 3. Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

READ/WRITE LOGIC

The Read/Write Logic accepts inputs from the system bus and generates control signals for the other functional blocks of the 82C54. A_1 and A_0 select one of the three counters or the Control Word Register to be read from/written into. A "low" on the $\overline{RD}$ input tells the 82C54 that the CPU is reading one of the counters. A "low" on the $\overline{WR}$ input tells the 82C54 that the CPU is writing either a Control Word or an initial count. Both $\overline{RD}$ and $\overline{WR}$ are qualified by $\overline{CS}$; $\overline{RD}$ and $\overline{WR}$ are ignored unless the 82C54 has been selected by holding $\overline{CS}$ low.

CONTROL WORD REGISTER

The Control Word Register (see Figure 4) is selected by the Read/Write Logic when $A_1, A_0 = 11$. If the CPU then does a write operation to the 82C54, the data is stored in the Control Word Register and is interpreted as a Control Word used to define the operation of the Counters.

The Control Word Register can only be written to; status information is available with the Read-Back Command.

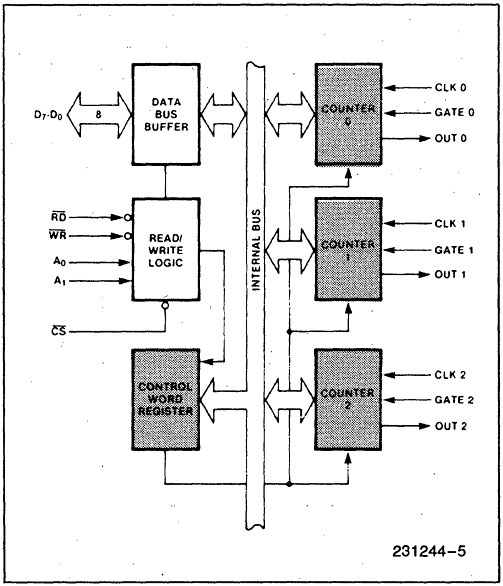


Figure 4. Block Diagram Showing Control Word Register and Counter Functions

COUNTER 0, COUNTER 1, COUNTER 2

These three functional blocks are identical in operation, so only a single Counter will be described. The internal block diagram of a single counter is shown in Figure 5.

The Counters are fully independent. Each Counter may operate in a different Mode.

The Control Word Register is shown in the figure; it is not part of the Counter itself, but its contents determine how the Counter operates.

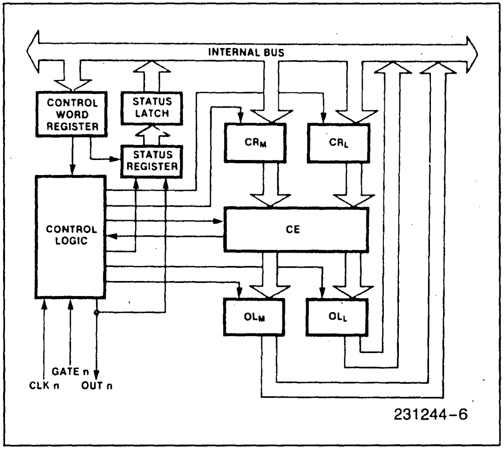


Figure 5. Internal Block Diagram of a Counter

The status register, shown in the Figure, when latched, contains the current contents of the Control Word Register and status of the output and null count flag. (See detailed explanation of the Read-Back command.)

The actual counter is labelled CE (for "Counting Element"). It is a 16-bit presettable synchronous down counter.

OL_M and OL_L are two 8-bit latches. OL stands for "Output Latch"; the subscripts M and L stand for "Most significant byte" and "Least significant byte" respectively. Both are normally referred to as one unit and called just OL. These latches normally "follow" the CE, but if a suitable Counter Latch Command is sent to the 82C54, the latches "latch" the present count until read by the CPU and then return to "following" the CE. One latch at a time is enabled by the counter's Control Logic to drive the internal bus. This is how the 16-bit Counter communicates over the 8-bit internal bus. Note that the CE itself cannot be read; whenever you read the count, it is the OL that is being read.

Similarly, there are two 8-bit registers called CR_M and CR_L (for "Count Register"). Both are normally referred to as one unit and called just CR. When a new count is written to the Counter, the count is stored in the CR and later transferred to the CE. The Control Logic allows one register at a time to be loaded from the internal bus. Both bytes are transferred to the CE simultaneously. CR_M and CR_L are cleared when the Counter is programmed. In this way, if the Counter has been programmed for one byte counts (either most significant byte only or least significant byte only) the other byte will be zero. Note that the CE cannot be written into; whenever a count is written, it is written into the CR.

The Control Logic is also shown in the diagram. CLK n, GATE n, and OUT n are all connected to the outside world through the Control Logic.

82C54 SYSTEM INTERFACE

The 82C54 is treated by the systems software as an array of peripheral I/O ports; three are counters and the fourth is a control register for MODE programming.

Basically, the select inputs A_0, A_1 connect to the A_0, A_1 address bus signals of the CPU. The $\overline{CS}$ can be derived directly from the address bus using a linear select method. Or it can be connected to the output of a decoder, such as an Intel 8205 for larger systems.

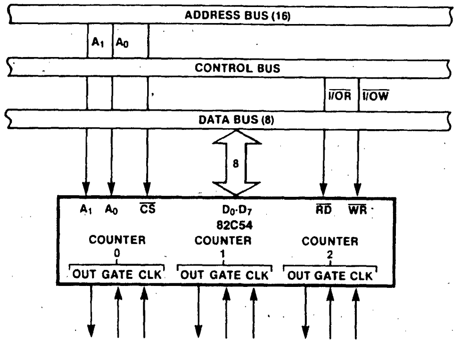


Figure 6. 82C54 System Interface

OPERATIONAL DESCRIPTION

General

After power-up, the state of the 82C54 is undefined. The Mode, count value, and output of all Counters are undefined.

How each Counter operates is determined when it is programmed. Each Counter must be programmed before it can be used. Unused counters need not be programmed.

Programming the 82C54

Counters are programmed by writing a Control Word and then an initial count. The control word format is shown in Figure 7.

All Control Words are written into the Control Word Register, which is selected when $A_1, A_0 = 11$. The Control Word itself specifies which Counter is being programmed.

By contrast, initial counts are written into the Counters, not the Control Word Register. The A_1, A_0 inputs are used to select the Counter to be written into. The format of the initial count is determined by the Control Word used.

Control Word Format

$A_1, A_0 = 11$ $\overline{CS} = 0$ $\overline{RD} = 1$ $\overline{WR} = 0$

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC — Select Counter:

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Read-Back Command (See Read Operations)

RW — Read/Write:

RW1	RW0	
0	0	Counter Latch Command (see Read Operations)
0	1	Read/Write least significant byte only.
1	0	Read/Write most significant byte only.
1	1	Read/Write least significant byte first, then most significant byte.

M — MODE:

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

BCD:

0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

NOTE: Don't care bits (X) should be 0 to insure compatibility with future Intel products.

Figure 7. Control Word Format

Write Operations

The programming procedure for the 82C54 is very flexible. Only two conventions need to be remembered:

1) For each Counter, the Control Word must be written before the initial count is written.
2) The initial count must follow the count format specified in the Control Word (least significant byte only, most significant byte only, or least significant byte and then most significant byte).

Since the Control Word Register and the three Counters have separate addresses (selected by the A_1, A_0 inputs), and each Control Word specifies the Counter it applies to (SC0, SC1 bits), 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 Counter at any time without affecting the Counter's programmed Mode in any way. Counting will be affected as described in the Mode definitions. The new count must follow the programmed count format.

If a Counter is 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 that same Counter. Otherwise, the Counter will be loaded with an incorrect count.

		A_1	A_0
Control Word —	Counter 0	1	1
LSB of count —	Counter 0	0	0
MSB of count —	Counter 0	0	0
Control Word —	Counter 1	1	1
LSB of count —	Counter 1	0	1
MSB of count —	Counter 1	0	1
Control Word —	Counter 2	1	1
LSB of count —	Counter 2	1	0
MSB of count —	Counter 2	1	0

		A_1	A_0
Control Word —	Counter 2	1	1
Control Word —	Counter 1	1	1
Control Word —	Counter 0	1	1
LSB of count —	Counter 2	1	0
MSB of count —	Counter 2	1	0
LSB of count —	Counter 1	0	1
MSB of count —	Counter 1	0	1
LSB of count —	Counter 0	0	0
MSB of count —	Counter 0	0	0

		A_1	A_0
Control Word —	Counter 0	1	1
Counter Word —	Counter 1	1	1
Control Word —	Counter 2	1	1
LSB of count —	Counter 2	1	0
LSB of count —	Counter 1	0	1
LSB of count —	Counter 0	0	0
MSB of count —	Counter 0	0	0
MSB of count —	Counter 1	0	1
MSB of count —	Counter 2	1	0

		A_1	A_0
Control Word —	Counter 1	1	1
Control Word —	Counter 0	1	1
LSB of count —	Counter 1	0	1
Control Word —	Counter 2	1	1
LSB of count —	Counter 0	0	0
MSB of count —	Counter 1	0	1
LSB of count —	Counter 2	1	0
MSB of count —	Counter 0	0	0
MSB of count —	Counter 2	1	0

NOTE:
In all four examples, all counters are programmed to read/write two-byte counts.
These are only four of many possible programming sequences.

Figure 8. A Few Possible Programming Sequences

Read Operations

It is often desirable to read the value of a Counter without disturbing the count in progress. This is easily done in the 82C54.

There are three possible methods for reading the counters: a simple read operation, the Counter Latch Command, and the Read-Back Command. Each is explained below. The first method is to perform a simple read operation. To read the Counter, which is selected with the A1, A0 inputs, the CLK input of the selected Counter must be inhibited by using either the GATE input or external logic. Otherwise, the count may be in the process of changing when it is read, giving an undefined result.

COUNTER LATCH COMMAND

The second method uses the "Counter Latch Command". Like a Control Word, this command is written to the Control Word Register, which is selected when A_1, A_0 = 11. Also like a Control Word, the SC0, SC1 bits select one of the three Counters, but two other bits, D5 and D4, distinguish this command from a Control Word.

A_1, A_0 = 11; $\overline{CS}$ = 0; $\overline{RD}$ = 1; $\overline{WR}$ = 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	0	0	X	X	X	X

SC1, SC0 - specify counter to be latched

SC1	SC0	Counter
0	0	0
0	1	1
1	0	2
1	1	Read-Back Command

D5,D4 - 00 designates Counter Latch Command

X - don't care

NOTE:
Don't care bits (X) should be 0 to insure compatibility with future Intel products.

Figure 9. Counter Latching Command Format

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 CPU (or until the Counter 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 Counter's OL holds its count until it is read. Counter Latch Commands do not affect the programmed Mode of the Counter in any way.

If a Counter is latched and then, some time later, latched again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued.

With either method, the count must be read according to the programmed format; specifically, if the Counter is programmed for two byte counts, two bytes must be read. The two bytes do not have to be read one right after the other; read or write or programming operations of other Counters may be inserted between them.

Another feature of the 82C54 is that reads and writes of the same Counter may be interleaved; for example, if the Counter 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 Counter 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 Counter. Otherwise, an incorrect count will be read.

READ-BACK COMMAND

The third method uses the Read-Back command. This command allows the user to check the count value, programmed Mode, and current state of the OUT pin and Null Count flag of the selected counter(s).

The command is written into the Control Word Register and has the format shown in Figure 10. The command applies to the counters selected by setting their corresponding bits D3,D2,D1 = 1.

A0, A1 = 11 $\overline{CS}$ = 0 $\overline{RD}$ = 1 $\overline{WR}$ = 0

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	$\overline{\text{COUNT}}$	$\overline{\text{STATUS}}$	CNT 2	CNT 1	CNT 0	0

D_5: 0 = Latch count of selected counter(s)
D_4: 0 = Latch status of selected counter(s)
D_3: 1 = Select counter 2
D_2: 1 = Select counter 1
D_1: 1 = Select counter 0
D_0: Reserved for future expansion; must be 0

Figure 10. Read-Back Command Format

The read-back command may be used to latch multiple counter output latches (OL) by setting the $\overline{\text{COUNT}}$ bit D5=0 and selecting the desired counter(s). This single command is functionally equivalent to several counter latch commands, one for each counter latched. Each counter's latched count is held until it is read (or the counter is reprogrammed). That counter is automatically unlatched when read, but other counters remain latched until they are read. If multiple count read-back commands are issued to the same counter without reading the count, all but the first are ignored; i.e., the count which will be read is the count at the time the first read-back command was issued.

The read-back command may also be used to latch status information of selected counter(s) by setting $\overline{\text{STATUS}}$ bit D4 = 0. Status must be latched to be read; status of a counter is accessed by a read from that counter.

The counter status format is shown in Figure 11. Bits D5 through D0 contain the counter's programmed Mode exactly as written in the last Mode Control Word. OUTPUT bit D7 contains the current state of the OUT pin. This allows the user to monitor the counter's output via software, possibly eliminating some hardware from a system.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
OUTPUT	NULL COUNT	RW1	RW0	M2	M1	M0	BCD

D_7 1 = Out Pin is 1
0 = Out Pin is 0
D_6 1 = Null count
0 = Count available for reading
D_5-D_0 Counter Programmed Mode (See Figure 7)

Figure 11. Status Byte

NULL COUNT bit D6 indicates when the last count written to the counter register (CR) has been loaded into the counting element (CE). The exact time this happens depends on the Mode of the counter and is described in the Mode Definitions, but until the count is loaded into the counting element (CE), it can't be read from the counter. If the count is latched or read before this time, the count value will not reflect the new count just written. The operation of Null Count is shown in Figure 12.

THIS ACTION:	CAUSES:
A. Write to the control word register:[1]	Null count = 1
B. Write to the count register (CR);[2]	Null count = 1
C. New count is loaded into CE (CR → CE);	Null count = 0

[1] Only the counter specified by the control word will have its null count set to 1. Null count bits of other counters are unaffected.

[2] If the counter is programmed for two-byte counts (least significant byte then most significant byte) null count goes to 1 when the second byte is written.

Figure 12. Null Count Operation

If multiple status latch operations of the counter(s) are performed without reading the status, all but the first are ignored; i.e., the status that will be read is the status of the counter at the time the first status read-back command was issued.

Both count and status of the selected counter(s) may be latched simultaneously by setting both $\overline{\text{COUNT}}$ and $\overline{\text{STATUS}}$ bits D5,D4 = 0. This is functionally the same as issuing two separate read-back commands at once, and the above discussions apply here also. Specifically, if multiple count and/or status read-back commands are issued to the same counter(s) without any intervening reads, all but the first are ignored. This is illustrated in Figure 13.

If both count and status of a counter are latched, the first read operation of that counter will return latched status, regardless of which was latched first. The next one or two reads (depending on whether the counter is programmed for one or two type counts) return latched count. Subsequent reads return unlatched count.

Command								Description	Results
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
1	1	0	0	0	0	1	0	Read back count and status of Counter 0	Count and status latched for Counter 0
1	1	1	0	0	1	0	0	Read back status of Counter 1	Status latched for Counter 1
1	1	1	0	1	1	0	0	Read back status of Counters 2, 1	Status latched for Counter 2, but not Counter 1
1	1	0	1	1	0	0	0	Read back count of Counter 2	Count latched for Counter 2
1	1	0	0	0	1	0	0	Read back count and status of Counter 1	Count latched for Counter 1, but not status
1	1	1	0	0	0	1	0	Read back status of Counter 1	Command ignored, status already latched for Counter 1

Figure 13. Read-Back Command Example

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	
0	1	0	0	0	Write into Counter 0
0	1	0	0	1	Write into Counter 1
0	1	0	1	0	Write into Counter 2
0	1	0	1	1	Write Control Word
0	0	1	0	0	Read from Counter 0
0	0	1	0	1	Read from Counter 1
0	0	1	1	0	Read from Counter 2
0	0	1	1	1	No-Operation (3-State)
1	X	X	X	X	No-Operation (3-State)
0	1	1	X	X	No-Operation (3-State)

Figure 14. Read/Write Operations Summary

Mode Definitions

The following are defined for use in describing the operation of the 82C54.

CLK PULSE: a rising edge, then a falling edge, in that order, of a Counter's CLK input.

TRIGGER: a rising edge of a Counter's GATE input.

COUNTER LOADING: the transfer of a count from the CR to the CE (refer to the "Functional Description")

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.

GATE = 1 enables counting; GATE = 0 disables counting. GATE has no effect on OUT.

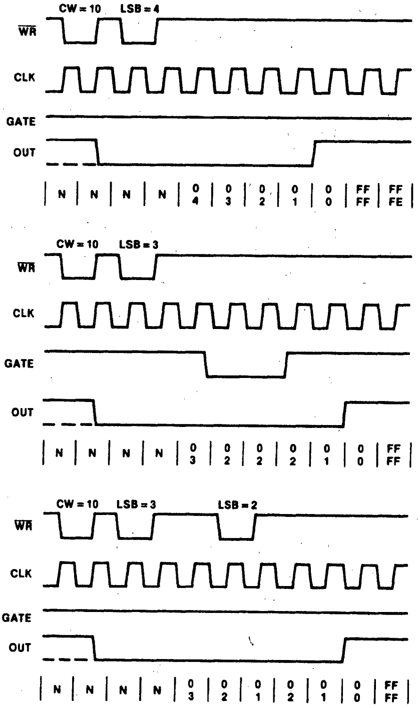
After the Control Word and initial count are written to a Counter, the initial count will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not go high until N + 1 CLK pulses after the initial count is written.

If a new count is written to the Counter, 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 (no clock 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 = 0, it will still 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.


CW=10 LSB=4
WR
CLK
GATE
OUT
N N N N 0 4 0 3 0 2 0 1 0 0 FF FF FF FE
CW=10 LSB=3
WR
CLK
GATE
OUT
N N N N 0 3 0 2 0 2 0 2 0 1 0 0 FF FF
CW=10 LSB=3 LSB=2
WR
CLK
GATE
OUT
N N N N 0 3 0 2 0 1 0 2 0 1 0 0 FF FF
231244-8

NOTE:
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 15. Mode 0

MODE 1: HARDWARE RETRIGGERABLE ONE-SHOT

OUT will be initially high. OUT will go low on the CLK pulse following a trigger to begin the one-shot pulse, and will remain low until the Counter reaches zero. OUT will then go high and remain high until the CLK pulse after the next trigger.

After writing the Control Word and initial count, the Counter is armed. A trigger results in loading the Counter and setting OUT low on the next CLK pulse, thus starting the one-shot pulse. An initial count of N will result in a one-shot pulse N CLK cycles in duration. The one-shot is retriggerable, hence OUT will remain low for N CLK pulses after any trigger. The one-shot pulse can be repeated without rewriting the same count into the counter. GATE has no effect on OUT.

If a new count is written to the Counter during a one-shot pulse, the current one-shot is not affected unless the Counter is retriggered. In that case, the Counter is loaded with the new count and the one-shot pulse continues until the new count expires.

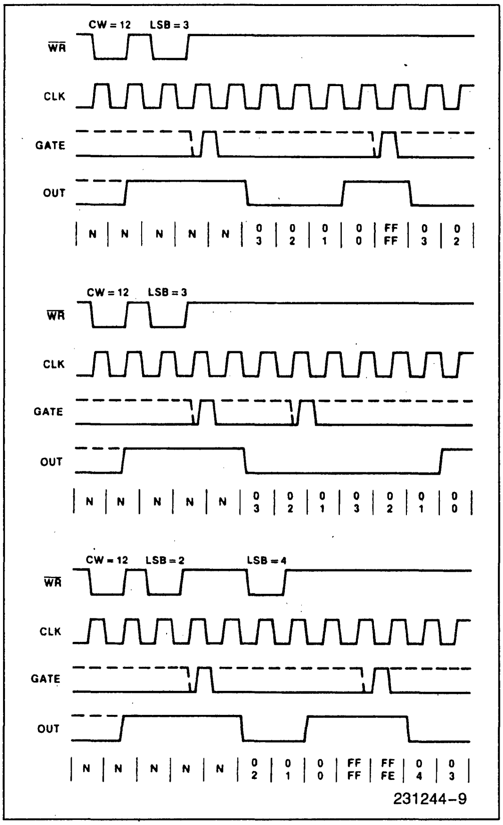

CW = 12 LSB = 3
WR
CLK
GATE
OUT
N N N N N 0 3 0 2 0 1 0 0 FF FF 0 3 0 2
CW = 12 LSB = 3
WR
CLK
GATE
OUT
N N N N N 0 3 0 2 0 1 0 3 0 2 0 1 0 0
CW = 12 LSB = 2 LSB = 4
WR
CLK
GATE
OUT
N N N N N 0 2 0 1 0 0 FF FF FF FE 0 4 0 3
231244-9

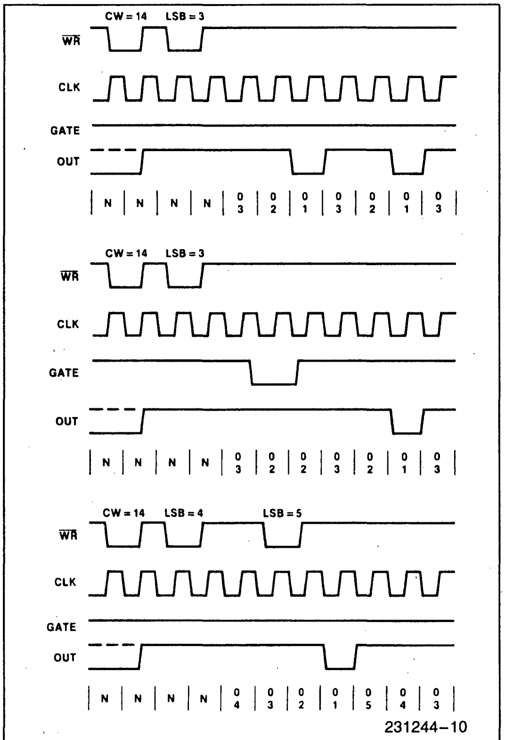
Figure 16. Mode 1

MODE 2: RATE GENERATOR

This Mode functions like a divide-by-N counter. It is typicially 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. OUT then goes high again, the Counter reloads the initial count and the process is repeated. Mode 2 is periodic; the same sequence is repeated indefinitely. For an initial count of N, the sequence repeats every N CLK cycles.

GATE = 1 enables counting; GATE = 0 disables counting. If GATE goes low during an output pulse, OUT is set high immediately. A trigger reloads the Counter with the initial count on the next CLK pulse; OUT goes low N CLK pulses after the trigger. Thus the GATE input can be used to synchronize the Counter.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. OUT goes low N CLK Pulses after the initial count is written. This allows the Counter to be synchronized by software also.


CW = 14 LSB = 3
WR
CLK
GATE
OUT
N N N N 0 3 0 2 0 1 0 3 0 2 0 1 0 3
CW = 14 LSB = 3
WR
CLK
GATE
OUT
N N N N 0 3 0 2 0 2 0 3 0 2 0 1 0 3
CW = 14 LSB = 4 LSB = 5
WR
CLK
GATE
OUT
N N N N 0 4 0 3 0 2 0 1 0 5 0 4 0 3
231244-10

NOTE:
A GATE transition should not occur one clock prior to terminal count.

Figure 17. Mode 2

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 period, the Counter 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 counting cycle. In mode 2, a COUNT of 1 is illegal.

MODE 3: SQUARE WAVE MODE

Mode 3 is typically used for Baud rate generation. Mode 3 is similar to Mode 2 except for the duty cycle of OUT. OUT will initially be high. When half the initial count has expired, OUT goes low for the remainder of the count. Mode 3 is periodic; the sequence above is repeated indefinitely. An initial count of N results in a square wave with a period of N CLK cycles.

GATE = 1 enables counting; GATE = 0 disables counting. If GATE goes low while OUT is low, OUT is set high immediately; no CLK pulse is required. A trigger reloads the Counter with the initial count on the next CLK pulse. Thus the GATE input can be used to synchronize the Counter.

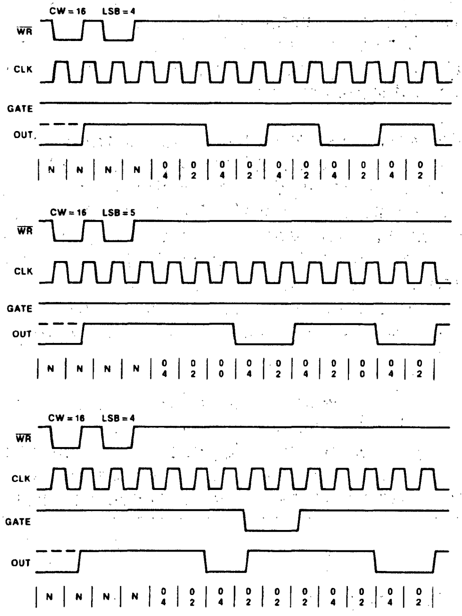
After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This allows the Counter to be synchronized by software also.

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 Counter 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.

Mode 3 is implemented as follows:

Even counts: OUT is initially high. The initial count is loaded on one CLK pulse and then is decremented by two on succeeding CLK pulses. When the count expires OUT changes value and the Counter is reloaded with the initial count. The above process is repeated indefinitely.

Odd counts: OUT is initially high. The initial count minus one (an even number) is loaded on one CLK pulse and then is decremented by two on succeeding CLK pulses. One CLK pulse *after* the count expires, OUT goes low and the Counter is reloaded with the initial count minus one. Succeeding CLK pulses decrement the count by two. When the count expires, OUT goes high again and the Counter 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.


CW = 16 LSB = 4
WR
CLK
GATE
OUT
N N N N 0 4 0 2 0 4 0 2 0 4 0 2 0 4 0 2 0 4 0 2
CW = 16 LSB = 5
WR
CLK
GATE
OUT
N N N N 0 4 0 2 0 0 0 4 0 2 0 4 0 2 0 0 0 4 0 2
CW = 16 LSB = 4
WR
CLK
GATE
OUT
N N N N 0 4 0 2 0 4 0 2 0 2 0 2 0 4 0 2 0 4 0 2

231244–11

NOTE:
A GATE transition should not occur one clock prior to terminal count.

Figure 18. Mode 3

MODE 4: SOFTWARE TRIGGERED STROBE

OUT will be initially high. When the initial count expires, OUT will go low for one CLK pulse and then go high again. The counting sequence is "triggered" by writing the initial count.

GATE = 1 enables counting; GATE = 0 disables counting. GATE has no effect on OUT.

After writing a Control Word and initial count, the Counter 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 the initial count is written.

If a new count is written during counting, 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 has no effect on counting.

2) Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the sequence to be "retriggered" by software. OUT strobes low N+1 CLK pulses after the new count of N is written.

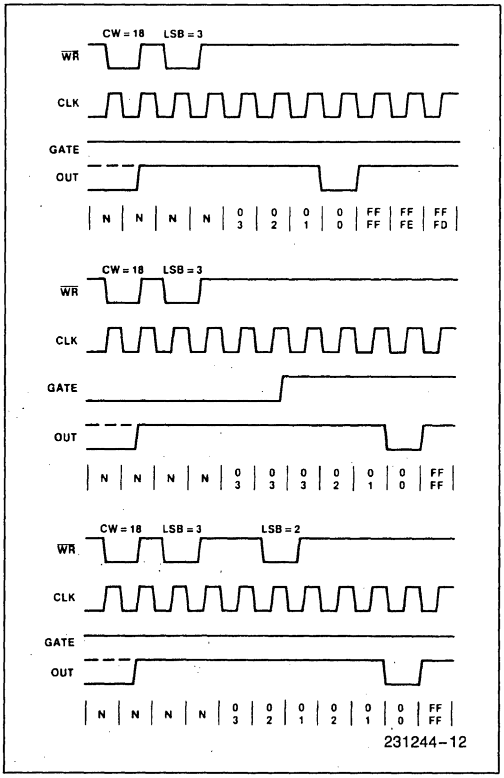

CW = 18
LSB = 3
WR
CLK
GATE
OUT
N N N N 0 3 0 2 0 1 0 0 FF FF FF FE FF FD
CW = 18
LSB = 3
WR
CLK
GATE
OUT
N N N N 0 3 0 3 0 3 0 2 0 1 0 0 FF FF
CW = 18
LSB = 3
LSB = 2
WR
CLK
GATE
OUT
N N N N 0 3 0 2 0 1 0 2 0 1 0 0 FF FF
231244-12

Figure 19. Mode 4

MODE 5: HARDWARE TRIGGERED STROBE (RETRIGGERABLE)

OUT will initially be high. Counting is triggered by a rising edge of GATE. When the initial count has expired, OUT will go low for one CLK pulse and then go high again.

After writing the Control Word and initial count, the counter will not be loaded until the CLK pulse after a trigger. 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 a trigger.

A trigger results in the Counter being loaded with the initial count on the next CLK pulse. The counting sequence is retriggerable. OUT will not strobe low for N + 1 CLK pulses after any trigger. GATE has no effect on OUT.

If a 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 Counter will be loaded with the new count on the next CLK pulse and counting will continue from there.

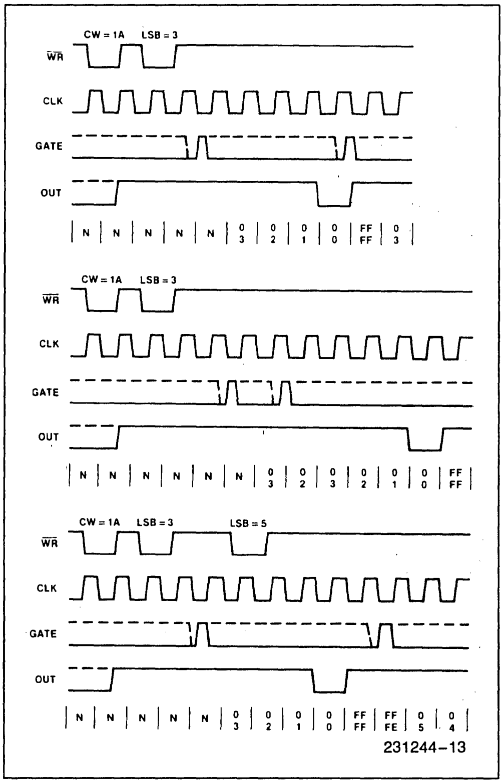

CW = 1A
LSB = 3
WR
CLK
GATE
OUT
N N N N N 0 3 0 2 0 1 0 0 FF FF 0 3
CW = 1A
LSB = 3
WR
CLK
GATE
OUT
N N N N N N 0 3 0 2 0 3 0 2 0 1 0 0 FF FF
CW = 1A
LSB = 3
LSB = 5
WR
CLK
GATE
OUT
N N N N N 0 3 0 2 0 1 0 0 FF FF FF FE 0 5 0 4
231244-13

Figure 20. Mode 5

Signal Status Modes	Low Or Going Low	Rising	High
0	Disables counting	—	Enables counting
1	—	1) Initiates counting 2) Resets output after next clock	—
2	1) Disables counting 2) Sets output immediately high	Initiates counting	Enables counting
3	1) Disables counting 2) Sets output immediately high	Initiates counting	Enables counting
4	Disables counting	—	Enables counting
5	—	Initiates counting	—

Figure 21. Gate Pin Operations Summary

MODE	MIN COUNT	MAX COUNT
0	1	0
1	1	0
2	2	0
3	2	0
4	1	0

NOTE:
0 is equivalent to 2^{16} for binary counting and 10^4 for BCD counting

Figure 22. Minimum and Maximum initial Counts

Operation Common to All Modes

Programming

When a Control Word is written to a Counter, all Control Logic is immediately reset and OUT goes to a known initial state; no CLK pulses are required for this.

GATE

The GATE input is always sampled on the rising edge of CLK. In Modes 0, 2, 3, and 4 the GATE input is level sensitive, and the logic level is sampled on the rising edge of CLK. 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 Counter. This flip-flop is then sampled on the next rising edge of CLK; the flip-flop is reset immediately after it is sampled. In this way, a trigger will be detected no matter when it occurs—a high logic level does not have to be maintained until the next rising edge of CLK. Note that in Modes 2 and 3, the GATE input is both edge- and level-sensitive. In Modes 2 and 3, if a CLK source other than the system clock is used, GATE should be pulsed immediately following $\overline{WR}$ of a new count value.

COUNTER

New counts are loaded and Counters are decremented on the falling edge of CLK.

The largest possible initial count is 0; this is equivalent to 2^{16} for binary counting and 10^4 for BCD counting.

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.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias	0°C to 70°C
Storage Temperature	−65° to +150°C
Supply Voltage	−0.5 to +8.0V
Operating Voltage	+4V to +7V
Voltage on any Input	GND −2V to +6.5V
Voltage on any Output	GND−0.5V to V_{CC} + 0.5V
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_A=0°C to 70°C, V_{CC}=5V± 10%, GND=0V) (T_A = −40°C to +85°C for Extended Temperature)

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		0.4	V	I_{OL} = 2.5 mA
V_{OH}	Output High Voltage	3.0 V_{CC} − 0.4		V V	I_{OH} = −2.5 mA I_{OH} = −100 μA
I_{IL}	Input Load Current		±2.0	μA	V_{IN}=V_{CC} to 0V
I_{OFL}	Output Float Leakage Current		±10	μA	V_{OUT}=V_{CC} to 0.0V
I_{CC}	V_{CC} Supply Current		20	mA	Clk Freq= 8MHz 82C54 10MHz 82C54-2
I_{CCSB}	V_{CC} Supply Current-Standby		10	μA	CLK Freq = DC $\overline{\text{CS}}$ = HIGH. All Inputs/Data Bus HIGH All Outputs Floating
I_{CCSB1}	V_{CC} Supply Current-Standby		150	μA	CLK Freq = DC $\overline{\text{CS}}$ = HIGH. All Other Inputs, I/O Pins = Low, Outputs Open
C_{IN}	Input Capacitance		10	pF	f_c = 1 MHz
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to GND(5)
C_{OUT}	Output Capacitance		20	pF	

A.C. CHARACTERISTICS

(T_A = 0°C to 70°C, V_{CC} = 5V ±10%, GND =0V) (T_A = −40°C to +85°C for Extended Temperature)

BUS PARAMETERS (Note 1)

READ CYCLE

Symbol	Parameter	82C54		82C54-2		Units
		Min	Max	Min	Max	
t_{AR}	Address Stable Before $\overline{\text{RD}}$ ↓	45		30		ns
t_{SR}	$\overline{\text{CS}}$ Stable Before $\overline{\text{RD}}$ ↓	0		0		ns
t_{RA}	Address Hold Time After $\overline{\text{RD}}$ ↑	0		0		ns
t_{RR}	$\overline{\text{RD}}$ Pulse Width	150		95		ns
t_{RD}	Data Delay from $\overline{\text{RD}}$ ↓		120		85	ns
t_{AD}	Data Delay from Address		220		185	ns
t_{DF}	$\overline{\text{RD}}$ ↑ to Data Floating	5	90	5	65	ns
t_{RV}	Command Recovery Time	200		165		ns

NOTE:

1. AC timings measured at V_{OH} = 2.0V, V_{OL} = 0.8V.

A.C. CHARACTERISTICS (Continued)

WRITE CYCLE

Symbol	Parameter	82C54		82C54-2		Units
		Min	Max	Min	Max	
t_{AW}	Address Stable Before $\overline{WR}$ ↓	0		0		ns
t_{SW}	$\overline{CS}$ Stable Before $\overline{WR}$ ↓	0		0		ns
t_{WA}	Address Hold Time After $\overline{WR}$ ↑	0		0		ns
t_{WW}	$\overline{WR}$ Pulse Width	150		95		ns
t_{DW}	Data Setup Time Before $\overline{WR}$ ↑	120		95		ns
t_{WD}	Data Hold Time After $\overline{WR}$ ↑	0		0		ns
t_{RV}	Command Recovery Time	200		165		ns

CLOCK AND GATE

Symbol	Parameter	82C54		82C54-2		Units
		Min	Max	Min	Max	
t_{CLK}	Clock Period	125	DC	100	DC	ns
t_{PWH}	High Pulse Width	60(3)		30(3)		ns
t_{PWL}	Low Pulse Width	60(3)		50(3)		ns
T_R	Clock Rise Time		25		25	ns
t_F	Clock Fall Time		25		25	ns
t_{GW}	Gate Width High	50		50		ns
t_{GL}	Gate Width Low	50		50		ns
t_{GS}	Gate Setup Time to CLK ↑	50		40		ns
t_{GH}	Gate Hold Time After CLK ↑	50(2)		50(2)		ns
T_{OD}	Output Delay from CLK ↓		150		100	ns
t_{ODG}	Output Delay from Gate ↓		120		100	ns
t_{WC}	CLK Delay for Loading(4)	0	55	0	55	ns
t_{WG}	Gate Delay for Sampling(4)	−5	50	−5	40	ns
t_{WO}	OUT Delay from Mode Write		260		240	ns
t_{CL}	CLK Set Up for Count Latch	−40	45	−40	40	ns

NOTES:

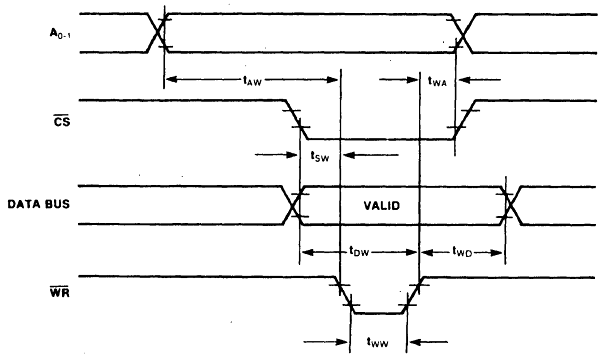
2. In Modes 1 and 5 triggers are sampled on each rising clock edge. A second trigger within 120 ns (70 ns for the 82C54-2) of the rising clock edge may not be detected.
3. Low-going glitches that violate t_{PWH}, t_{PWL} may cause errors requiring counter reprogramming.
4. Except for Extended Temp., See Extended Temp. A.C. Characteristics below.
5. Sampled not 100% tested. $T_A = 25°C$.
6. If CLK present at T_{WC} min then Count equals N+2 CLK pulses, T_{WC} max equals Count N+1 CLK pulse. T_{WC} min to T_{WC} max, count will be either N+1 or N+2 CLK pulses.
7. In Modes 1 and 5, if GATE is present when writing a new Count value, at T_{WG} min Counter will not be triggered, at T_{WG} max Counter will be triggered.
8. If CLK present when writing a Counter Latch or ReadBack Command, at T_{CL} min CLK will be reflected in count value latched, at T_{CL} max CLK will not be reflected in the count value latched. Writing a Counter Latch or ReadBack Command between T_{CL} min and T_{WL} max will result in a latched count vallue which is ± one least significant bit.

EXTENDED TEMPERATURE ($T_A = -40°C$ to $+85°C$ for Extended Temperature)

Symbol	Parameter	82C54		82C54-2		Units
		Min	Max	Min	Max	
t_{WC}	CLK Delay for Loading	−25	25	−25	25	ns
t_{WG}	Gate Delay for Sampling	−25	25	−25	25	ns

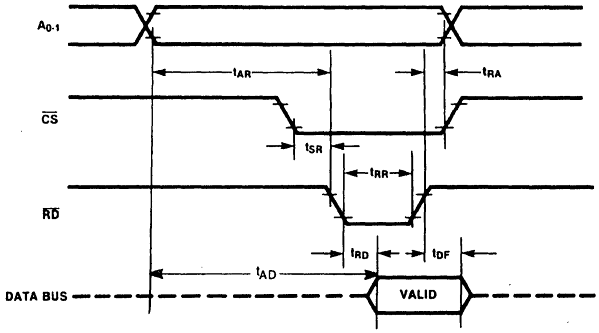
WAVEFORMS

WRITE



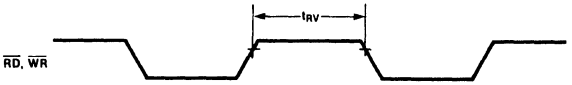
231244–14

READ



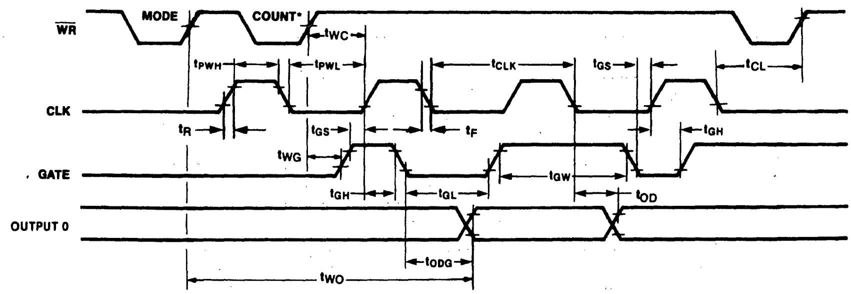
231244–15

RECOVERY



231244–16

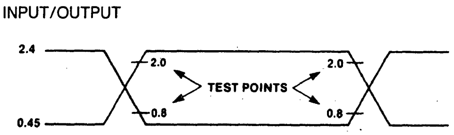
CLOCK AND GATE



231244-17

* Last byte of count being written

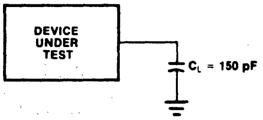
A.C. TESTING INPUT, OUTPUT WAVEFORM



231244-18

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



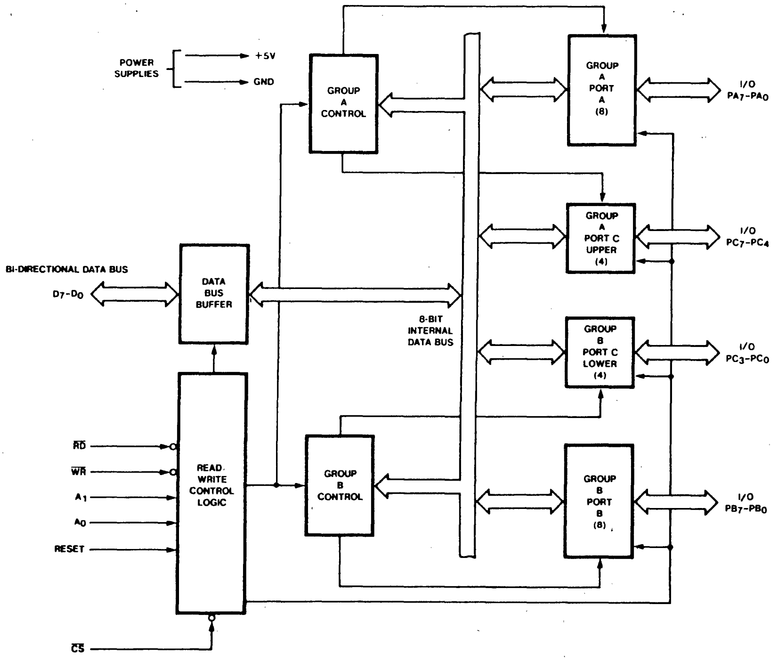
231244-19

C_L = 150 pF
C_L includes jig capacitance

8255A/8255A-5
PROGRAMMABLE PERIPHERAL INTERFACE

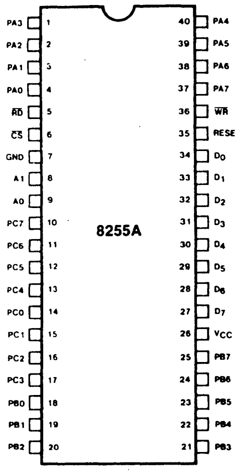
- **MCS-85™ Compatible 8255A-5**
- **24 Programmable I/O Pins**
- **Completely TTL Compatible**
- **Fully Compatible with Intel Microprocessor Families**
- **Improved Timing Characteristics**
- **Direct Bit Set/Reset Capability Easing Control Application Interface**
- **Reduces System Package Count**
- **Improved DC Driving Capability**
- **Available in EXPRESS**
  — Standard Temperature Range
  — Extended Temperature Range
- **40 Pin DIP Package or 44 Lead PLCC**
  (See Intel Packaging: Order Number: 231369)

The Intel 8255A is a general purpose programmable I/O device designed for use with Intel microprocessors. It has 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. In the first mode (MODE 0), each group of 12 I/O pins may be programmed in sets of 4 to be input or output. In MODE 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining 4 pins, 3 are used for handshaking and interrupt control signals. The third mode of operation (MODE 2) is a bidirectional bus mode which uses 8 lines for a bidirectional bus, and 5 lines, borrowing one from the other group, for handshaking.



231308-1

Figure 1. 8255A Block Diagram



231308-2

Figure 2. Pin Configuration

8255A FUNCTIONAL DESCRIPTION

General

The 8255A is a programmable peripheral interface (PPI) device designed for use in Intel microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 8255A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 8255A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

$(\overline{\text{CS}})$

Chip Select. A "low" on this input pin enables the communication between the 8255A and the CPU.

$(\overline{\text{RD}})$

Read. A "low" on this input pin enables the 8255A to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to "read from" the 8255A.

$(\overline{\text{WR}})$

Write. A "low" on this input pin enables the CPU to write data or control words into the 8255A.

(A_0 and A_1)

Port Select 0 and Port Select 1. These input signals, in conjunction with the RD and WR inputs, control the selection of one of the three ports or the control word registers. They are normally connected to the least significant bits of the address bus (A_0 and A_1).

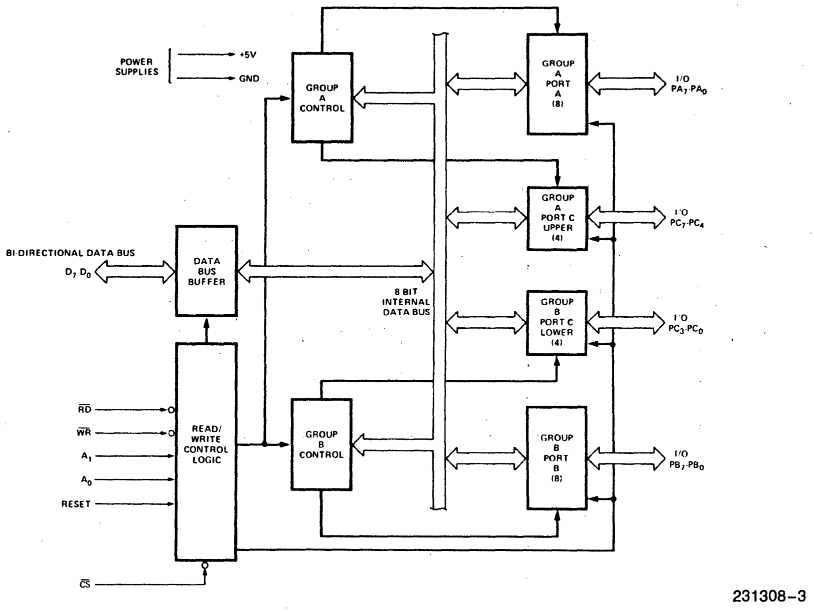


Figure 3. 8255A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions

8255A BASIC OPERATION

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	**Input Operation (READ)**
0	0	0	1	0	Port A → Data Bus
0	1	0	1	0	Port B → Data Bus
1	0	0	1	0	Port C → Data Bus
					Output Operation (WRITE)
0	0	1	0	0	Data Bus → Port A
0	1	1	0	0	Data Bus → Port B
1	0	1	0	0	Data Bus → Port C
1	1	1	0	0	Data Bus → Control
					Disable Function
X	X	X	X	1	Data Bus → 3-State
1	1	0	1	0	Illegal Condition
X	X	1	1	0	Data Bus → 3-State

(RESET)

Reset. A "high" on this input clears the control register and all ports (A, B, C) are set to the input mode.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 8255A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 8255A.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A—Port A and Port C upper (C7-C4)

Control Group B—Port B and Port C lower (C3-C0)

The Control Word Register can **Only** be written into. No Read operation of the Control Word Register is allowed.

Ports A, B, and C

The 8255A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 8255A.

Port A. One 8-bit data output latch/buffer and one 8-bit data input latch.

Port B. One 8-bit data input/output latch/buffer and one 8-bit data input buffer.

Port C. One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.

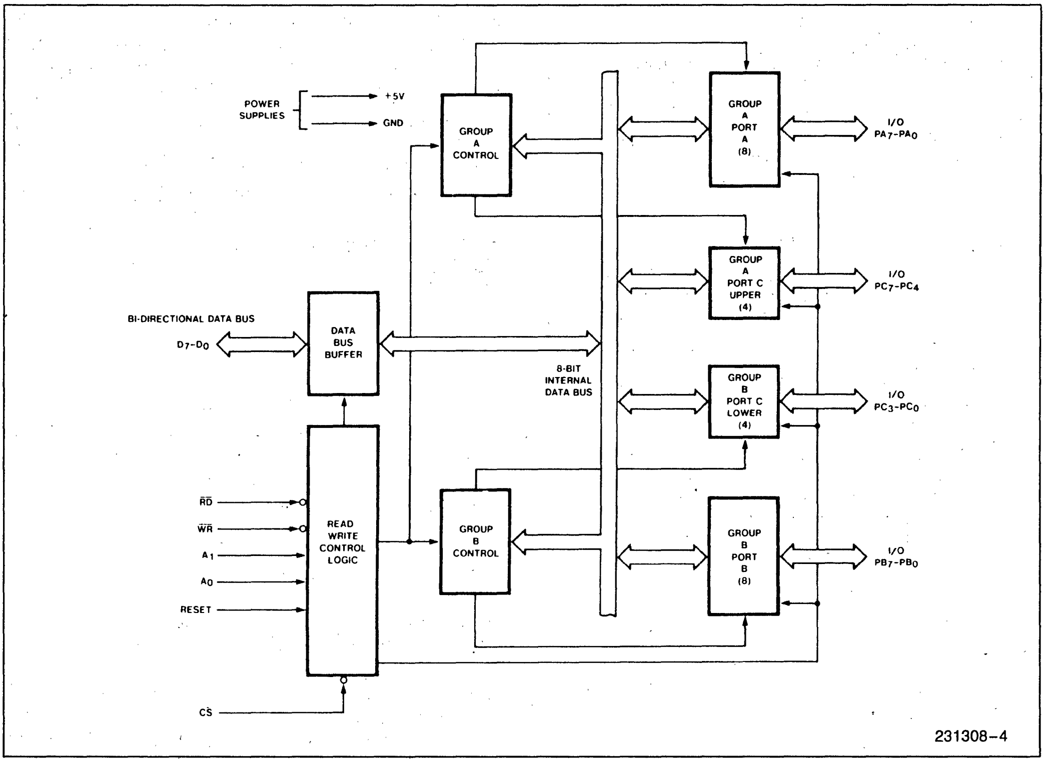
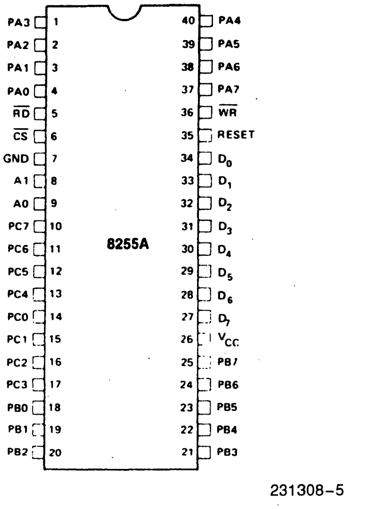


Figure 4. 8225A Block Diagram Showing Group A and Group B Control Functions

Pin Configuration



Pin Names

D_7–D_0	Data Bus (Bi-Directional)
RESET	Reset Input
$\overline{CS}$	Chip Select
$\overline{RD}$	Read Input
$\overline{WR}$	Write Input
A0, A1	Port Address
PA7–PA0	Port A (BIT)
PB7–PB0	Port B (BIT)
PC7–PC0	Port C (BIT)
V_{CC}	+5 Volts
GND	0 Volts

8255A OPERATIONAL DESCRIPTION

Mode Selection

There are three basic modes of operation that can be selected by the system software:

Mode 0—Basic Input/Output

Mode 1—Strobed Input/Output

Mode 2—Bi-Directional Bus

When the reset input goes "high" all ports will be set to the input mode (i.e., all 24 lines will be in the high impedance state). After the reset is removed the 8255A can remain in the input mode with no additional initialization required. During the execution of the system program any of the other modes may be selected using a single output instruction. This allows a single 8255A to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

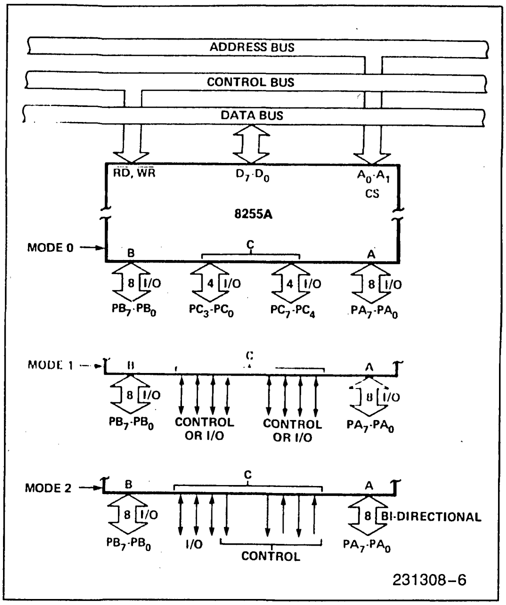


Figure 5. Basic Mode Definitions and Bus Interface

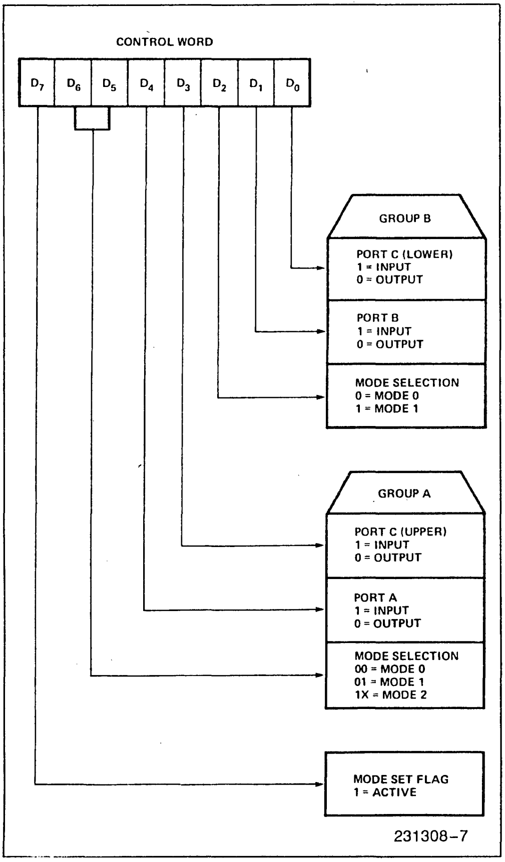


Figure 6. Mode Definition Format

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 8255A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.

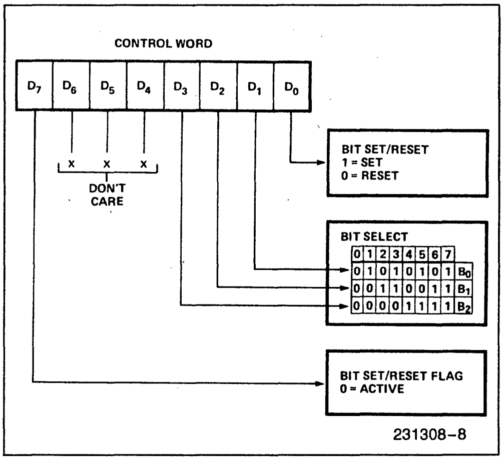


Figure 7. Bit Set/Reset Format

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

Interrupt Control Functions

When the 8255A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET)—INTE is set—Interrupt enable

(BIT-RESET)—INTE is RESET—Interrupt disable

NOTE:
All Mask flip-flops are automatically reset during mode selection and device Reset.

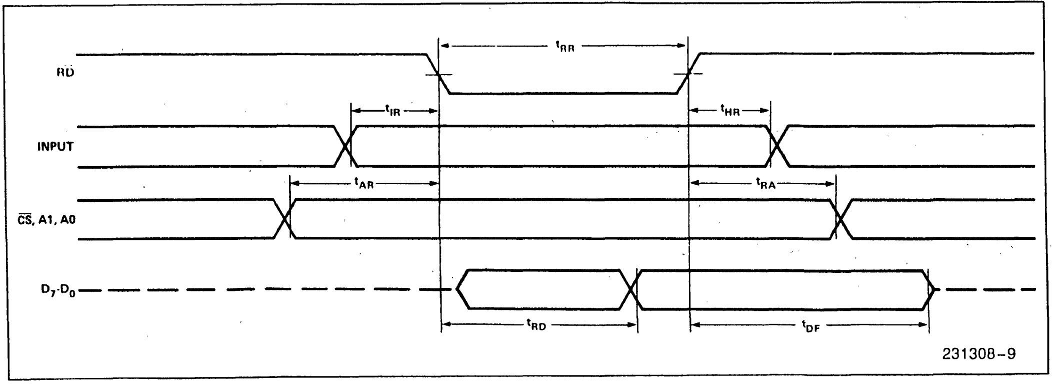
Operating Modes

MODE 0 (Basic Input/Output). This functional configuration provides simple input and output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

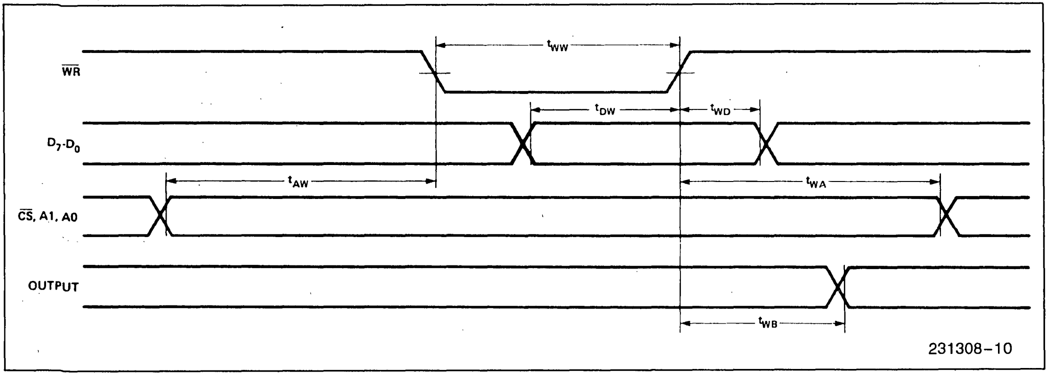
Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.

MODE 0 (BASIC INPUT)



MODE 0 (BASIC OUTPUT)



231308-10

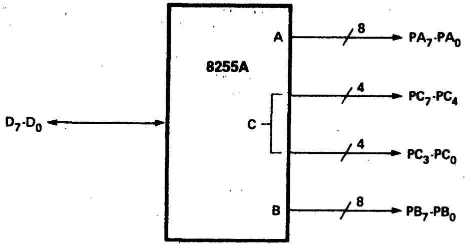
MODE 0 PORT DEFINITION

A		B		Group A			Group B	
D_4	D_3	D_1	D_0	Port A	Port C (Upper)	#	Port B	Port C (Lower)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

MODE CONFIGURATIONS

CONTROL WORD #0

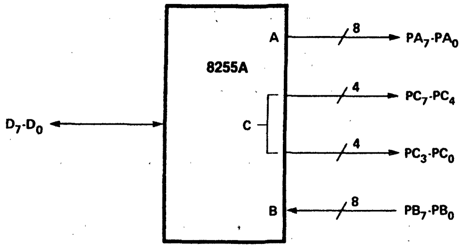
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	0	0	0	0



231308-11

CONTROL WORD #2

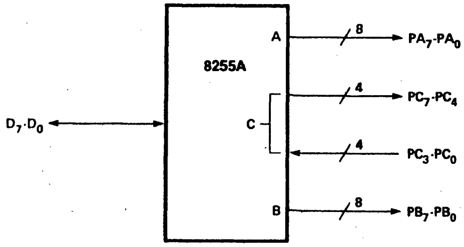
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	0	0	1	0



231308-12

CONTROL WORD #1

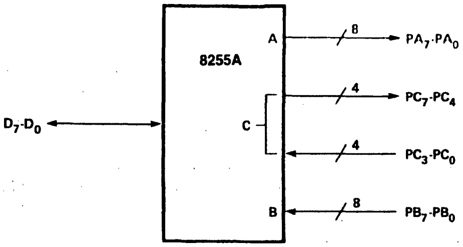
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	0	0	0	1



231308-13

CONTROL WORD #3

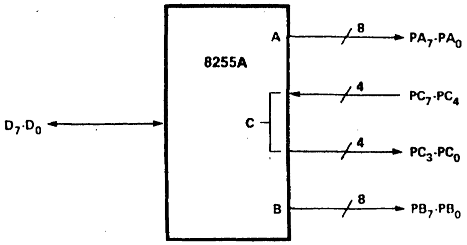
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	0	0	1	1



231308-14

CONTROL WORD #4

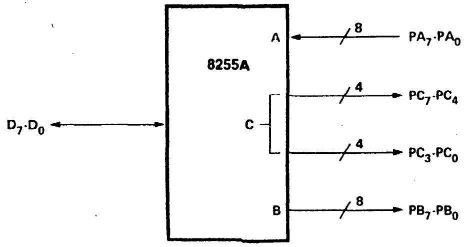
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	0



231308-15

CONTROL WORD #8

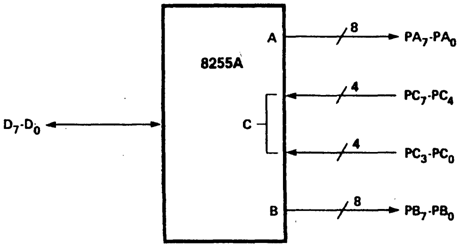
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	0



231308-16

CONTROL WORD #5

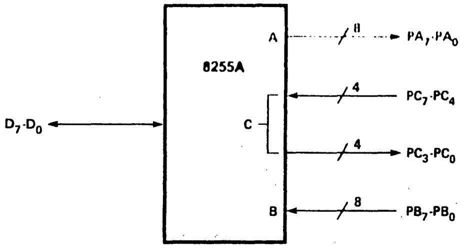
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	1



231308–17

CONTROL WORD #6

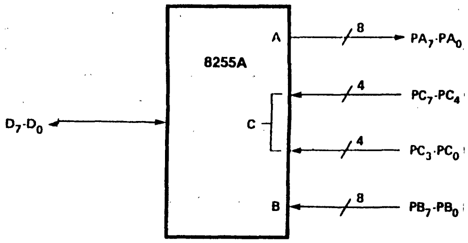
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	0



231308–19

CONTROL WORD #7

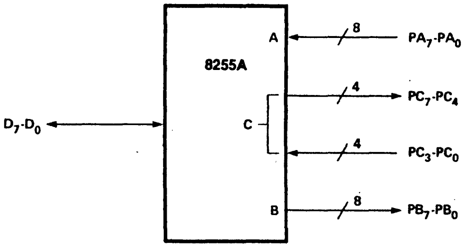
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	1



231308–21

CONTROL WORD #9

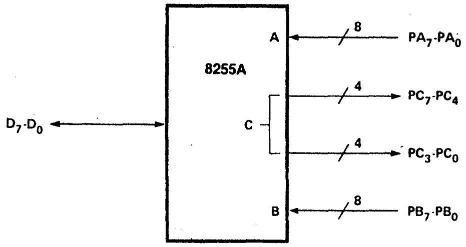
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	1



231308–18

CONTROL WORD #10

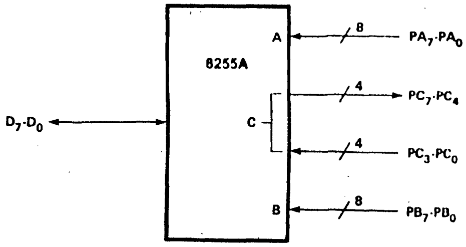
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	0



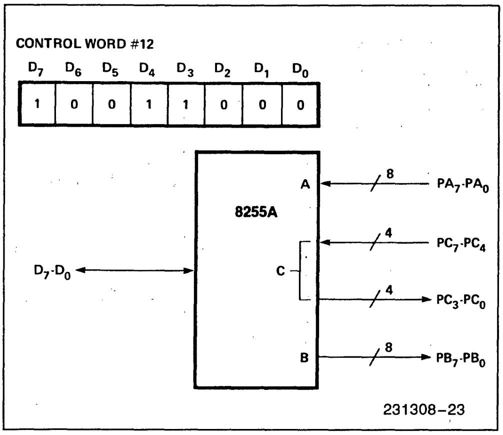
231308–20

CONTROL WORD #11

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	1



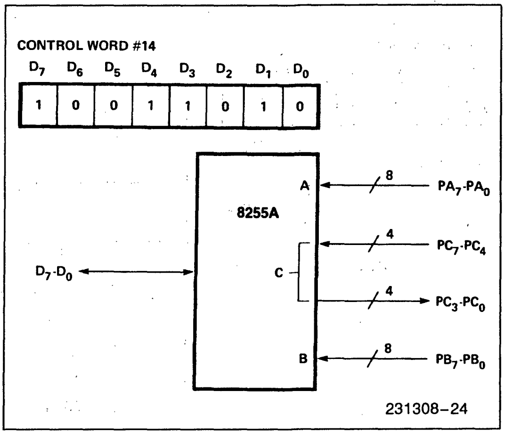
231308–22



CONTROL WORD #12

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	1	0	0	0

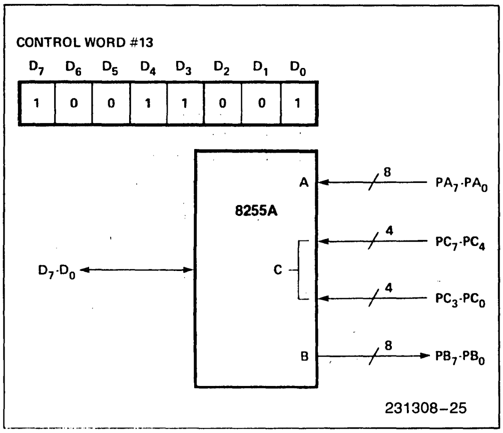
231308–23



CONTROL WORD #14

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	1	0	1	0

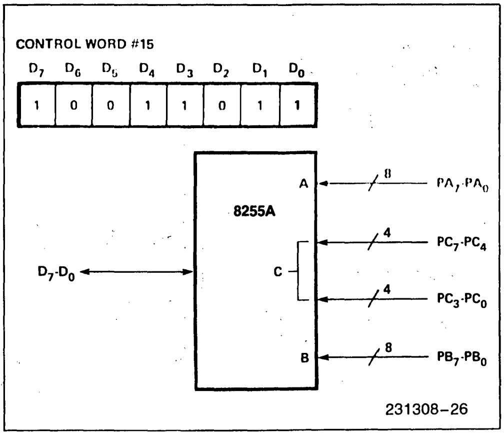
231308–24



CONTROL WORD #13

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	1	0	0	1

231308–25



CONTROL WORD #15

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	1	0	1	1

231308–26

Operating Modes

MODE 1 (Strobed Input/Output). This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In mode 1, port A and port B use the lines on port C to generate or accept these "handshaking" signals.

Mode 1 Basic Functional Definitions:

- Two Groups (Group A and Group B)
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output. Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

Input Control Signal Definition

$\overline{\text{STB}}$ (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by STB input being low and is reset by the rising edge of the RD input.

INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the $\overline{\text{STB}}$ is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{\text{RD}}$. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

INTE A

Controlled by bit set/reset of PC_4.

INTE B

Controlled by bit set/reset of PC_2.

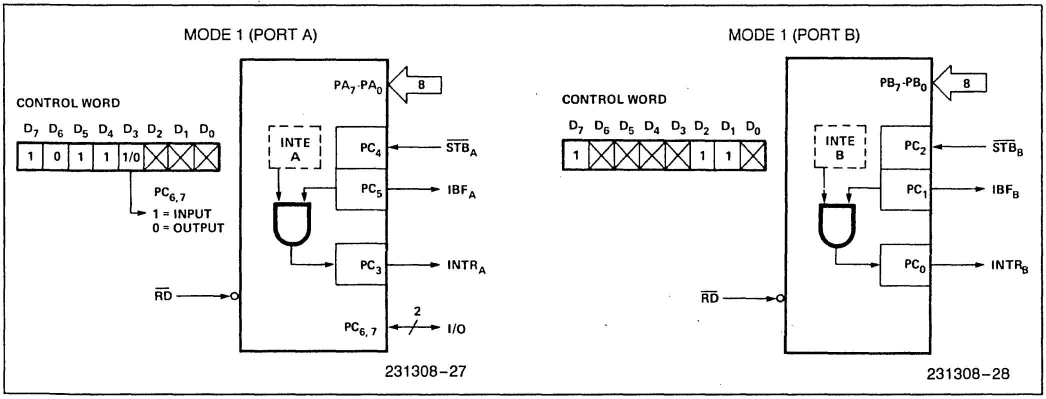


Figure 8. MODE 1 Input

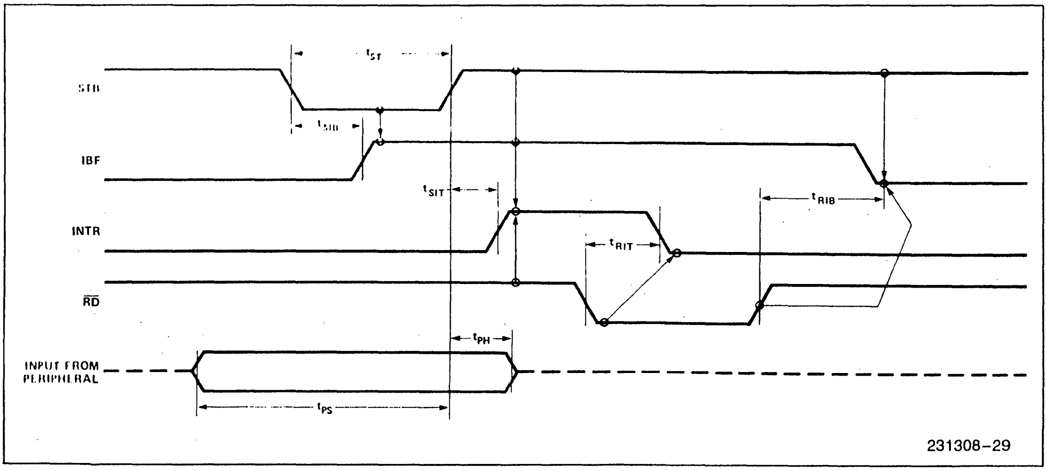


Figure 9. MODE 1 (Strobed Input)

Output Control Signal Definition

$\overline{\text{OBF}}$ (Output Buffer Full F/F). The $\overline{\text{OBF}}$ output will go "low" to indicate that the CPU has written data out to the specified port. The $\overline{\text{OBF}}$ F/F will be set by the rising edge of the $\overline{\text{WR}}$ input and reset by $\overline{\text{ACK}}$ input being low.

$\overline{\text{ACK}}$ (Acknowledge Input). A "low" on this input informs the 8255A that the data from port A or port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

INTR (Interrupt Request). A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set when $\overline{\text{ACK}}$ is a "one", $\overline{\text{OBF}}$ is a "one", and INTE is a "one". It is reset by the falling edge of $\overline{\text{WR}}$.

INTE A

Controlled by bit set/reset of PC_6.

INTE B

Controlled by bit set/reset of PC_2.

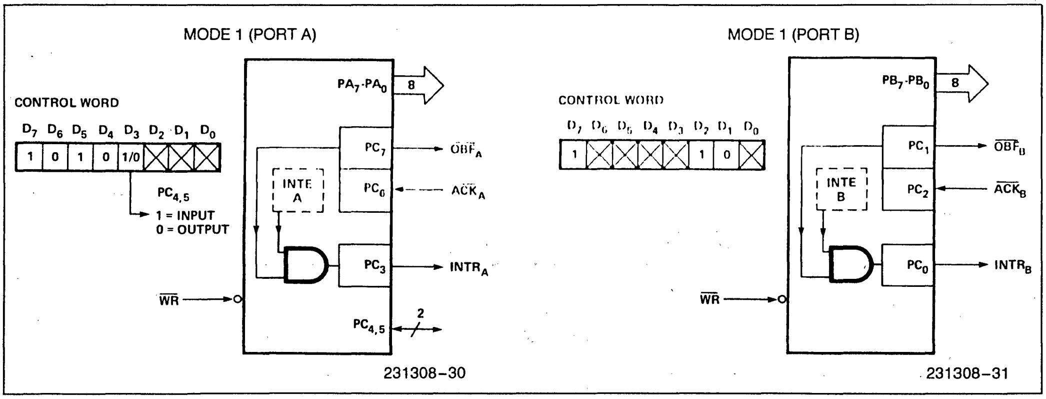


Figure 10. MODE 1 Output

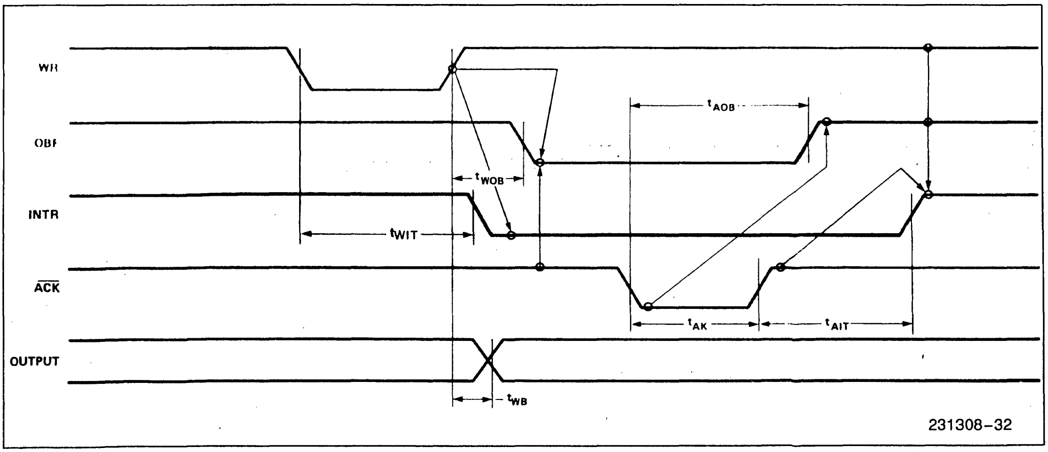
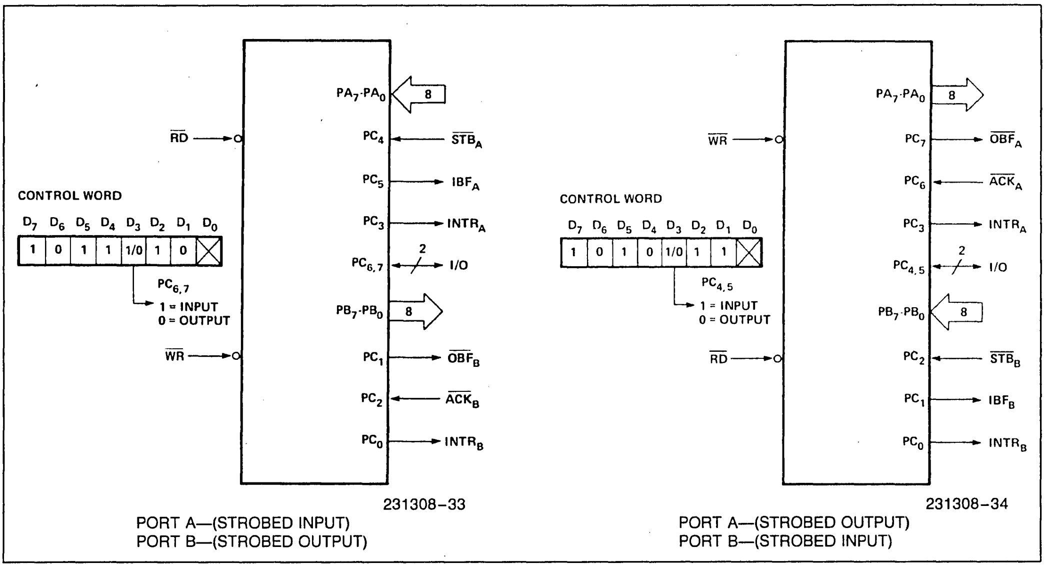


Figure 11. MODE 1 (Strobed Output)



231308-33

PORT A—(STROBED INPUT)
PORT B—(STROBED OUTPUT)

231308-34

PORT A—(STROBED OUTPUT)
PORT B—(STROBED INPUT)

Figure 12. Combinations of MODE 1

Combinations of MODE 1

Port A and Port B can be individually defined as input or output in MODE 1 to support a wide variety of strobed I/O applications.

Operating Modes

MODE 2 (Strobed Bidirectional Bus I/O). This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

MODE 2 Basic Functional Definitions:

- Used in Group A **only**.
- One 8-bit, bi-directional bus Port (Port A) and a 5-bit control Port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

Bidirectional Bus I/O Control Signal Definition

INTR (Interrupt Request). A high on this output can be used to interrupt the CPU for both input or output operations.

Output Operations

$\overline{\text{OBF}}$ (Output Buffer Full). The $\overline{\text{OBF}}$ output will go "low" to indicate that the CPU has written data out to port A.

$\overline{\text{ACK}}$ (Acknowledge). A "low" on this input enables the tri-state output buffer of port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

INTE 1 (The INTE Flip-Flop Associated with OBF). Controlled by bit set/reset of PC_6.

Input Operations

$\overline{\text{STB}}$ (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F). A "high" on this output indicates that data has been loaded into the input latch.

INTE 2 (The INTE Flip-Flop Associated with IBF). Controlled by bit set/reset of PC_4.

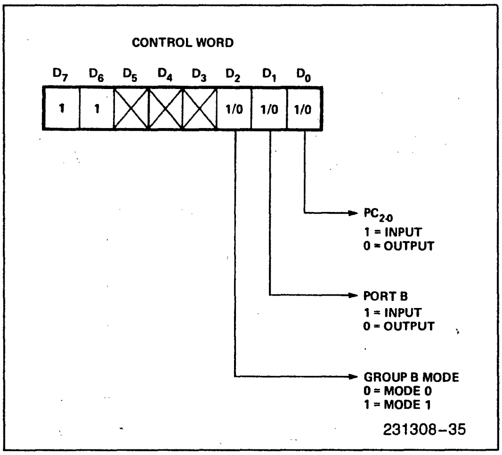


Figure 13. MODE Control Word

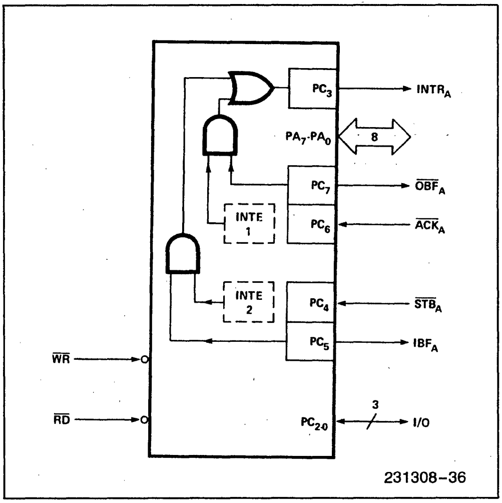
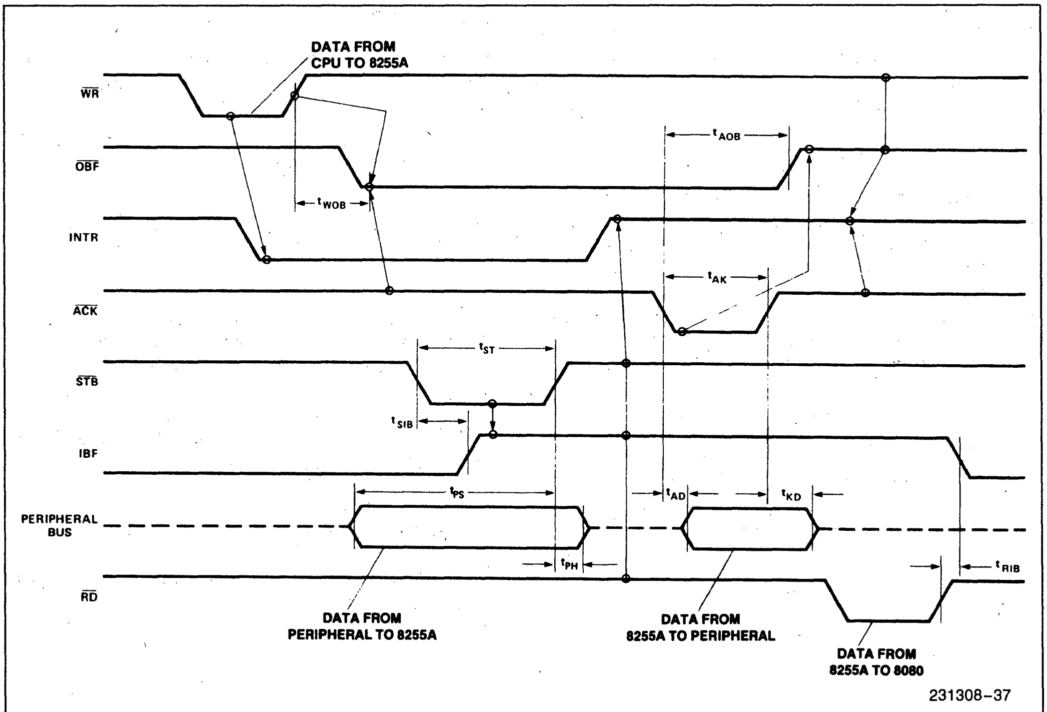
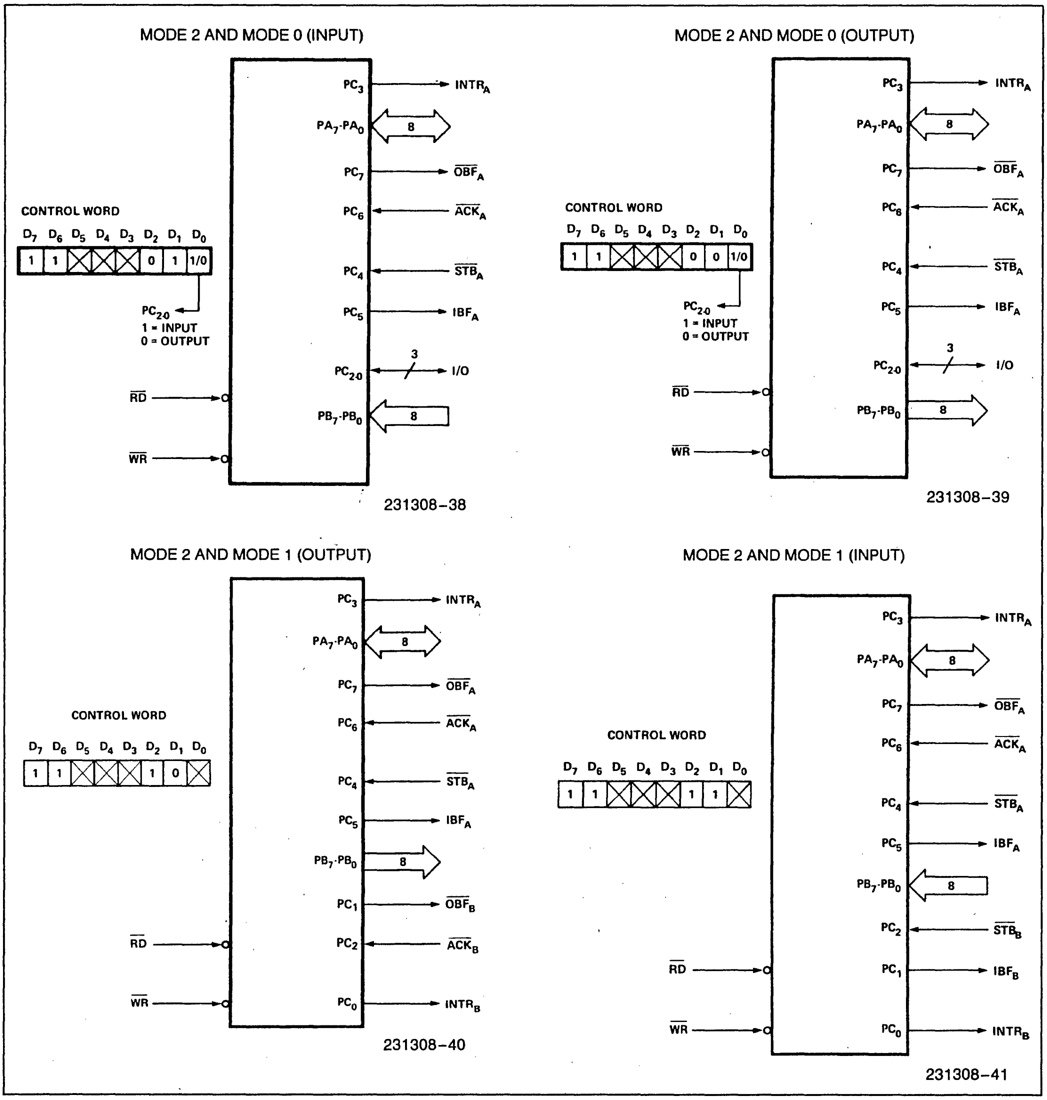


Figure 14. MODE 2



NOTE:
Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ and $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
(INTR = IBF • $\overline{MASK}$ • $\overline{STB}$ • $\overline{RD}$ + $\overline{OBF}$ • $\overline{MASK}$ • $\overline{ACK}$ • $\overline{WR}$)

Figure 15. MODE 2 (Bidirectional)



MODE 2 AND MODE 0 (INPUT)

MODE 2 AND MODE 0 (OUTPUT)

MODE 2 AND MODE 1 (OUTPUT)

MODE 2 AND MODE 1 (INPUT)

Figure 16. MODE ¼ Combinations

Mode Definition Summary

	MODE 0		MODE 1		MODE 2	
	IN	OUT	IN	OUT	GROUP A ONLY	
PA_0	IN	OUT	IN	OUT	↔	
PA_1	IN	OUT	IN	OUT	↔	
PA_2	IN	OUT	IN	OUT	↔	
PA_3	IN	OUT	IN	OUT	↔	
PA_4	IN	OUT	IN	OUT	↔	
PA_5	IN	OUT	IN	OUT	↔	
PA_6	IN	OUT	IN	OUT	↔	
PA_7	IN	OUT	IN	OUT	↔	
PB_0	IN	OUT	IN	OUT	—	MODE 0 OR MODE 1 ONLY
PB_1	IN	OUT	IN	OUT	—	
PB_2	IN	OUT	IN	OUT	—	
PB_3	IN	OUT	IN	OUT	—	
PB_4	IN	OUT	IN	OUT	—	
PB_5	IN	OUT	IN	OUT	—	
PB_6	IN	OUT	IN	OUT	—	
PB_7	IN	OUT	IN	OUT	—	
PC_0	IN	OUT	$INTR_B$	$INTR_B$	I/O	
PC_1	IN	OUT	IBF_B	$\overline{OBF}_B$	I/O	
PC_2	IN	OUT	$\overline{STB}_B$	$\overline{ACK}_B$	I/O	
PC_3	IN	OUT	$INTR_A$	$INTR_A$	$INTR_A$	
PC_4	IN	OUT	$\overline{STB}_A$	I/O	$\overline{STB}_A$	
PC_5	IN	OUT	IBF_A	I/O	IBF_A	
PC_6	IN	OUT	I/O	$\overline{ACK}_A$	$\overline{ACK}_A$	
PC_7	IN	OUT	I/O	$\overline{OBF}_A$	$\overline{OBF}_A$	

Special Mode Combination Considerations

There are several combinations of modes when not all of the bits in Port C are used for control or status. The remaining bits can be used as follows:

If Programmed as Inputs—

All input lines can be accessed during a normal Port C read.

If Programmed as Outputs—

Bits in C upper (PC_7–PC_4) must be individually accessed using the bit set/reset function.

Bits in C lower (PC_3–PC_0) can be accessed using the bit set/reset function or accessed as a threesome by writing into Port C.

Source Current Capability on Port B and Port C

Any set of **eight** output buffers, selected randomly from Ports B and C can source 1 mA at 1.5 volts. This feature allows the 8255 to directly drive Darlington type drivers and high-voltage displays that require such source current.

Reading Port C Status

In Mode 0, Port C transfers data to or from the peripheral device. When the 8255 is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.

INPUT CONFIGURATION

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
I/O	I/O	IBF$_A$	INTE$_A$	INTR$_A$	INTE$_B$	IBF$_B$	INTR$_B$
GROUP A					GROUP B		

OUTPUT CONFIGURATION

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$\overline{OBF}_A$	INTE$_A$	I/O	I/O	INTR$_A$	INTE$_B$	$\overline{OBF}_B$	INTR$_B$
GROUP A					GROUP B		

231308-59

Figure 17. MODE 1 Status Word Format

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$\overline{OBF}_A$	INTE$_1$	IBF$_A$	INTE$_2$	INTR$_A$	☒	☒	☒
GROUP A					GROUP B		

(DEFINED BY MODE 0 OR MODE 1 SELECTION)

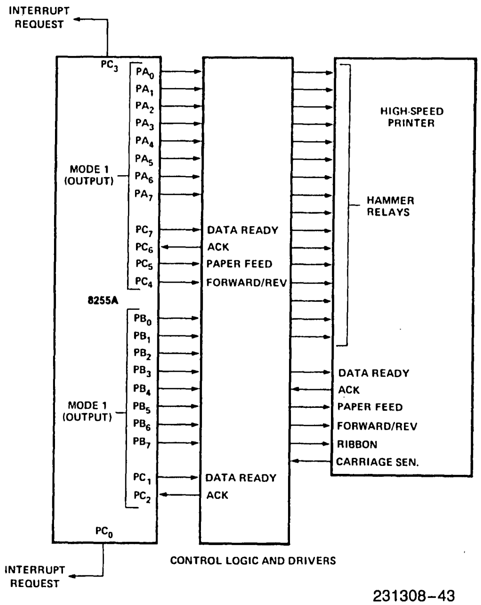
231308-42

Figure 18. MODE 2 Status Word Format

APPLICATIONS OF THE 8255A

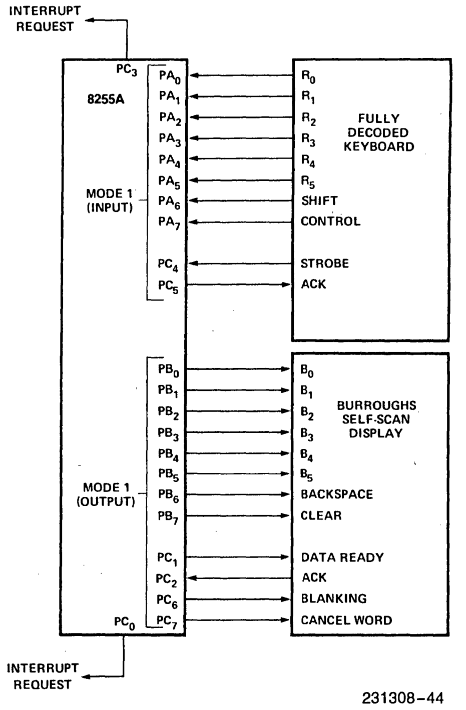
The 8255A is a very powerful tool for interfacing peripheral equipment to the microcomputer system. It represents the optimum use of available pins and is flexible enough to interface almost any I/O device without the need for additional external logic.

Each peripheral device in a microcomputer system usually has a "service routine" associated with it. The routine manages the software interface between the device and the CPU. The functional definition of the 8255A is programmed by the I/O service routine and becomes an extension of the system software. By examining the I/O devices interface characteristics for both data transfer and timing, and matching this information to the examples and tables in the detailed operational description, a control word can easily be developed to initialize the 8255A to exactly "fit" the application. Figures 19 through 25 represent a few examples of typical applications of the 8255A.



231308-43

Figure 19. Printer Interface



231308-44

Figure 20. Keyboard and Display Interface

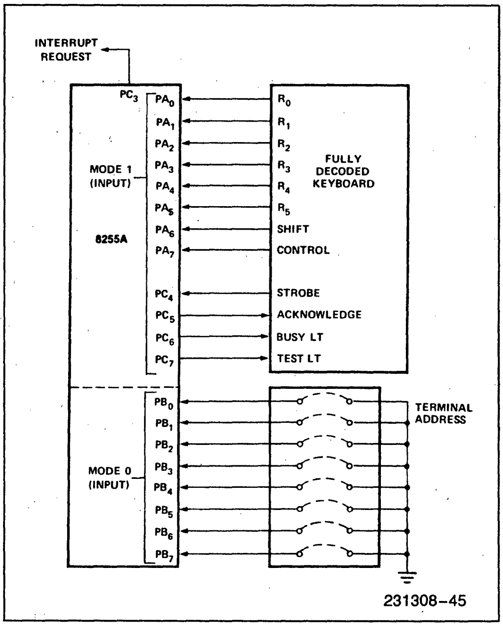


Figure 21. Keyboard and Terminal Address Interface

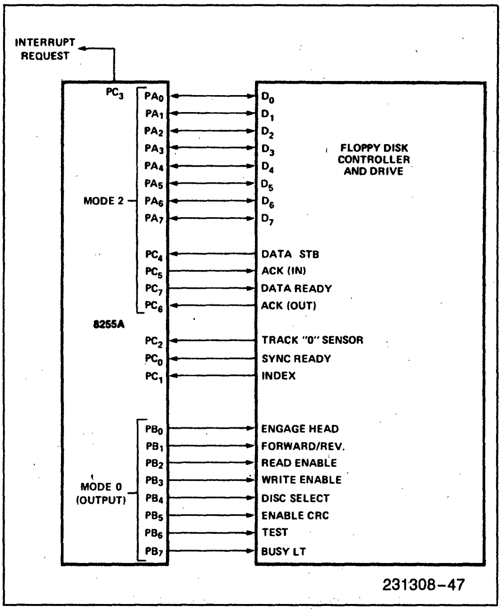


Figure 23. Basic Floppy Disk Interface

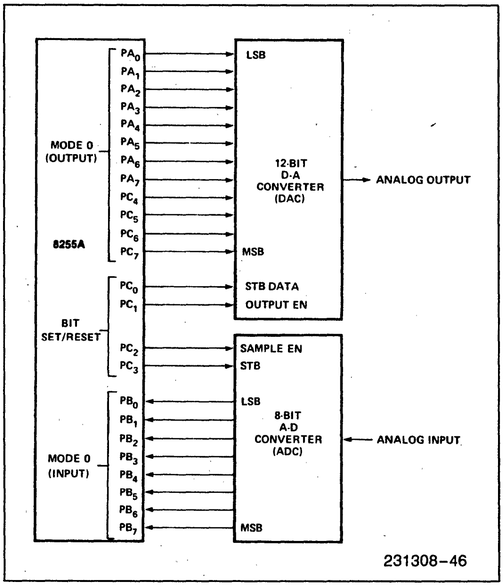


Figure 22. Digital to Analog, Analog to Digital

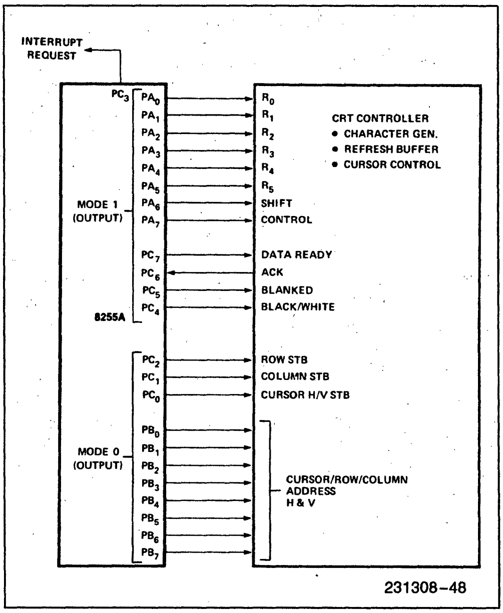


Figure 24. Basic CRT Controller Interface

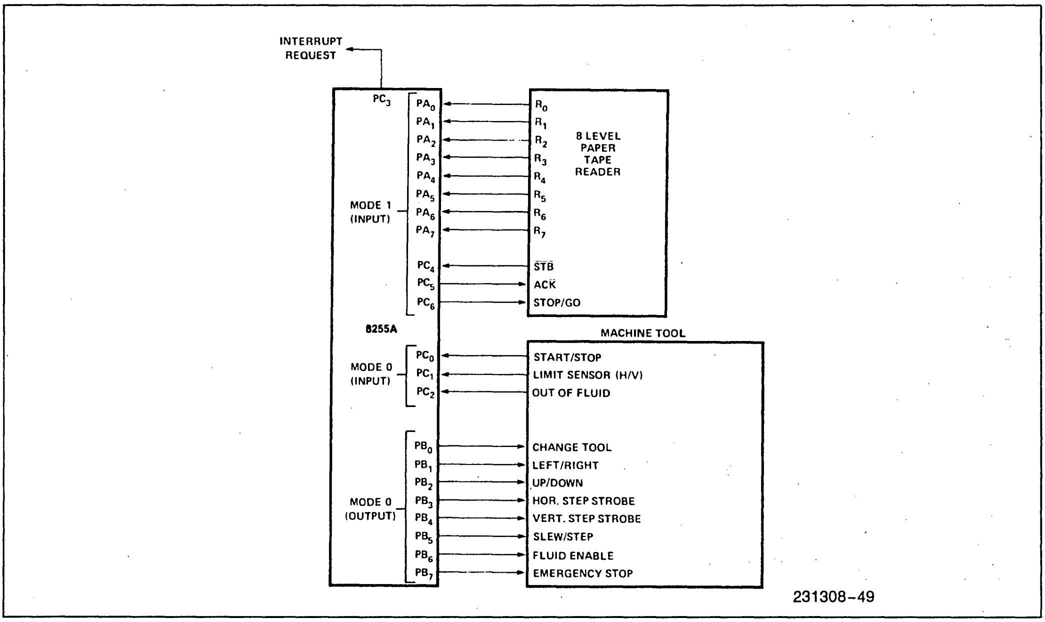


Figure 25. Machine Tool Controller Interface

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 ....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_A = 0°C to 70°C, V_{CC} = +5V ±10%, GND = 0V*

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}	V	
V_{OL} (DB)	Output Low Voltage (Data Bus)		0.45*	V	I_{OL} = 2.5 mA
V_{OL} (PER)	Output Low Voltage (Peripheral Port)		0.45*	V	I_{OL} = 1.7 mA
V_{OH} (DB)	Output High Voltage (Data Bus)	2.4		V	I_{OH} = −400 μA
V_{OH} (PER)	Output High Voltage (Peripheral Port)	2.4		V	I_{OH} = −200 μA
I_{DAR}[1]	Darlington Drive Current	−1.0	−4.0	mA	R_{EXT} = 750Ω; V_{EXT} = 1.5V
I_{CC}	Power Supply Current		120	mA	
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to 0.45V

NOTE:
1. Available on any 8 pins from Port B and C.

CAPACITANCE $T_A = 25°C, V_{CC} = GND = 0V$

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	$f_c = 1$ MHz(4)
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to GND(4)

A.C. CHARACTERISTICS $T_A = 0°C$ to $70°C$, $V_{CC} = +5V \pm 10\%$, GND = 0V*

Bus Parameters

READ

Symbol	Parameter	8255A		8255A-5		Unit
		Min	Max	Min	Max	
t_{AR}	Address Stable before READ	0		0		ns
t_{RA}	Address Stable after READ	0		0		ns
t_{RR}	READ Pulse Width	300		300		ns
t_{RD}	Data Valid from READ(1)		250		200	ns
t_{DF}	Data Float after READ	10	150	10	100	ns
t_{RV}	Time between READs and/or WRITEs	850		850		ns

WRITE

Symbol	Parameter	8255A		8255A-5		Unit
		Min	Max	Min	Max	
t_{AW}	Address Stable before WRITE	0		0		ns
t_{WA}	Address Stable after WRITE	20		20		ns
t_{WW}	WRITE Pulse Width	400		300		ns
t_{DW}	Data Valid to WRITE (T.E.)	100		100		ns
t_{WD}	Data Valid after WRITE	30		30		ns

OTHER TIMINGS

Symbol	Parameter	8255A		8255A-5		Unit
		Min	Max	Min	Max	
t_{WB}	WR = 1 to Output(1)		350		350	ns
t_{IR}	Peripheral Data before RD	0		0		ns
t_{HR}	Peripheral Data after RD	0		0		ns
t_{AK}	ACK Pulse Width	300		300		ns
t_{ST}	STB Pulse Width	500		500		ns
t_{PS}	Per. Data before T.E. of STB	0		0		ns
t_{PH}	Per. Data after T.E. of STB	180		180		ns
t_{AD}	ACK = 0 to Output(1)		300		300	ns
t_{KD}	ACK = 1 to Output Float	20	250	20	250	ns

A.C. CHARACTERISTICS (Continued)

OTHER TIMINGS (Continued)

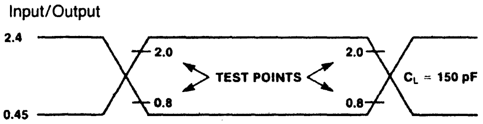
Symbol	Parameter	8255A		8255A-5		Unit
		Min	Max	Min	Max	
t_{WOB}	WR = 1 to OBF = 0[1]		650		650	ns
t_{AOB}	ACK = 0 to OBF = 1[1]		350		350	ns
t_{SIB}	STB = 0 to IBF = 1[1]		300		300	ns
t_{RIB}	RD = 1 to IBF = 0[1]		300		300	ns
t_{RIT}	RD = 0 to INTR = 0[1]		400		400	ns
t_{SIT}	STB = 1 to INTR = 1[1]		300		300	ns
t_{AIT}	ACK = 1 to INTR = 1[1]		350		350	ns
t_{WIT}	WR = 0 to INTR = 0[1, 3]		850		850	ns

NOTES:

1. Test Conditions: C_L = 150 pF.
2. Period of Reset pulse must be at least 50 μs during or after power on. Subsequent Reset pulse can be 500 ns min.
3. INTR ↑ may occur as early as $\overline{WR}$ ↓.
4. Sampled, not 100% tested.

*For Extended Temperature EXPRESS, use M8255A electrical parameters.

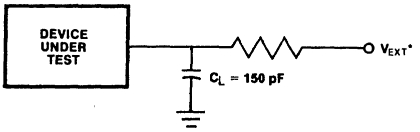
A.C. TESTING INPUT, OUTPUT WAVEFORM



231308-50

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

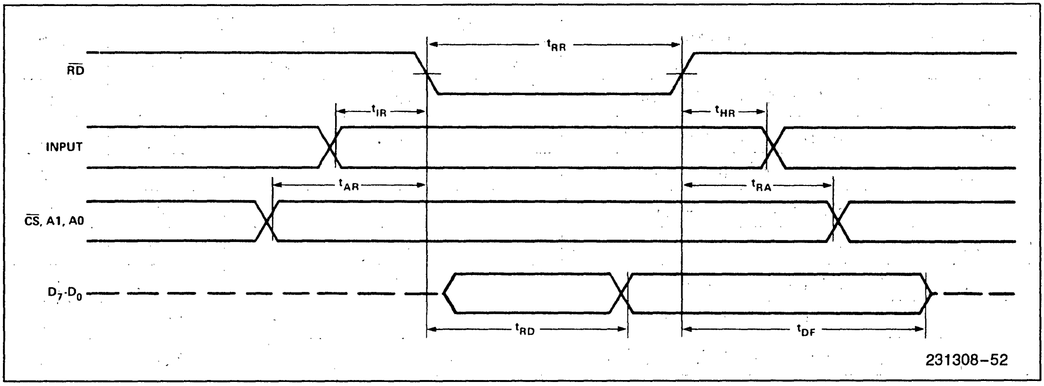


231308-51

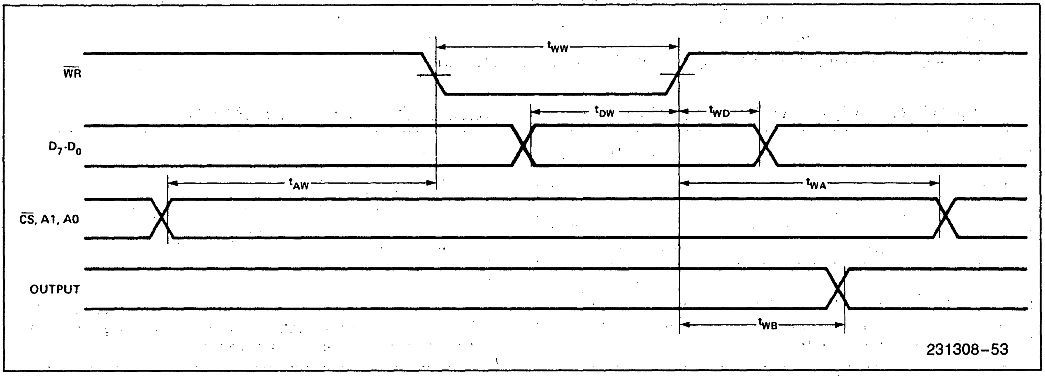
*V_{EXT} is set at various voltages during testing to guarantee the specification. C_L includes jig capacitance.

WAVEFORMS

MODE 0 (BASIC INPUT)

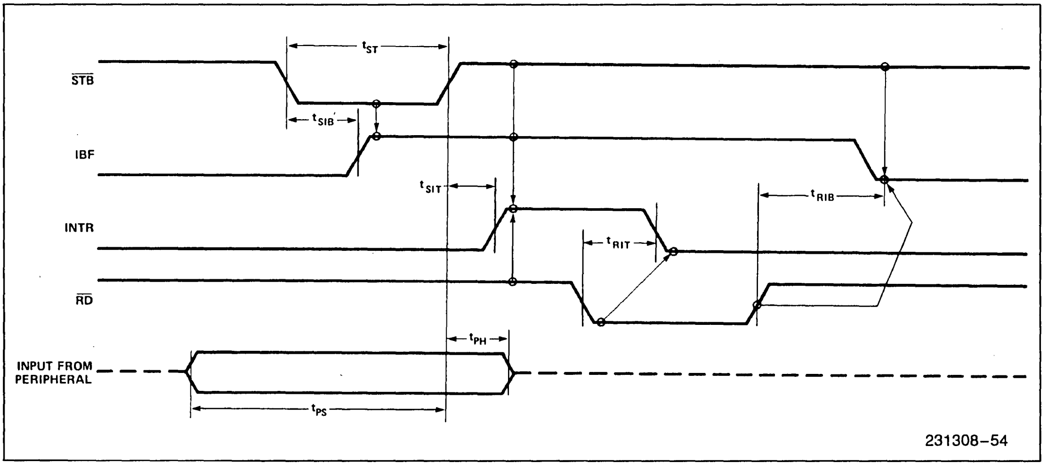


MODE 0 (BASIC OUTPUT)



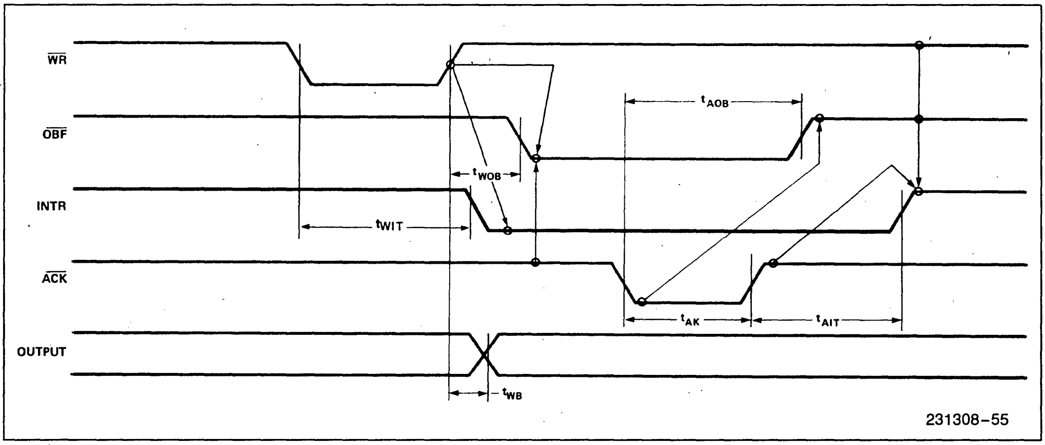
WAVEFORMS (Continued)

MODE 1 (STROBED INPUT)



231308-54

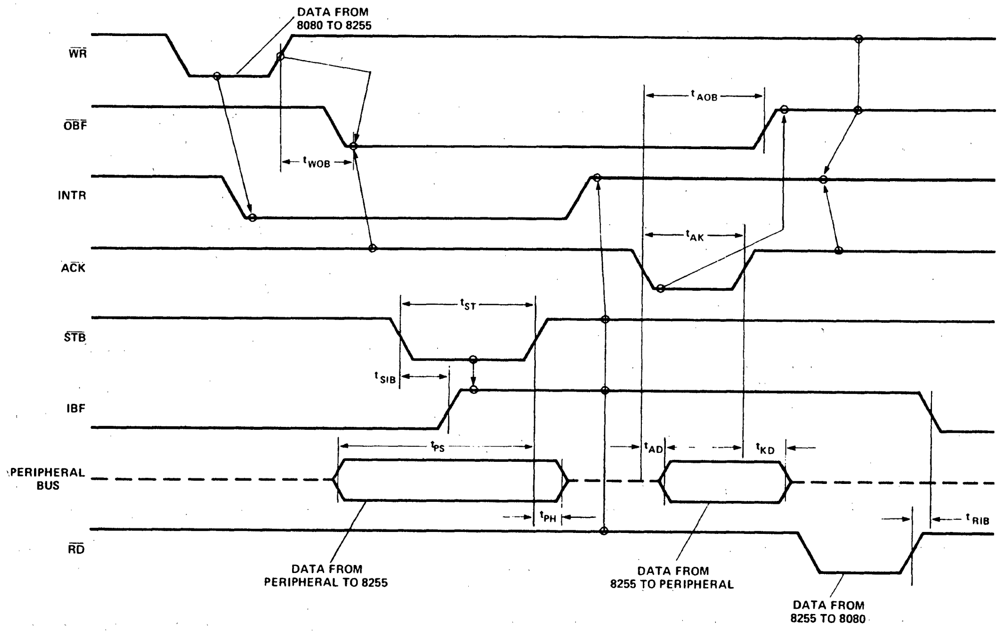
MODE 1 (STROBED OUTPUT)



231308-55

WAVEFORMS (Continued)

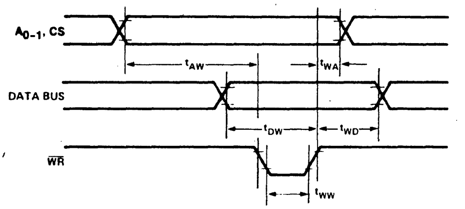
MODE 2 (BIDIRECTIONAL)



231308-56

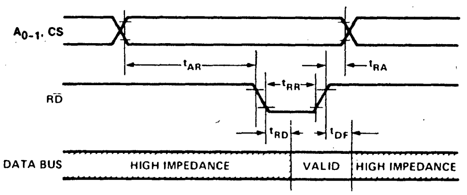
NOTE:
Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ and $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
(INTR = IBF • $\overline{MASK}$ • $\overline{STB}$ • $\overline{RD}$ + $\overline{OBF}$ • $\overline{MASK}$ • $\overline{ACK}$ • $\overline{WR}$)

WRITE TIMING



231308-57

READ TIMING



231308-58

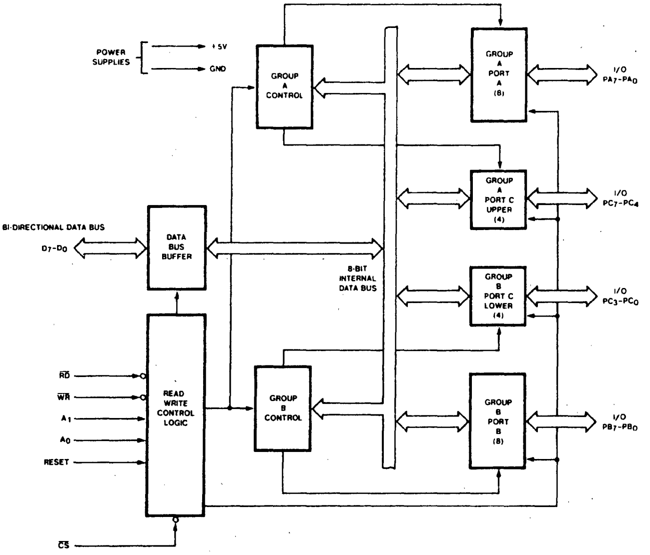
82C55A
CHMOS PROGRAMMABLE PERIPHERAL INTERFACE

- **Compatible with all Intel and Most Other Microprocessors**
- **High Speed, "Zero Wait State" Operation with 8 MHz 8086/88 and 80186/188**
- **24 Programmable I/O Pins**
- **Low Power CHMOS**
- **Completely TTL Compatible**
- **Control Word Read-Back Capability**
- **Direct Bit Set/Reset Capability**
- **2.5 mA DC Drive Capability on all I/O Port Outputs**
- **Available in 40-Pin DIP and 44-Pin PLCC**
- **Available in EXPRESS**
  — Standard Temperature Range
  — Extended Temperature Range

The Intel 82C55A is a high-performance, CHMOS version of the industry standard 8255A general purpose programmable I/O device which is designed for use with all Intel and most other microprocessors. It provides 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. The 82C55A is pin compatible with the NMOS 8255A and 8255A-5.

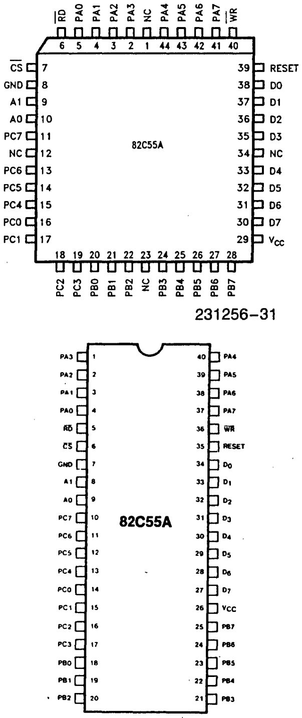
In MODE 0, each group of 12 I/O pins may be programmed in sets of 4 and 8 to be inputs or outputs. In MODE 1, each group may be programmed to have 8 lines of input or output. 3 of the remaining 4 pins are used for handshaking and interrupt control signals. MODE 2 is a strobed bi-directional bus configuration.

The 82C55A is fabricated on Intel's advanced CHMOS III technology which provides low power consumption with performance equal to or greater than the equivalent NMOS product. The 82C55A is available in 40-pin DIP and 44-pin plastic leaded chip carrier (PLCC) packages.



231256-1

Figure 1. 82C55A Block Diagram



231256-2

Figure 2. 82C55A Pinout

Diagrams are for pin reference only. Package sizes are not to scale.

Table 1. Pin Description

<table>
<tr><th rowspan="2">Symbol</th><th colspan="2">Pin Number</th><th rowspan="2">Type</th><th rowspan="2" colspan="6">Name and Function</th></tr>
<tr><th>Dip</th><th>PLCC</th></tr>
<tr><td>PA$_{3-0}$</td><td>1–4</td><td>2–5</td><td>I/O</td><td colspan="6"><b>PORT A, PINS 0–3:</b> Lower nibble of an 8-bit data output latch/buffer and an 8-bit data input latch.</td></tr>
<tr><td>$\overline{RD}$</td><td>5</td><td>6</td><td>I</td><td colspan="6"><b>READ CONTROL:</b> This input is low during CPU read operations.</td></tr>
<tr><td>$\overline{CS}$</td><td>6</td><td>7</td><td>I</td><td colspan="6"><b>CHIP SELECT:</b> A low on this input enables the 82C55A to respond to $\overline{RD}$ and $\overline{WR}$ signals. $\overline{RD}$ and WR are ignored otherwise.</td></tr>
<tr><td>GND</td><td>7</td><td>8</td><td></td><td colspan="6"><b>System Ground</b></td></tr>
<tr><td rowspan="14">A$_{1-0}$</td><td rowspan="14">8–9</td><td rowspan="14">9–10</td><td rowspan="14">I</td><td colspan="6"><b>ADDRESS:</b> These input signals, in conjunction $\overline{RD}$ and $\overline{WR}$, control the selection of one of the three ports or the control word registers.</td></tr>
<tr><th>A$_1$</th><th>A$_0$</th><th>$\overline{RD}$</th><th>$\overline{WR}$</th><th>$\overline{CS}$</th><th>Input Operation (Read)</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>Port A - Data Bus</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>Port B - Data Bus</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>Port C - Data Bus</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>Control Word - Data Bus</td></tr>
<tr><td colspan="5"></td><th>Output Operation (Write)</th></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>Data Bus - Port A</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>Data Bus - Port B</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>Data Bus - Port C</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>Data Bus - Control</td></tr>
<tr><td colspan="5"></td><th>Disable Function</th></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>1</td><td>Data Bus - 3 - State</td></tr>
<tr><td>X</td><td>X</td><td>1</td><td>1</td><td>0</td><td>Data Bus - 3 - State</td></tr>
<tr><td>PC$_{7-4}$</td><td>10–13</td><td>11,13–15</td><td>I/O</td><td colspan="6"><b>PORT C, PINS 4–7:</b> Upper nibble of an 8-bit data output latch/buffer and an 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.</td></tr>
<tr><td>PC$_{0-3}$</td><td>14–17</td><td>16–19</td><td>I/O</td><td colspan="6"><b>PORT C, PINS 0–3:</b> Lower nibble of Port C.</td></tr>
<tr><td>PB$_{0-7}$</td><td>18–25</td><td>20–22,
24–28</td><td>I/O</td><td colspan="6"><b>PORT B, PINS 0–7:</b> An 8-bit data output latch/buffer and an 8-bit data input buffer.</td></tr>
<tr><td>V$_{CC}$</td><td>26</td><td>29</td><td></td><td colspan="6"><b>SYSTEM POWER:</b> + 5V Power Supply.</td></tr>
<tr><td>D$_{7-0}$</td><td>27–34</td><td>30–33,
35–38</td><td>I/O</td><td colspan="6"><b>DATA BUS:</b> Bi-directional, tri-state data bus lines, connected to system data bus.</td></tr>
<tr><td>RESET</td><td>35</td><td>39</td><td>I</td><td colspan="6"><b>RESET:</b> A high on this input clears the control register and all ports are set to the input mode.</td></tr>
<tr><td>$\overline{WR}$</td><td>36</td><td>40</td><td>I</td><td colspan="6"><b>WRITE CONTROL:</b> This input is low during CPU write operations.</td></tr>
<tr><td>PA$_{7-4}$</td><td>37–40</td><td>41–44</td><td>I/O</td><td colspan="6"><b>PORT A, PINS 4–7:</b> Upper nibble of an 8-bit data output latch/buffer and an 8-bit data input latch.</td></tr>
<tr><td>NC</td><td></td><td>1, 12,
23, 34</td><td></td><td colspan="6">No Connect</td></tr>
</table>

82C55A FUNCTIONAL DESCRIPTION

General

The 82C55A is a programmable peripheral interface device designed for use in Intel microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 82C55A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 82C55A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 82C55A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 82C55A.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A - Port A and Port C upper (C7–C4)
Control Group B - Port B and Port C lower (C3–C0)

The control word register can be both written and read as shown in the address decode table in the pin descriptions. Figure 6 shows the control word format for both Read and Write operations. When the control word is read, bit D7 will always be a logic "1", as this implies control word mode information.

Ports A, B, and C

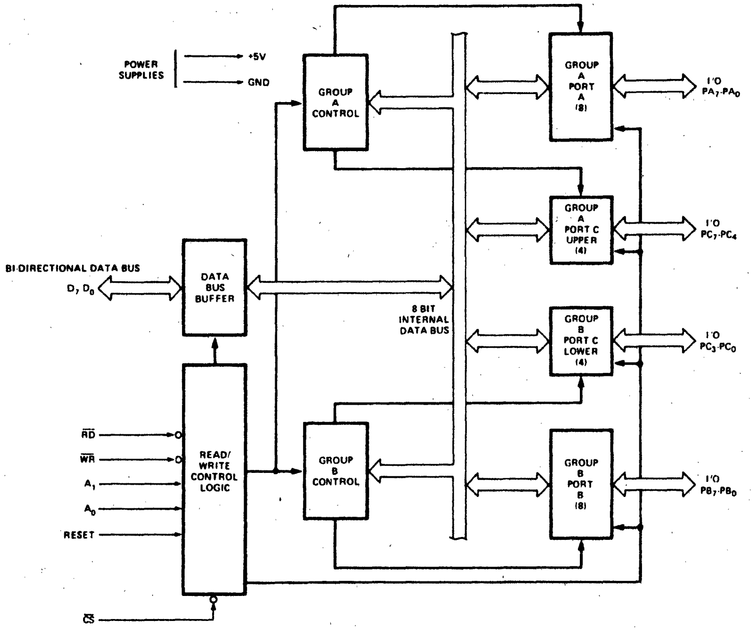
The 82C55A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 82C55A.

Port A. One 8-bit data output latch/buffer and one 8-bit input latch buffer. Both "pull-up" and "pull-down" bus hold devices are present on Port A.

Port B. One 8-bit data input/output latch/buffer. Only "pull-up" bus hold devices are present on Port B.

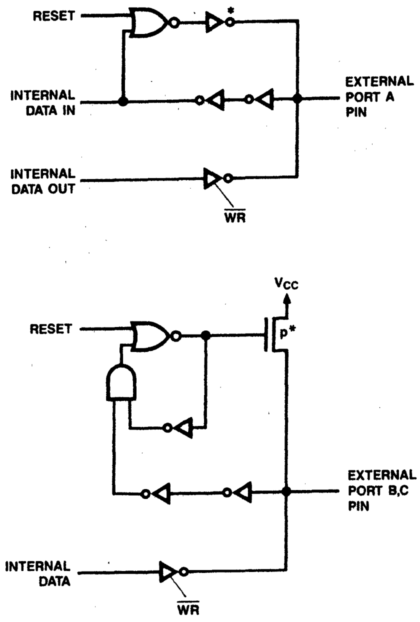
Port C. One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B. Only "pull-up" bus hold devices are present on Port C.

See Figure 4 for the bus-hold circuit configuration for Port A, B, and C.



231256-3

Figure 3. 82C55A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions



***NOTE:**

231256-4

Port pins loaded with more than 20 pF capacitance may not have their logic level guaranteed following a hardware reset.

Figure 4. Port A, B, C, Bus-hold Configuration

82C55A OPERATIONAL DESCRIPTION

Mode Selection

There are three basic modes of operation that can be selected by the system software:

Mode 0 — Basic input/output
Mode 1 — Strobed Input/output
Mode 2 — Bi-directional Bus

When the reset input goes "high" all ports will be set to the input mode with all 24 port lines held at a logic "one" level by the internal bus hold devices (see Figure 4 Note). After the reset is removed the 82C55A can remain in the input mode with no additional initialization required. This eliminates the need for pullup or pulldown devices in "all CMOS" designs. During the execution of the system program, any of the other modes may be selected by using a single output instruction. This allows a single 82C55A to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

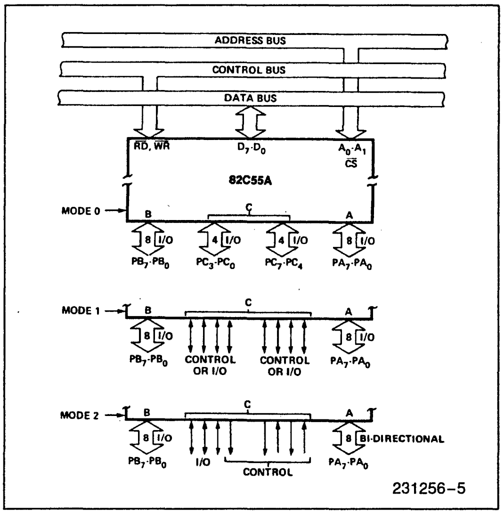


Figure 5. Basic Mode Definitions and Bus Interface

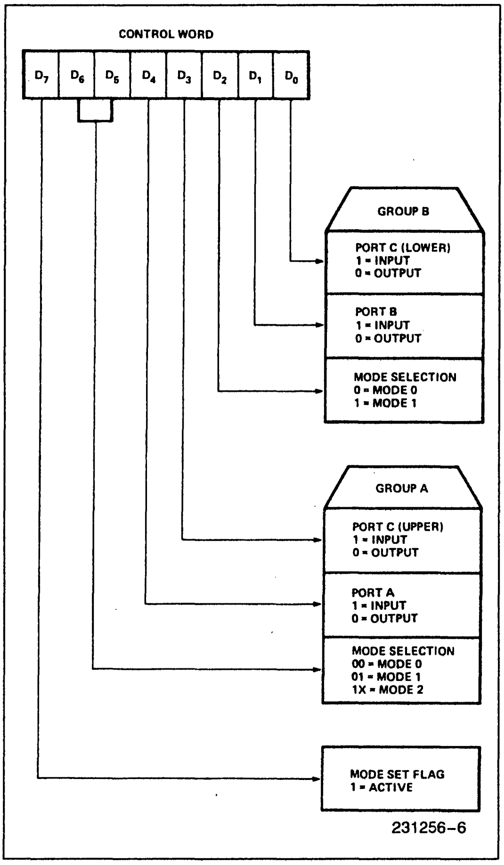


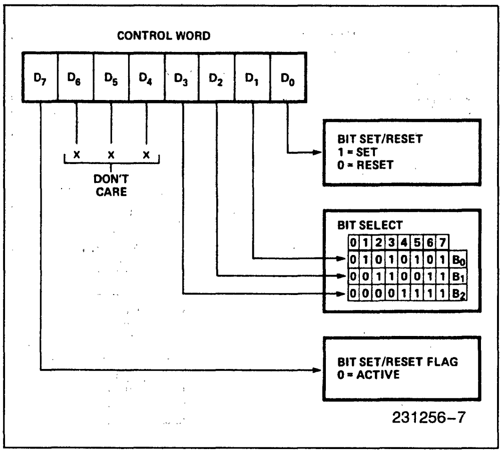
Figure 6. Mode Definition Format

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 82C55A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.



0	1	2	3	4	5	6	7	
0	1	0	1	0	1	0	1	B_0
0	0	1	1	0	0	1	1	B_1
0	0	0	0	1	1	1	1	B_2

Figure 7. Bit Set/Reset Format

Interrupt Control Functions

When the 82C55A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET)—INTE is SET—Interrupt enable
(BIT-RESET)—INTE is RESET—Interrupt disable

Note:
All Mask flip-flops are automatically reset during mode selection and device Reset.

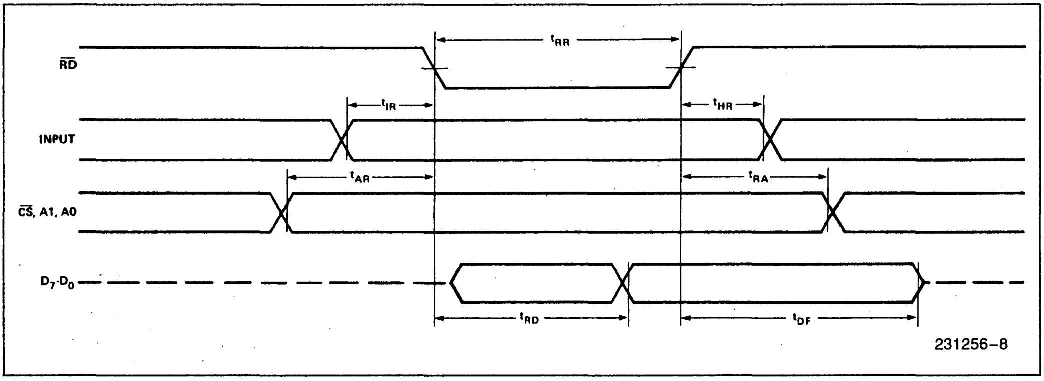
Operating Modes

Mode 0 (Basic Input/Output). This functional configuration provides simple input and output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

Mode 0 Basic Functional Definitions:

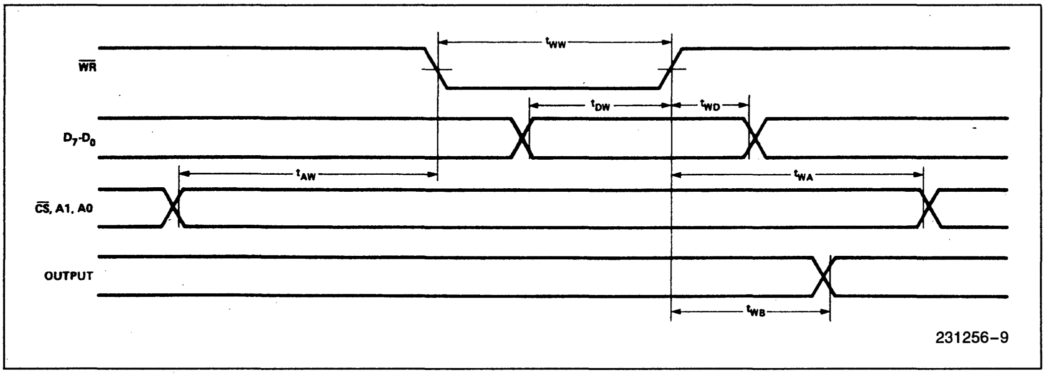
- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.

MODE 0 (BASIC INPUT)



231256–8

MODE 0 (BASIC OUTPUT)

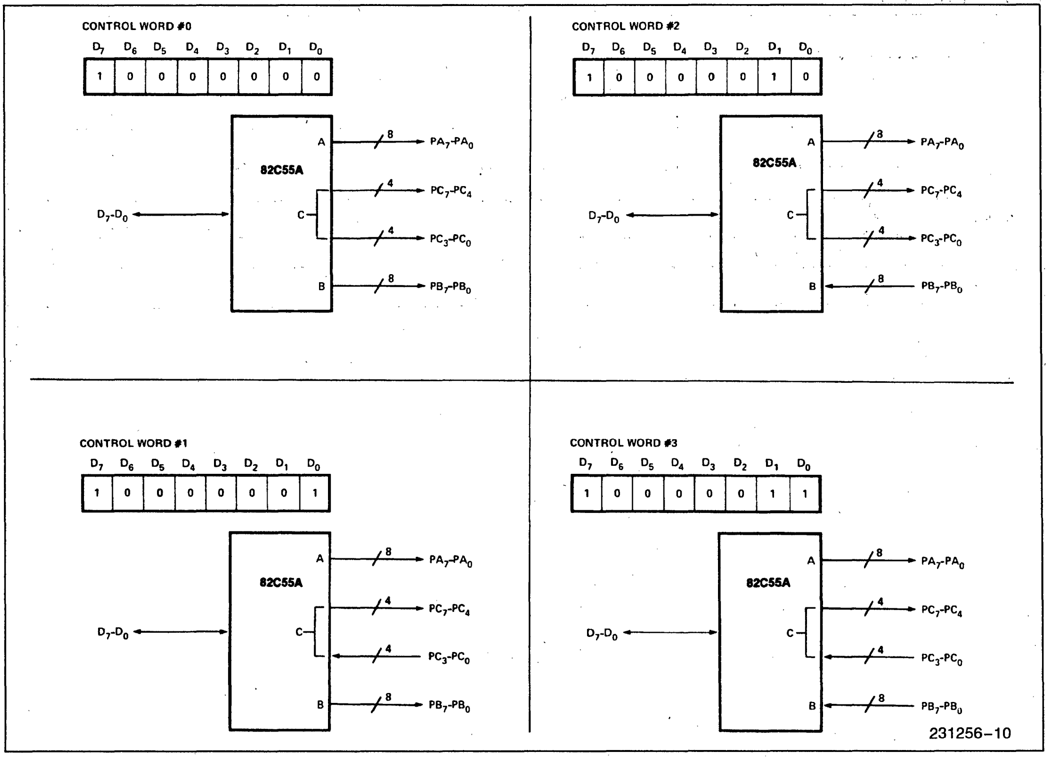


231256–9

MODE 0 Port Definition

A		B		GROUP A			GROUP B	
D_4	D_3	D_1	D_0	PORT A	PORT C (UPPER)	#	PORT B	PORT C (LOWER)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

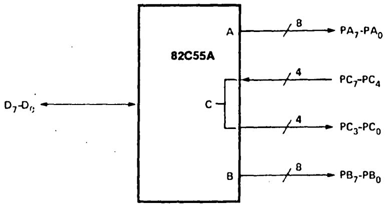
MODE 0 Configurations



MODE 0 Configurations (Continued)

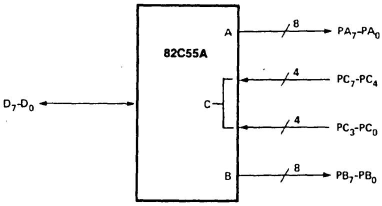
CONTROL WORD #4

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	0



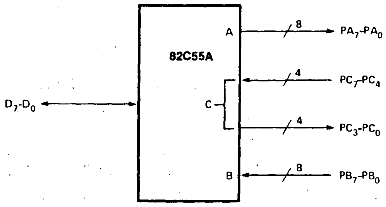
CONTROL WORD #5

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	1



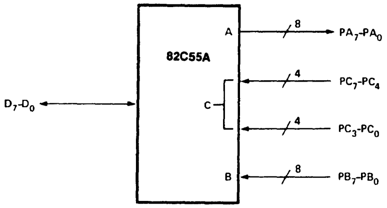
CONTROL WORD #6

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	0



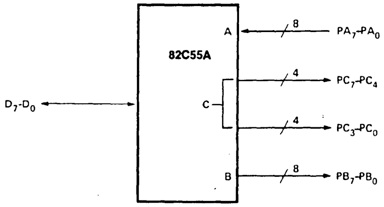
CONTROL WORD #7

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	1



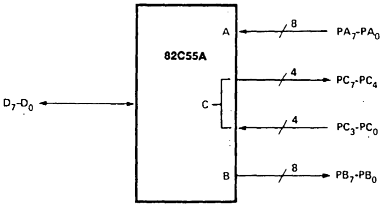
CONTROL WORD #8

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	0



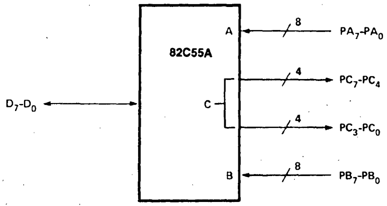
CONTROL WORD #9

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	1



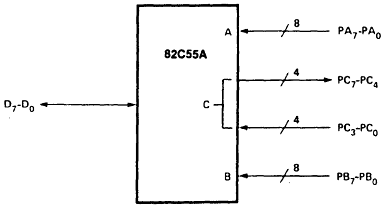
CONTROL WORD #10

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	0



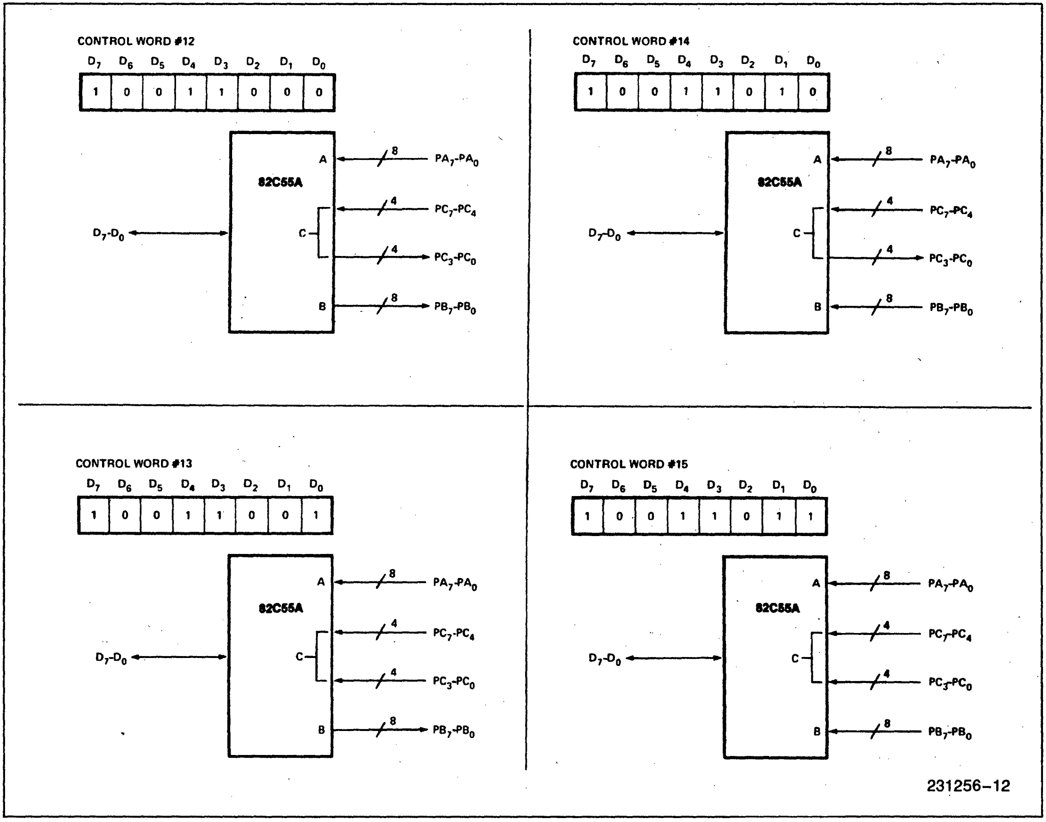
CONTROL WORD #11

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	1



231256–11

MODE 0 Configurations (Continued)



231256-12

Operating Modes

MODE 1 (Strobed Input/Output). This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In mode 1, Port A and Port B use the lines on Port C to generate or accept these "handshaking" signals.

Mode 1 Basic functional Definitions:

- Two Groups (Group A and Group B).
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

Input Control Signal Definition

$\overline{\text{STB}}$ (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by $\overline{\text{STB}}$ input being low and is reset by the rising edge of the $\overline{\text{RD}}$ input.

INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the $\overline{\text{STB}}$ is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{\text{RD}}$. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

INTE A

Controlled by bit set/reset of PC_4.

INTE B

Controlled by bit set/reset of PC_2.

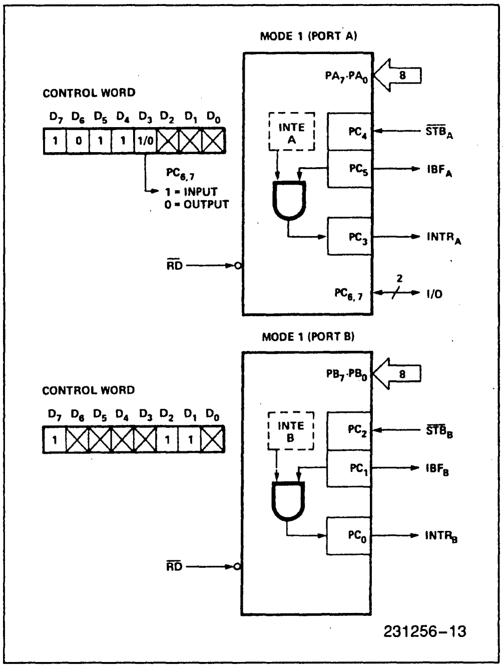


Figure 8. MODE 1 Input

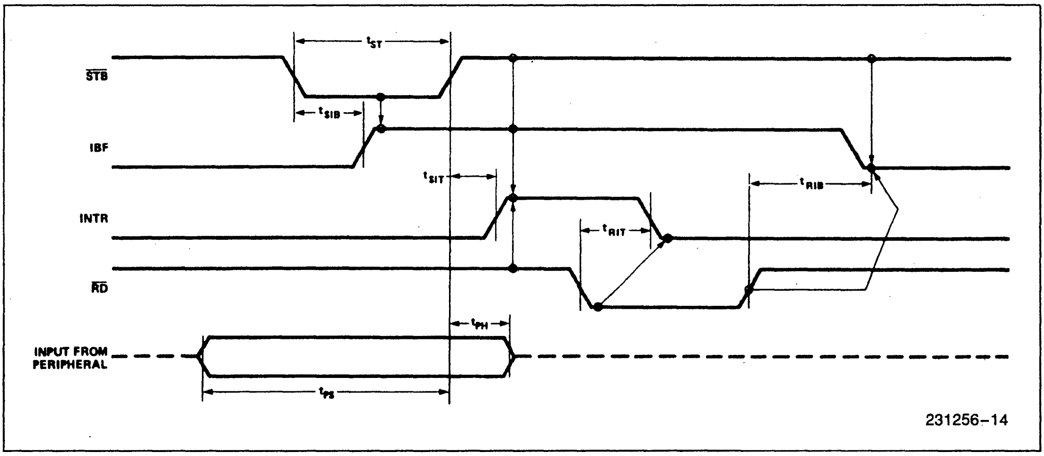


Figure 9. MODE 1 (Strobed Input)

Output Control Signal Definition

$\overline{\text{OBF}}$ (Output Buffer Full F/F). The $\overline{\text{OBF}}$ output will go "low" to indicate that the CPU has written data out to the specified port. The $\overline{\text{OBF}}$ F/F will be set by the rising edge of the $\overline{\text{WR}}$ input and reset by $\overline{\text{ACK}}$ Input being low.

$\overline{\text{ACK}}$ (Acknowledge Input). A "low" on this input informs the 82C55A that the data from Port A or Port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

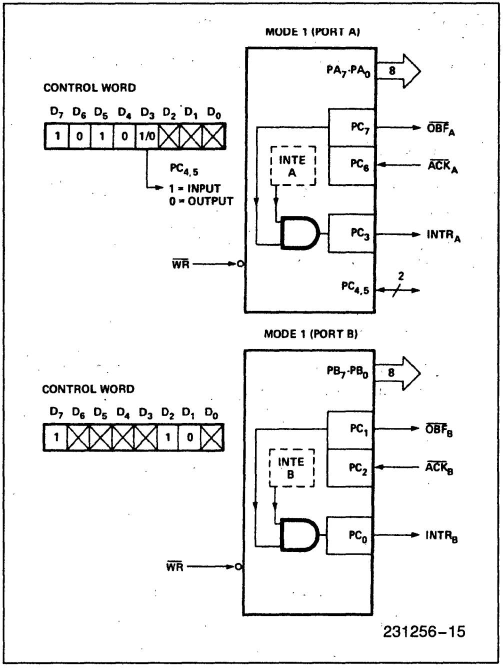
INTR (Interrupt Request). A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set when $\overline{\text{ACK}}$ is a "one", $\overline{\text{OBF}}$ is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{\text{WR}}$.

INTE A

Controlled by bit set/reset of PC_6.

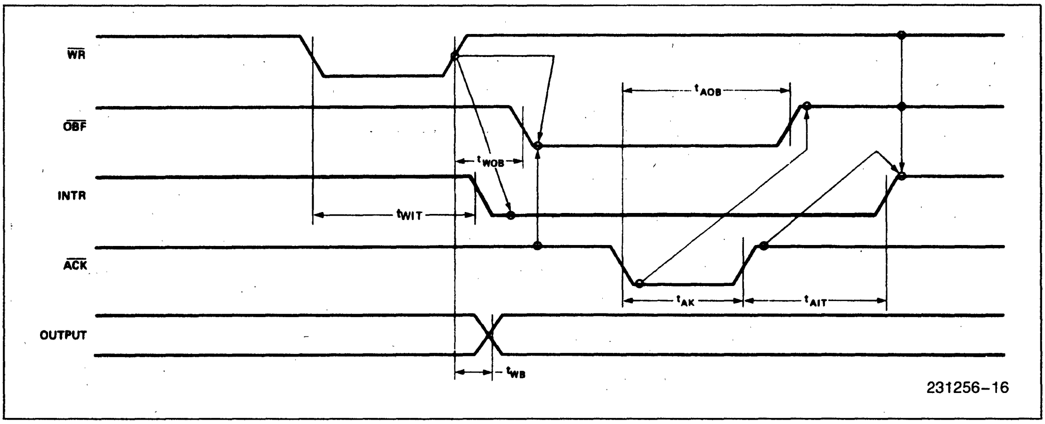
INTE B

Controlled by bit set/reset of PC_2.



231256–15

Figure 10. MODE 1 Output



231256–16

Figure 11. MODE 1 (Strobed Output)

Combinations of MODE 1

Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.

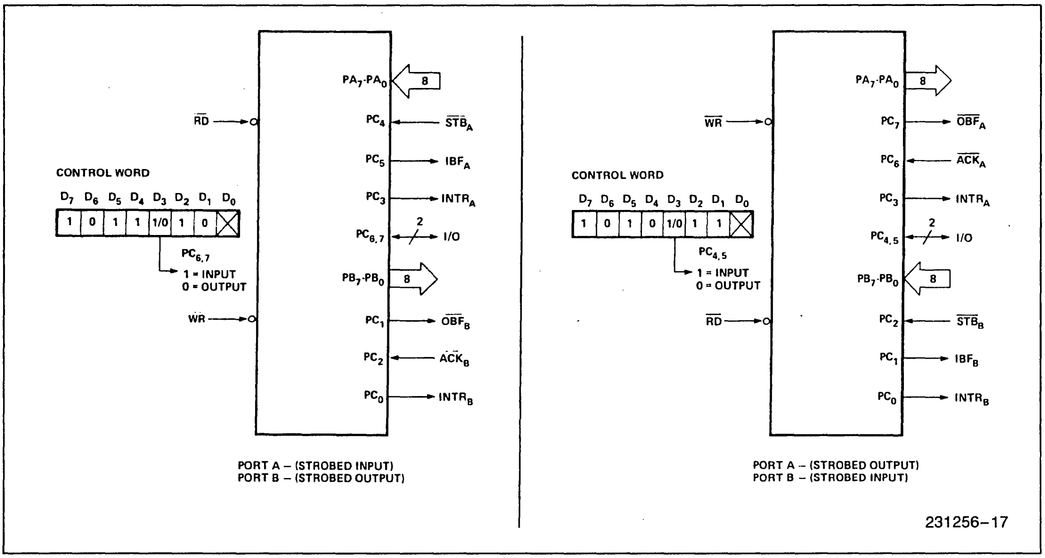


Figure 12. Combinations of MODE 1

Operating Modes

MODE 2 (Strobed Bidirectional Bus I/O). This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

MODE 2 Basic Functional Definitions:

- Used in Group A **only**.
- One 8-bit, bi-directional bus port (Port A) and a 5-bit control port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

Bidirectional Bus I/O Control Signal Definition

INTR (Interrupt Request). A high on this output can be used to interrupt the CPU for input or output operations.

Output Operations

$\overline{\text{OBF}}$ (Output Buffer Full). The $\overline{\text{OBF}}$ output will go "low" to indicate that the CPU has written data out to port A.

$\overline{\text{ACK}}$ (Acknowledge). A "low" on this input enables the tri-state output buffer of Port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

INTE 1 (The INTE Flip-Flop Associated with $\overline{\text{OBF}}$). Controlled by bit set/reset of PC_6.

Input Operations

$\overline{\text{STB}}$ (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F). A "high" on this output indicates that data has been loaded into the input latch.

INTE 2 (The INTE Flip-Flop Associated with IBF). Controlled by bit set/reset of PC_4.

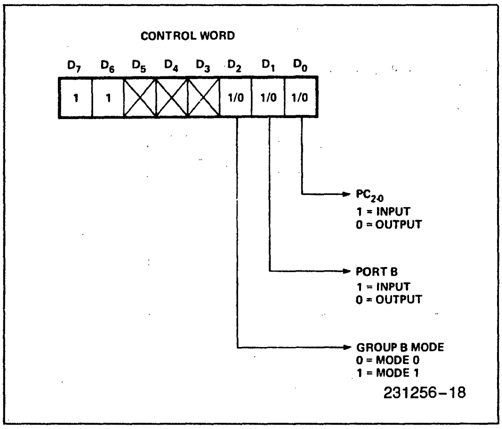


Figure 13. MODE Control Word

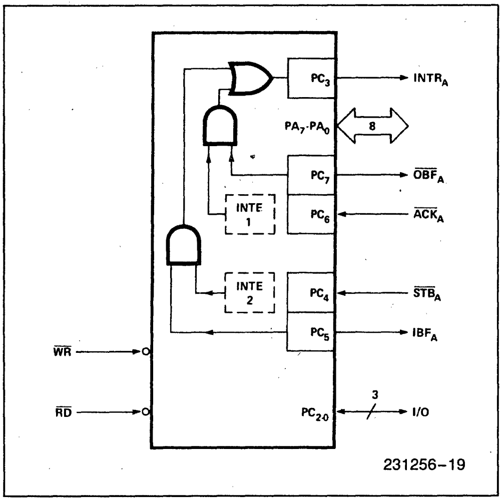


Figure 14. MODE 2

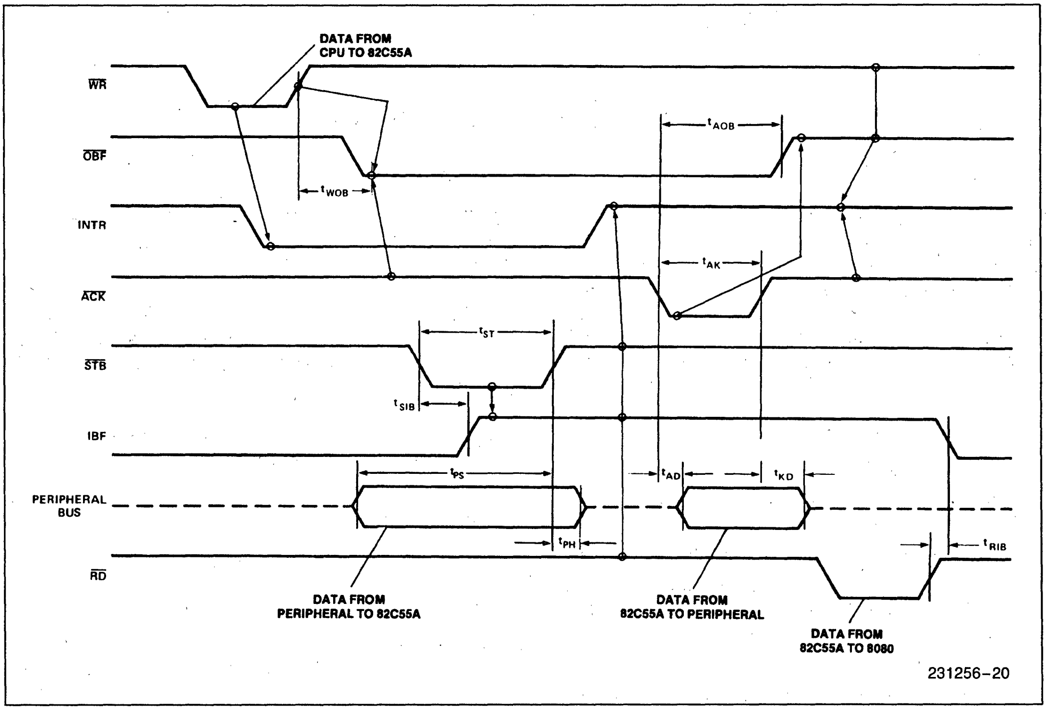


Figure 15. MODE 2 (Bidirectional)

NOTE:
Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$, and $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
(INTR = IBF • $\overline{MASK}$ • $\overline{STB}$ • $\overline{RD}$ + $\overline{OBF}$ • $\overline{MASK}$ • $\overline{ACK}$ • $\overline{WR}$)

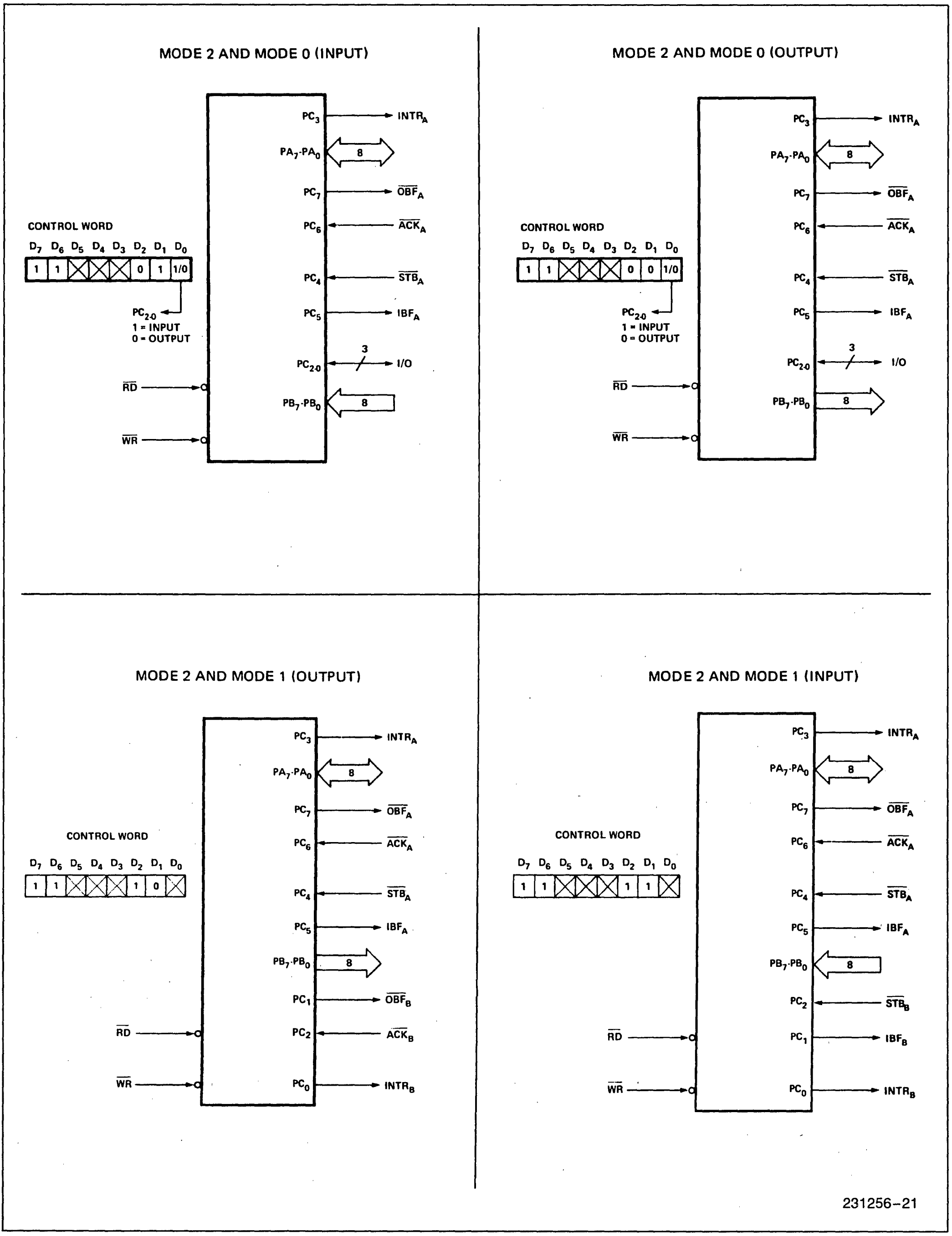


Figure 16. MODE 1/4 Combinations

Mode Definition Summary

	MODE 0		MODE 1		MODE 2
	IN	OUT	IN	OUT	GROUP A ONLY
PA_0	IN	OUT	IN	OUT	↔
PA_1	IN	OUT	IN	OUT	↔
PA_2	IN	OUT	IN	OUT	↔
PA_3	IN	OUT	IN	OUT	↔
PA_4	IN	OUT	IN	OUT	↔
PA_5	IN	OUT	IN	OUT	↔
PA_6	IN	OUT	IN	OUT	↔
PA_7	IN	OUT	IN	OUT	↔
PB_0	IN	OUT	IN	OUT	—
PB_1	IN	OUT	IN	OUT	—
PB_2	IN	OUT	IN	OUT	—
PB_3	IN	OUT	IN	OUT	—
PB_4	IN	OUT	IN	OUT	—
PB_5	IN	OUT	IN	OUT	—
PB_6	IN	OUT	IN	OUT	—
PB_7	IN	OUT	IN	OUT	—
PC_0	IN	OUT	$INTR_B$	$INTR_B$	I/O
PC_1	IN	OUT	IBF_B	$\overline{OBF}_B$	I/O
PC_2	IN	OUT	$\overline{STB}_B$	$\overline{ACK}_B$	I/O
PC_3	IN	OUT	$INTR_A$	$INTR_A$	$INTR_A$
PC_4	IN	OUT	$\overline{STB}_A$	I/O	$\overline{STB}_A$
PC_5	IN	OUT	IBF_A	I/O	IBF_A
PC_6	IN	OUT	I/O	$\overline{ACK}_A$	$\overline{ACK}_A$
PC_7	IN	OUT	I/O	$\overline{OBF}_A$	$\overline{OBF}_A$

MODE 0 OR MODE 1 ONLY

Special Mode Combination Considerations

There are several combinations of modes possible. For any combination, some or all of the Port C lines are used for control or status. The remaining bits are either inputs or outputs as defined by a "Set Mode" command.

During a read of Port C, the state of all the Port C lines, except the $\overline{\text{ACK}}$ and $\overline{\text{STB}}$ lines, will be placed on the data bus. In place of the $\overline{\text{ACK}}$ and $\overline{\text{STB}}$ line states, flag status will appear on the data bus in the PC2, PC4, and PC6 bit positions as illustrated by Figure 18.

Through a "Write Port C" command, only the Port C pins programmed as outputs in a Mode 0 group can be written. No other pins can be affected by a "Write Port C" command, nor can the interrupt enable flags be accessed. To write to any Port C output programmed as an output in a Mode 1 group or to change an interrupt enable flag, the "Set/Reset Port C Bit" command must be used.

With a "Set/Reset Port C Bit" command, any Port C line programmed as an output (including INTR, IBF and $\overline{\text{OBF}}$) can be written, or an interrupt enable flag can be either set or reset. Port C lines programmed as inputs, including $\overline{\text{ACK}}$ and $\overline{\text{STB}}$ lines, associated with Port C are not affected by a "Set/Reset Port C Bit" command. Writing to the corresponding Port C bit positions of the $\overline{\text{ACK}}$ and $\overline{\text{STB}}$ lines with the "Set/Reset Port C Bit" command will affect the Group A and Group B interrupt enable flags, as illustrated in Figure 18.

Current Drive Capability

Any output on Port A, B or C can sink or source 2.5 mA. This feature allows the 82C55A to directly drive Darlington type drivers and high-voltage displays that require such sink or source current.

Reading Port C Status

In Mode 0, Port C transfers data to or from the peripheral device. When the 82C55A is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.

INPUT CONFIGURATION

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
I/O	I/O	IBF_A	$INTE_A$	$INTR_A$	$INTE_B$	IBF_B	$INTR_B$
GROUP A					GROUP B		

OUTPUT CONFIGURATIONS

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$\overline{OBF}_A$	$INTE_A$	I/O	I/O	$INTR_A$	$INTE_B$	$\overline{OBF}_B$	$INTR_B$
GROUP A					GROUP B		

Figure 17a. MODE 1 Status Word Format

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$\overline{OBF}_A$	$INTE_1$	IBF_A	$INTE_2$	$INTR_A$			
GROUP A					GROUP B		

(Defined By Mode 0 or Mode 1 Selection)

Figure 17b. MODE 2 Status Word Format

Interrupt Enable Flag	Position	Alternate Port C Pin Signal (Mode)
INTE B	PC2	$\overline{ACK}_B$ (Output Mode 1) or $\overline{STB}_B$ (Input Mode 1)
INTE A2	PC4	$\overline{STB}_A$ (Input Mode 1 or Mode 2)
INTE A1	PC6	$\overline{ACK}_A$ (Output Mode 1 or Mode 2

Figure 18. Interrupt Enable Flags in Modes 1 and 2

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias....0°C to + 70°C
Storage Temperature ....− 65°C to + 150°C
Supply Voltage ....− 0.5 to + 8.0V
Operating Voltage ....+ 4V to + 7V
Voltage on any Input..........GND−2V to + 6.5V
Voltage on any Output ..GND−0.5V to V_{CC} + 0.5V
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_A = 0°C to 70°C, V_{CC} = +5V ±10%, GND = 0V (T_A = −40°C to +85°C for Extended Temperture)

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}	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 V	I_{OH} = −2.5 mA I_{OH} = −100 μA
I_{IL}	Input Leakage Current		±1	μA	V_{IN} = V_{CC} to 0V (Note 1)
I_{OFL}	Output Float Leakage Current		±10	μA	V_{IN} = V_{CC} to 0V (Note 2)
I_{DAR}	Darlington Drive Current	±2.5	(Note 4)	mA	Ports A, B, C R_{ext} = 500Ω V_{ext} = 1.7V
I_{PHL}	Port Hold Low Leakage Current	+50	+300	μA	V_{OUT} = 1.0V Port A only
I_{PHH}	Port Hold High Leakage Current	−50	−300	μA	V_{OUT} = 3.0V Ports A, B, C
I_{PHLO}	Port Hold Low Overdrive Current	−350		μA	V_{OUT} = 0.8V
I_{PHHO}	Port Hold High Overdrive Current	+350		μA	V_{OUT} = 3.0V
I_{CC}	V_{CC} Supply Current		10	mA	(Note 3)
I_{CCSB}	V_{CC} Supply Current-Standby		10	μA	V_{CC} = 5.5V V_{IN} = V_{CC} or GND Port Conditions If I/P = Open/High O/P = Open Only With Data Bus = High/Low $\overline{CS}$ = High Reset = Low Pure Inputs = Low/High

NOTES:
1. Pins A_1, A_0, $\overline{CS}$, $\overline{WR}$, $\overline{RD}$, Reset.
2. Data Bus; Ports B, C.
3. Outputs open.
4. Limit output current to 4.0 mA.

CAPACITANCE

T_A = 25°C, V_{CC} = GND = 0V

Symbol	Parameter	Min	Max	Units	Test Conditions
C_{IN}	Input Capacitance		10	pF	Unmeasured pIns returned to GND f_c = 1 MHz(5)
$C_{I/O}$	I/O Capacitance		20	pF	

NOTE:

5. Sampled not 100% tested.

A.C. CHARACTERISTICS

T_A = 0° to 70°C, V_{CC} = +5V ±10%, GND = 0V

T_A = −40°C to +85°C for Extended Temperature

BUS PARAMETERS

READ CYCLE

Symbol	Parameter	82C55A-2		Units	Test Conditions
		Min	Max		
t_{AR}	Address Stable Before $\overline{\text{RD}}$ ↓	0		ns	
t_{RA}	Address Hold Time After $\overline{\text{RD}}$ ↑	0		ns	
t_{RR}	$\overline{\text{RD}}$ Pulse Width	150		ns	
t_{RD}	Data Delay from $\overline{\text{RD}}$ ↓		120	ns	
t_{DF}	$\overline{\text{RD}}$ ↑ to Data Floating	10	75	ns	
t_{RV}	Recovery Time between $\overline{\text{RD}}$/$\overline{\text{WR}}$	200		ns	

WRITE CYCLE

Symbol	Parameter	82C55A-2		Units	Test Conditions
		Min	Max		
t_{AW}	Address Stable Before $\overline{\text{WR}}$ ↓	0		ns	
t_{WA}	Address Hold Time After $\overline{\text{WR}}$ ↑	20		ns	Ports A & B
		20		ns	Port C
t_{WW}	$\overline{\text{WR}}$ Pulse Width	100		ns	
t_{DW}	Data Setup Time Before $\overline{\text{WR}}$ ↑	100		ns	
t_{WD}	Data Hold Time After $\overline{\text{WR}}$ ↑	30		ns	Ports A & B
		30		ns	Port C

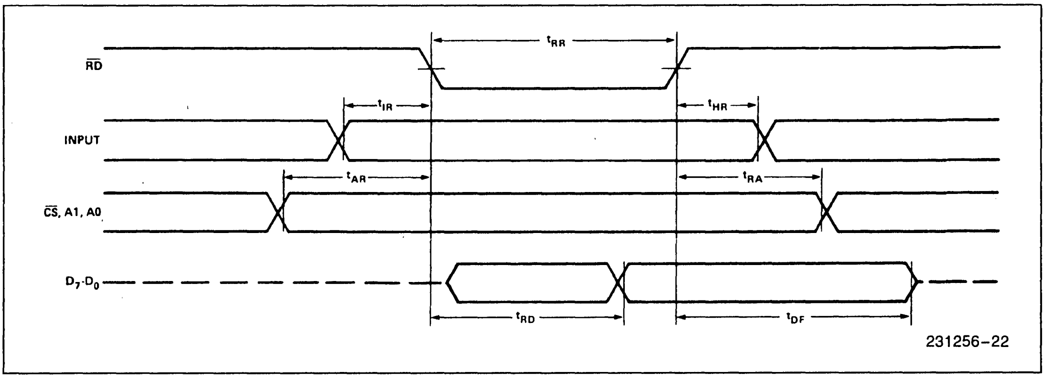
OTHER TIMINGS

Symbol	Parameter	82C55A-2		Units Conditions	Test
		Min	Max		
t_{WB}	$\overline{WR}$ = 1 to Output		350	ns	
t_{IR}	Peripheral Data Before $\overline{RD}$	0		ns	
t_{HR}	Peripheral Data After $\overline{RD}$	0		ns	
t_{AK}	$\overline{ACK}$ Pulse Width	200		ns	
t_{ST}	$\overline{STB}$ Pulse Width	100		ns	
t_{PS}	Per. Data Before $\overline{STB}$ High	20		ns	
t_{PH}	Per. Data After $\overline{STB}$High	50		ns	
t_{AD}	$\overline{ACK}$ = 0 to Output		175	ns	
t_{KD}	$\overline{ACK}$ = 1 to Output Float	20	250	ns	
t_{WOB}	$\overline{WR}$ = 1 to $\overline{OBF}$ = O		150	ns	
t_{AOB}	$\overline{ACK}$ = 0 to $\overline{OBF}$ = 1		150	ns	
t_{SIB}	$\overline{STB}$ = 0 to IBF = 1		150	ns	
t_{RIB}	$\overline{RD}$ = 1 to IBF = 0		150	ns	
t_{RIT}	$\overline{RD}$ = 0 to INTR = 0		200	ns	
t_{SIT}	$\overline{STB}$ = 1 to INTR = 1		150	ns	
t_{AIT}	$\overline{ACK}$ = 1 to INTR = 1		150	ns	
t_{WIT}	$\overline{WR}$ = 0 to INTR = 0		200	ns	see note 1
t_{RES}	Reset Pulse Width	500		ns	see note 2

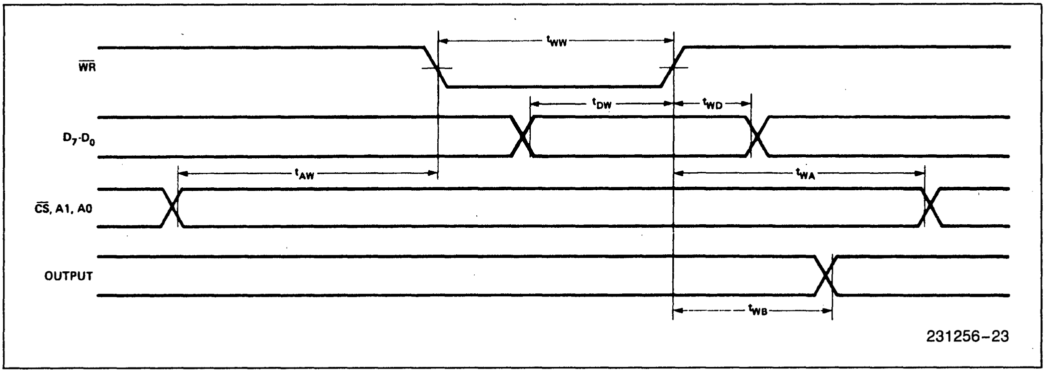
NOTE:
1. INTR ↑ may occur as early as $\overline{WR}$ ↓.
2. Pulse width of initial Reset pulse after power on must be at least 50 μSec. Subsequent Reset pulses may be 500 ns minimum.

WAVEFORMS

MODE 0 (BASIC INPUT)

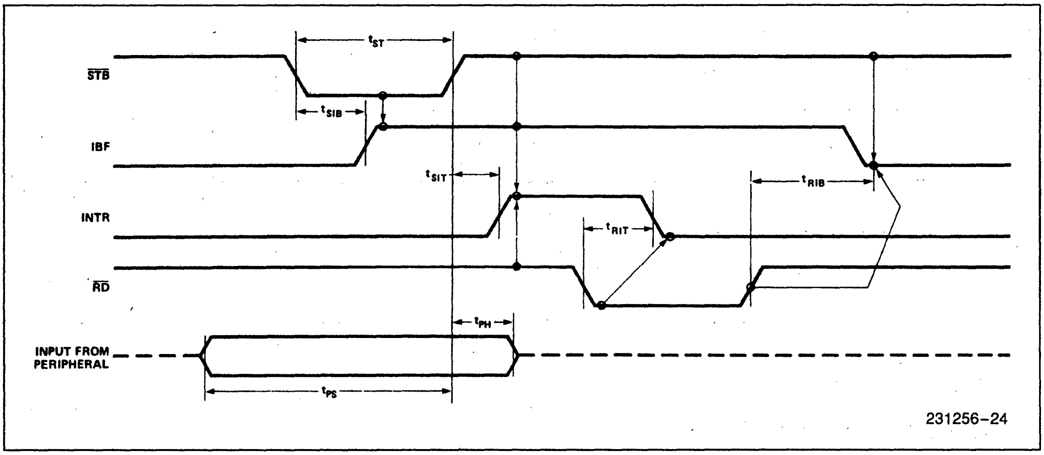


MODE 0 (BASIC OUTPUT)



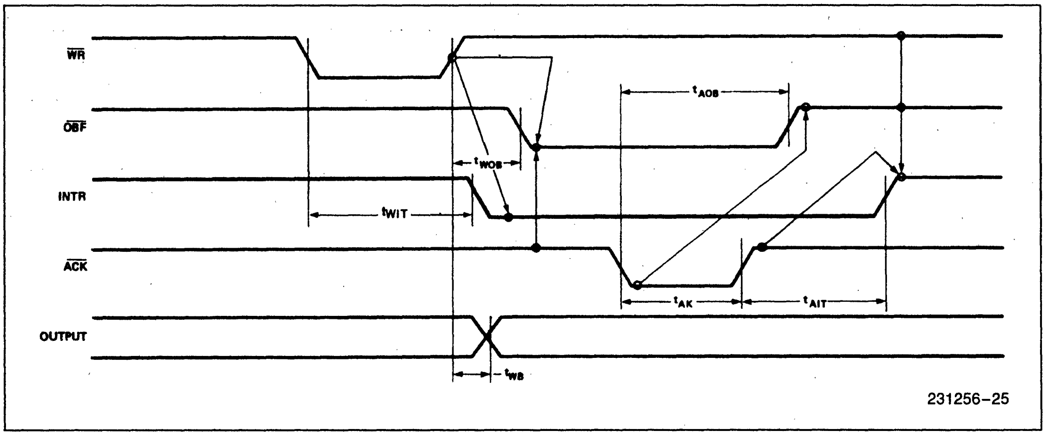
WAVEFORMS (Continued)

MODE 1 (STROBED INPUT)



231256–24

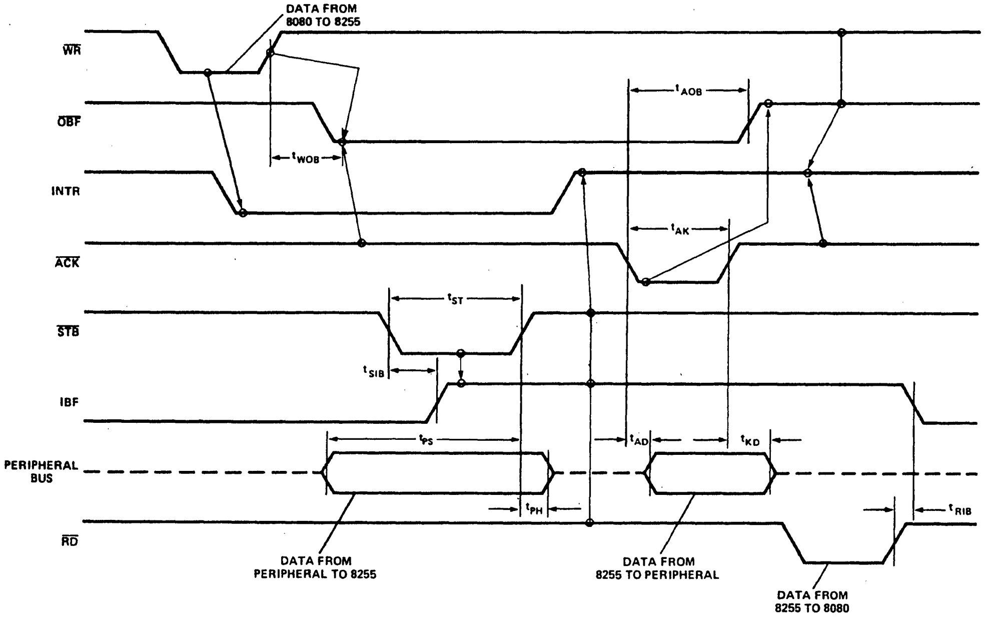
MODE 1 (STROBED OUTPUT)



231256–25

WAVEFORMS (Continued)

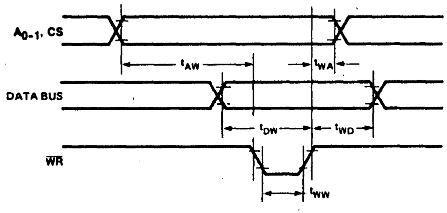
MODE 2 (BIDIRECTIONAL)



231256–26

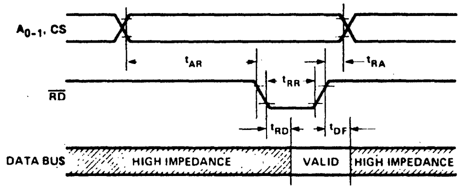
Note:
Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ AND $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
$(INTR = IBF \bullet \overline{MASK} \bullet \overline{STB} \bullet \overline{RD} + \overline{OBF} \bullet \overline{MASK} \bullet \overline{ACK} \bullet \overline{WR})$

WRITE TIMING



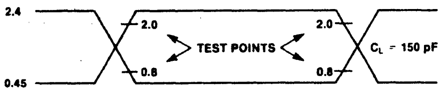
231256–27

READ TIMING



231256–28

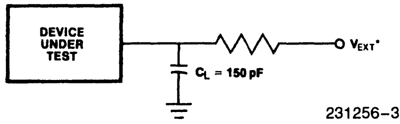
A.C. TESTING INPUT, OUTPUT WAVEFORM



231256–29

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 0.8 For A Logic 0.

A.C. TESTING LOAD CIRCUIT



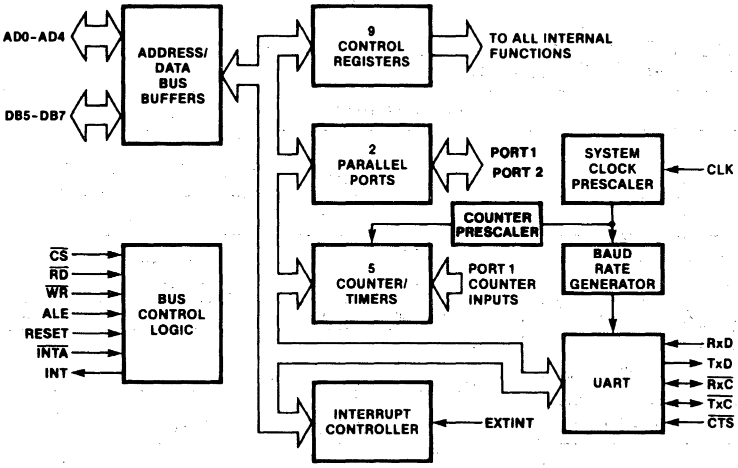
231256–30

*V_{EXT} Is Set At Various Voltages During Testing To Guarantee The Specification. C_L Includes Jig Capacitance.

8256AH MULTIFUNCTION MICROPROCESSOR SUPPORT CONTROLLER

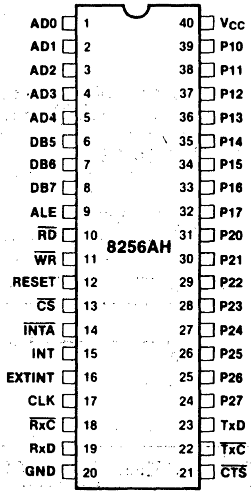
- **Programmable Serial Asynchronous Communications Interface for 5-, 6-, 7-, or 8-Bit Characters, 1, 1½, or 2 Stop Bits, and Parity Generation**
- **On-Board Baud Rate Generator Programmable for 13 Common Baud Rates up to 19.2 KBits/Second, or an External Baud Clock Maximum of 1M Bit/Second**
- **Five 8-Bit Programmable Timer/Counters; Four Can Be Cascaded to Two 16-Bit Timer/Counters**
- **Two 8-Bit Programmable Parallel I/O Ports; Port 1 Can Be Programmed for Port 2 Handshake Controls and Event Counter Inputs**
- **Eight-Level Priority Interrupt Controller Programmable for 8085 or iAPX 86, iAPX 88 Systems and for Fully Nested Interrupt Capability**
- **Programmable System Clock to 1 ×, 2 ×, 3 ×, or 5 × 1.024 MHz**

The Intel® 8256AH Multifunction Universal Asynchronous Receiver-Transmitter (MUART) combines five commonly used functions into a single 40-pin device. It is designed to interface to the 8086/88, iAPX 186/188, and 8051 to perform serial communications, parallel I/O, timing, event counting, and priority interrupt functions. All of these functions are fully programmable through nine internal registers. In addition, the five timer/counters and two parallel I/O ports can be accessed directly by the microprocessor.



230759-1

Figure 1. MUART Block Diagram



230759-2

Figure 2. MUART Pin Configuration

Table 1. Pin Description

Symbol	Pin	Type	Name and Function
AD0–AD4 DB5–DB7	1–5 6–8	I/O	**ADDRESS/DATA:** Three-state address/data lines which interface to the lower 8 bits of the microprocessor's multiplexed address/data bus. The 5-bit address is latched on the falling edge of ALE. In the 8-bit mode, AD0–AD3 are used to select the proper register, while AD1–AD4 are used in the 16-bit mode. AD4 in the 8-bit mode is ignored as an address, while AD0 in the 16-bit mode is used as a second chip select, active low.
ALE	9	I	**ADDRESS LATCH ENABLE:** Latches the 5 address lines on AD0–AD4 and $\overline{CS}$ on the falling edge.
$\overline{RD}$	10	I	**READ CONTROL:** When this signal is low, the selected register is gated onto the data bus.
$\overline{WR}$	11	I	**WRITE CONTROL:** When this signal is low, the value on the data bus is written into the selected register.
RESET	12	I	**RESET:** An active high pulse on this pin forces the chip into its initial state. The chip remains in this state until control information is written.
$\overline{CS}$	13	I	**CHIP SELECT:** A low on this signal enables the MUART. It is latched with the address on the falling edge of ALE, and $\overline{RD}$ and $\overline{WR}$ have no effect unless $\overline{CS}$ was latched low during the ALE cycle.
$\overline{INTA}$	14	I	**INTERRUPT ACKNOWLEDGE:** If the MUART has been enabled to respond to interrupts, this signal informs the MUART that its interrupt request is being acknowledged by the microprocessor. During this acknowledgement the MUART puts an RSTn instruction on the data bus for the 8-bit mode or a vector for the 16-bit mode.
INT	15	O	**INTERRUPT REQUEST:** A high signals the microprocessor that the MUART needs service.
EXTINT	16	I	**EXTERNAL INTERRUPT:** An external device can request interrupt service through this input. The input is level sensitive (high), therefore it must be held high until an $\overline{INTA}$ occurs or the interrupt address register is read.
CLK	17	I	**SYSTEM CLOCK:** The reference clock for the baud rate generator and the timers.
RxC	18	I/O	**RECEIVE CLOCK:** If the baud rate bits in the Command Register 2 are all 0, this pin is an input which clocks serial data into the RxD pin on the rising edge of RxC. If baud rate bits in Command Register 2 are programmed from 1–0FH, this pin outputs a square wave whose rising edge indicates when the data on RxD is being sampled. This output remains high during start, stop, and parity bits.
RxD	19	I	**RECEIVE DATA:** Serial data input.
GND	20	PS	**GROUND:** Power supply and logic ground reference.

Table 1. Pin Description (Continued)

Symbol	Pin	Type	Name and Function
$\overline{\text{CTS}}$	21	I	**CLEAR TO SEND:** This input enables the serial transmitter. If 1, 1.5, or 2 stop bits are selected $\overline{\text{CTS}}$ is level sensitive. As long as $\overline{\text{CTS}}$ is low, any character loaded into the transmitter buffer register will be transmitted serially. A single negative going pulse causes the transmission of a single character previously loaded into the transmitter buffer register. If a baud rate from 1–0FH is selected, $\overline{\text{CTS}}$ must be low for at least $\frac{1}{32}$ of a bit, or it will be ignored. If the transmitter buffer is empty, this pulse will be ignored. If this pulse occurs during the transmission of a character up to the time where $\frac{1}{2}$ the first (or only) stop bit is sent out, it will be ignored. If it occurs afterwards, but before the end of the stop bits, the next character will be transmitted immediately following the current one. If $\overline{\text{CTS}}$ is still high when the transmitter register is sending the last stop bit, the transmitter will enter its idle state until the next high-to-low transition on $\overline{\text{CTS}}$ occurs. If 0.75 stop bits is chosen, the $\overline{\text{CTS}}$ input is edge sensitive. A negative edge on $\overline{\text{CTS}}$ results in the immediate transmission of the next character. The length of the stop bits is determined by the time interval between the beginning of the first stop bit and the next negative edge on $\overline{\text{CTS}}$. A high-to-low transition has no effect if the transmitter buffer is empty or if the time interval between the beginning of the stop bit and next negative edge is less than 0.75 bits. A high or a low level or a low-to-high transition has no effect on the transmitter for the 0.75 stop bit mode.
TxC	22	I/O	**TRANSMIT CLOCK:** If the baud rate bits in command register 2 are all set to 0, this input clocks data out of the transmitter on the falling edge. If baud rate bits are programmed for 1 or 2, this input permits the user to provide a 32× or 64× clock which is used for the receiver and transmitter. If the baud rate bits are programmed for 3–0FH, the internal transmitter clock is output. As an output it delivers the transmitter clock at the selected bit rate. If $1\frac{1}{2}$ or 0.75 stop bits are selected, the transmitter divider will be asynchronously reset at the beginning of each start bit, immediately causing a high-to-low transition on TxC. TxC makes a high-to-low transition at the beginning of each serial bit, and a low-to-high transition at the center of each bit.
TxD	23	O	**TRANSMIT DATA:** Serial data output.
P27–P20	24–31	I/O	**PARALLEL I/O PORT 2:** Eight bit general purpose I/O port. Each nibble (4 bits) of this port can be either an input or an output. The outputs are latched whereas the input signals are not. Also, this port can be used as an 8-bit input or output port when using the two-wire handshake. In the handshake mode both inputs and outputs are latched.
P17–P10	32–39	I/O	**PARALLEL I/O PORT 1:** Each pin can be programmed as an input or an output to perform general purpose I/O. All outputs are latched whereas inputs are not. Alternatively these pins can serve as control pins which extend the functional spectrum of the chip.
V_{CC}	40	PS	**POWER:** +5V power supply.

FUNCTIONAL DESCRIPTION

The 8256AH Multi-Function Universal Asynchronous Receiver-Transmitter (MUART) combines five commonly used functions into a single 40-pin device. The MUART performs asynchronous serial communications, parallel I/O, timing, event counting, and interrupt control. For detailed application information, see Intel AP Note #153, Designing with the 8256.

Serial Communications

The serial communications portion of the MUART contains a full-duplex asynchronous receiver-transmitter (UART). A programmable baud rate generator is included on the MUART to permit a variety of operating speeds without external components. The UART can be programmed by the CPU for a variety of character sizes, parity generation and detection, error detection, and start/stop bit handling. The receiver checks the start and stop bits in the center of the bit, and a break halts the reception of data. The transmitter can send breaks and can be controlled by an external enable pin.

Parallel I/O

The MUART includes 16 bits of general purpose parallel I/O. Eight bits (Port 1) can be individually changed from input to output or used for special I/O functions. The other eight bits (Port 2) can be used as nibbles (4 bits) or as bytes. These eight bits also include a handshaking capability using two pins on Port 1.

Counter/Timers

There are five 8-bit counter/timers on the MUART. The timers can be programmed to use either a 1 kHz or 16 kHz clock generated from the system clock. Four of the 8-bit counter/timers can be cascaded to two 16-bit counter/timers, and one of the 8-bit counter/timers can be reset to its initial value by an external signal.

Interrupts

An eight-level priority interrupt controller can be configured for fully nested or normal interrupt priority. Seven of the eight interrupts service functions on the MUART (counter/timers, UART), and one external interrupt is provided which can be used for a particular function or for chaining interrupt controllers or more MUARTs. The MUART will support 8085 and 8086/88 systems with direct interrupt vectoring, or the MUART can be polled to determine the cause of the interrupt. If additional interrupt control capability is needed, the MUART's interrupt controller can be cascaded into another MUART, into an Intel 8259A Programmable Interrupt Controller, or into the interrupt controller of the iAPX 186/188 High-Integration Microprocessor.

INITIALIZATION

In general the MUART's functions are independent of each other and only the registers and bits associated with a particular function need to be initialized, not the entire chip. The command sequence is arbitrary since every register is directly addressable; however, **Command Byte 1** must be loaded first. To put the device into a fully operational condition, it is necessary to write the following commands:

Command byte 1
Command byte 2
Command byte 3
Mode byte
Port 1 control
Set Interrupts

The modification register may be loaded if required for special applications; normally this operation is not necessary. The MUART should be reset before initialization. (Either a hardware or a software reset will do.)

INTERFACING

This section describes the hardware interface between the 8256 MUART and the 80186 microprocessor. Figure 3 displays the block diagram for this interface. The MUART can be interfaced to many other microprocessors using these basic principles.

In all cases the 8256 will be connected directly to the CPU's multiplexed address/data bus. If latches or data bus buffers are used in a system, the MUART should be on the microprocessor side of the address/data bus. The MUART latches the address internally on the falling edge of ALE. The address consists of Chip Select (CS) and four address lines. For 8-bit microprocessors, AD0–AD3 are the address lines. For 16-bit microprocessors, AD1–AD4 are the address lines; AD0 is used as a second chip select which is active low. Since chip select is internally latched along with the address, it does not have to remain active during the entire instruction cycle. As long as the chip select setup and hold times are met, it can be derived from multiplexed ad-

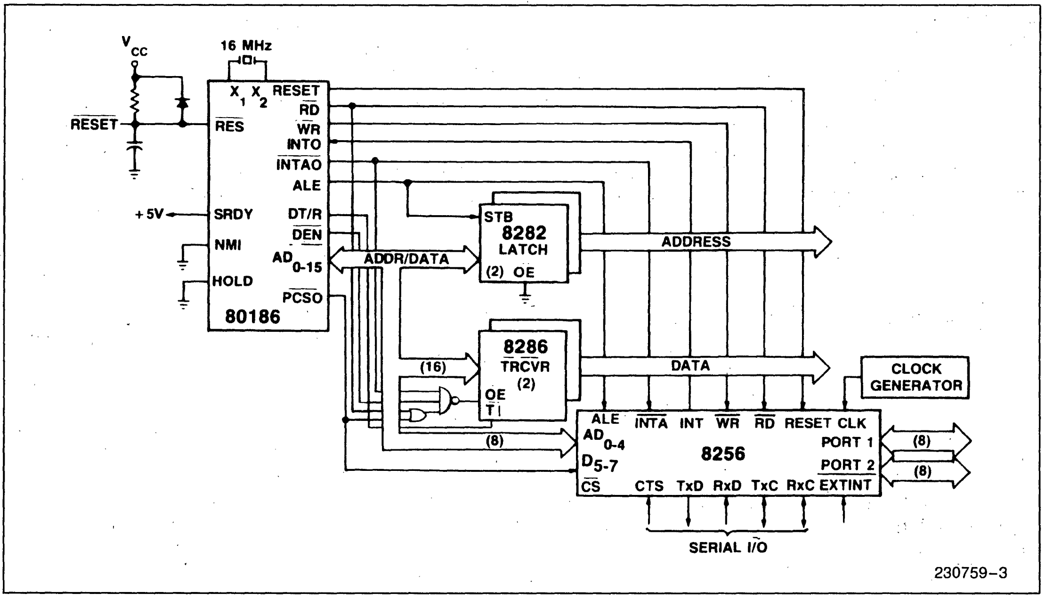


Figure 3. 80186/8256 Interface

dress/data lines or multiplexed address/status lines. When the 8256 is in the 16-bit mode, A0 serves as a second chip select. As a result the MUART's internal registers will all have even addresses since A0 must be zero to select the device. Normally the MUART will be placed on the lower data byte. If the MUART is placed on the upper data byte, the internal registers will be 512 address locations apart and the chip would occupy an 8k word address space.

DESCRIPTION OF THE REGISTERS

The following section will provide a description of the registers and define the bits within the registers where appropriate. Table 2 lists the registers and their addresses.

Command Register 1

L1	L0	S1	S0	BRKI	BITI	8086	FRQ
(0R)				(0W)			

FRQ—TIMER FREQUENCY SELECT

This bit selects between two frequencies for the five timers. If FRQ = 0, the timer input frequency is 16 kHz (62.5 μs). If FRQ = 1, the timer input frequency is 1 kHz (1 ms). The selected clock frequency is shared by all the counter/timers enabled for timing; thus, all timers must run with the same time base.

8086—8086 MODE ENABLE

This bit selects between 8085 mode and 8086/8088 mode. In 8085 mode (8086 = 0), A0 to A3 are used to address the internal registers, and an RSTn instruction is generated in response to the first $\overline{\text{INTA}}$. In 8086 mode (8086 = 1), A1 to A4 are used to address the internal registers, and A0 is used as an extra chip select (A0 must equal zero to be enabled). The response to $\overline{\text{INTA}}$ is for 8086 interrupts where the first $\overline{\text{INTA}}$ is ignored, and an interrupt vector (40H to 47H) is placed on the bus in response to the second $\overline{\text{INTA}}$.

BITI—INTERRUPT ON BIT CHANGE

This bit selects between one of two interrupt sources on Priority Level 1, either Counter/Timer 2 or Port 1 P17 interrupt. When this bit equals 0, Counter/Timer 2 will be mapped into Priority Level 1. If BITI equals 0 and Level 1 interrupt is enabled, a transition from 1 to 0 in Counter/Timer 2 will generate an interrupt request on Level 1. When BITI equals 1, Port 1 P17 external edge triggered interrupt source is mapped into Priority Level 1. In this case if Level 1 is enabled, a low-to-high transition on P17 generates an interrupt request on Level 1.

Table 2. MUART Registers

Read Registers														Write Registers								
									8085 Mode:	AD3	AD2	AD1	AD0									
									8086 Mode:	AD4	AD3	AD2	AD1									
L1	L0	S1	S0	BRKI	BITI	8086	FRQ	Command 1		0	0	0	0	L1	L0	S1	S0	BRKI	BITI	8086	FRQ	Command 1
PEN	EP	C1	C0	B3	B2	B1	B0	Command 2		0	0	0	1	PEN	EP	C1	C0	B3	B2	B1	B0	Command 2
0	RxE	IAE	NIE	0	SBRK	TBRK	0	Command 3		0	0	1	0	SET	RxE	IAE	NIE	END	SBRK	TBRK	RST	Command 3
T35	T24	T5C	CT3	CT2	P2C2	P2C1	P2C0	Mode		0	0	1	1	T35	T24	T5C	CT3	CT2	P2C2	P2C1	P2C0	Mode
P17	P16	P15	P14	P13	P12	P11	P10	Port 1 Control		0	1	0	0	P17	P16	P15	P14	P13	P12	P11	P10	Port 1 Control
L7	L6	L5	L4	L3	L2	L1	L0	Interrupt Enable		0	1	0	1	L7	L6	L5	L4	L3	L2	L1	L0	Set Interrupts
D7	D6	D5	D4	D3	D2	D1	D0	Interrupt Address		0	1	1	0	L7	L6	L5	L4	L3	L2	L1	L0	Reset Interrupts
D7	D6	D5	D4	D3	D2	D1	D0	Receiver Buffer		0	1	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Transmitter Buffer
D7	D6	D5	D4	D3	D2	D1	D0	Port 1		1	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0	Port 1
D7	D6	D5	D4	D3	D2	D1	D0	Port 2		1	0	0	1	D7	D6	D5	D4	D3	D2	D1	D0	Port 2
D7	D6	D5	D4	D3	D2	D1	D0	Timer 1		1	0	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Timer 1
D7	D6	D5	D4	D3	D2	D1	D0	Timer 2		1	0	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Timer 2
D7	D6	D5	D4	D3	D2	D1	D0	Timer 3		1	1	0	0	D7	D6	D5	D4	D3	D2	D1	D0	Timer 3
D7	D6	D5	D4	D3	D2	D1	D0	Timer 4		1	1	0	1	D7	D6	D5	D4	D3	D2	D1	D0	Timer 4
D7	D6	D5	D4	D3	D2	D1	D0	Timer 5		1	1	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Timer 5
INT	RBF	TBE	TRE	BD	PE	OE	FE	Status		1	1	1	1	0	RS4	RS3	RS2	RS1	RS0	TME	DSC	Modification

BRKI—BREAK-IN DETECT ENABLE

If this bit equals 0, Port 1 P16 is a general purpose I/O port. When BRKI equals 1, the Break-In Detect feature is enabled on Port 1 P16. A Break-In condition is present on the transmission line when it is forced to the start bit voltage level by the receiving station. Port 1 P16 must be connected externally to the transmission line in order to detect a Break-In. A Break-In is polled by the MUART during the transmission of the last or only stop bit of a character.

A Break-In Detect is OR-ed with Break Detect in Bit 3 of the Status Register. The distinction can be made through the interrupt controller. If the transmit and receive interrupts are enabled, a Break-In will generate an interrupt on Level 5, the transmit interrupt, while Break will generate an interrupt on Level 4, the receive interrupt.

S0, S1—STOP BIT LENGTH

S1	S0	Stop Bit Length
0	0	1
0	1	1.5
1	0	2
1	1	0.75

The relationship of the number of stop bits and the function of input $\overline{\text{CTS}}$ is discussed in the Pin Description section under "$\overline{\text{CTS}}$".

L0, L1—CHARACTER LENGTH

L1	L0	Character Length
0	0	8
0	1	7
1	0	6
1	1	5

Command Register 2

PEN	EP	C1	C0	B3	B2	B1	B0

(1R) (1W)

Programming bits 0 . . . 3 with values from 3H to FH enables the internal baud rate generator as a common clock source for the transmitter and receiver and determines its divider ratio.

Programming bits 0 . . . 3 with values of 1H or 2H enables input TxC as a common clock source for the transmitter and receiver. The external clock must provide a frequency of either 32× or 64× the baud rate. The data transmission rates range from 0. . .32 Kbaud.

If bits 0 . . . 3 are set to 0, separate clocks must be input to pin RxC for the receiver and pin TxC for the transmitter. Thus, different baud rates can be used for transmission and reception. In this case, prescalers are disabled and the input serial clock frequency must match the baud rate. The input serial clock frequency can range from 0 MHz to 1.024 MHz.

B0, B1, B2, B3—BAUD RATE SELECT

These four bits select the bit clock's source, sampling rate, and serial rate for the internal baud rate generator.

B3	B2	B1	B0	Baud Rate	Sampling Rate
0	0	0	0	$\overline{\text{TxC}}$, $\overline{\text{RxC}}$	1
0	0	0	1	$\overline{\text{TxC}}$/64	64
0	0	1	0	$\overline{\text{TxC}}$/32	32
0	0	1	1	19200	32
0	1	0	0	9600	64
0	1	0	1	4800	64
0	1	1	0	2400	64
0	1	1	1	1200	64
1	0	0	0	600	64
1	0	0	1	300	64
1	0	1	0	200	64
1	0	1	1	150	64
1	1	0	0	110	64
1	1	0	1	100	64
1	1	1	0	75	64
1	1	1	1	50	64

The following table gives an overview of the function of pins TxC and RxC:

Bits 3 to 0 (Hex.)	TxC	RxC
0	Input: 1 × baud rate clock for the transmitter	Input: 1 × baud rate clock for the receiver
1, 2	Input: 32 × or 64 × baud rate for transmitter and receiver	Output: receiver bit clock with a low-to-high transition at data bit sampling time. Otherwise: high level
3 to F	Output: baud rate clock of the transmitter	Output: as above

As an output, RxC outputs a low-to-high transition at sampling time of every data bit of a character. Thus, data can be loaded, e.g., into a shift register externally. The transition occurs only if data bits of a character are present. It does not occur for start, parity, and stop bits (RxC = high).

As an output, TxC outputs the internal baud rate clock of the transmitter. There will be a high-to-low transition at every beginning of a bit.

C0, C1—SYSTEM CLOCK PRESCALER (BITS 4, 5)

Bits 4 and 5 define the system clock prescaler divider ratio. The internal operating frequency of 1.024 MHz is derived from the system clock.

C1	C0	Divider Ratio	Clock at Pin CLK
0	0	5	5.12 MHz
0	1	3	3.072 MHz
1	0	2	2.048 MHz
1	1	1	1.024 MHz

EP—EVEN PARITY (BIT 6)

EP = 0: Odd parity
EP = 1: Even parity

PEN—PARITY ENABLE (BIT 7)

Bit 7 enables parity generation and checking.

PEN = 0: No parity bit
PEN = 1: Even parity bit

The parity bit according to Command Register 2 bit 6 (see above) is inserted between the last data bit of a character and the first or only stop bit. The parity bit is checked during reception. A false parity bit generates an error indication in the Status Register and an Interrupt Request on Level 4.

Command Register 3

SET	RxE	IAE	NIW	END	SBRK	TBRK	RST

(2R) (2W)

Command Register 3 is different from the first two registers because it has a bit set/reset capability. Writing a byte with Bit 7 high sets any bits which were also high. Writing a byte with Bit 7 low resets any bits which were high. If any bit 0–6 is low, no change occurs to that bit. When command Register 3 is read, bits 0, 3, and 7 will always be zero.

RST—RESET

If RST is set, the following events occur:

1. All bits in the Status Register except bits 4 and 5 are cleared, and bits 4 and 5 are set.
2. The Interrupt Enable, Interrupt Request, and Interrupt Service Registers are cleared. Pending requests and indications for interrupts in service will be cancelled. Interrupt signal INT will go low.
3. The receiver and transmitter are reset. The transmitter goes idle (TxD is high), and the receiver enters start bit search mode.
4. If Port 2 is programmed for handshake mode, IBF and OBF are reset high.

RST does *not* alter ports, data registers or command registers, but it halts any operation in progress. RST is automatically cleared.

RST = 0 has no effect. The reset operation triggered by Command Register 3 is a subset of the hardware reset.

TBRK—TRANSMIT BREAK

The transmission data output TxD will be set low as soon as the transmission of the previous character has been finished. It stays low until TBRK is cleared. The state of $\overline{\text{CTS}}$ is of no significance for this operation. As long as break is active, data transfer from the Transmitter Buffer to the Transmitter Register will be inhibited. As soon as TBRK is reset, the break condition will be deactivated and the transmitter will be re-enabled.

SBRK—SINGLE CHARACTER BREAK

This causes the transmitter data to be set low for one character including start bit, data bits, parity bit, and stop bits. SBRK is automatically cleared when time for the last data bit has passed. It will start after the character in progress completes, and will delay the next data transfer from the Transmitter Buffer to the Transmitter Register until TxD returns to an idle (marking) state. If both TBRK and SBRK are set, break will be set as long as TBRK is set, but SBRK will be cleared after one character time of break. If SBRK is set again, it remains set for another character. The user can send a definite number of break characters in this manner by clearing TBRK after setting SBRK for the last character time.

END—END OF INTERRUPT

If fully nested interrupt mode is selected, this bit resets the currently served interrupt level in the Interrupt Service Register. *This command must occur at the end of each interrupt service routine during fully nested interrupt mode.* END is automatically cleared when the Interrupt Service Register (internal) is cleared. END is ignored if nested interrupts are not enabled.

NIE—NESTED INTERRUPT ENABLE

When NIE equals 1, the interrupt controller will operate in the nested interrupt mode. When NIE equals 0, the interrupt controller will operate in the normal interrupt mode. Refer to the "Interrupt controller" section of AP-153 under "Normal Mode" and "Nested Mode" for a detailed description of these operations.

IAE—INTERRUPT ACKNOWLEDGE ENABLE

This bit enables an automatic response to $\overline{\text{INTA}}$. The particular response is determined by the 8086 bit in Command Register 1.

RxE—RECEIVE ENABLE

This bit enables the serial receiver and its associated status bits in the status register. If this bit is reset, the serial receiver will be disabled and the receive status bits will not be updated.

Note that the detection of break characters remains enabled while the receiver is disabled; i.e., Status Register Bit 3 (BD) will be set while the receiver is disabled whenever a break character has been recognized at the receive data input RxD.

SET—BIT SET/RESET

If this bit is high during a write to Command Register 3, then any bit marked by a high will set. If this bit is low, then any bit marked by a high will be cleared.

MODE REGISTER

T35	T24	T5C	CT3	CT2	P2C2	P2C1	P2C0

(3R) (3W)

If test mode is selected, the output from the internal baud rate generator is placed on bit 4 of Port 1 (pin 35).

To achieve this, it is necessary to program bit 4 of Port 1 as an output (Port 1 Control Register Bit P14 = 1), and to program Command Register 2 bits B3–B0 with a value ≥ 3H.

P2C2, P2C1, P2C0—PORT 2 CONTROL

P2C2	P2C1	P2C0	Mode	Direction	
				Upper	Lower
0	0	0	Nibble	Input	Input
0	0	1	Nibble	Input	Output
0	1	0	Nibble	Output	Input
0	1	1	Nibble	Output	Output
1	0	0	Byte Handshake	Input	
1	0	1	Byte Handshake	Output	
1	1	0	*DO NOT USE*		
1	1	1	Test		

NOTE:
If Port 2 is operating in handshake mode, Interrupt Level 7 is not available for Timer 5. Instead it is assigned to Port 2 handshaking.

CT2, CT3—COUNTER/TIMER MODE

Bit 3 and 4 defines the mode of operation of event counter/timers 2 and 3 regardless of its use as a single unit or as a cascaded one.

If CT2 or CT3 are high, then counter/timer 2 or 3 respectively is configured as an event counter on bit 2 or 3 respectively of Port 1 (pins 37 or 36). The event counter decrements the count by one on each low-to-high transition of the external input. If CT2 or CT3 is low, then the respective counter/timer is configured as a timer and the Port 1 pins are used for parallel I/O.

T5C—TIMER 5 CONTROL

If T5C is set, then Timer 5 can be preset and started by an external signal. Writing to the Timer 5 register loads the Timer 5 save register and stops the timer. A high-to-low transition on bit 5 of Port 1 (pin 34) loads the timer with the saved value and starts the timer. The next high-to-low transition on pin 34 retriggers the timer by reloading it with the initial value and continues timing.

Following a hardware reset, the save register is reset to 00H and both clock and trigger inputs are disabled. Transferring an instruction with T5C = 1 enables the trigger input; the save register can now be loaded with an initial value. The first trigger pulse causes the initial value to be loaded from the save register and enables the counter to count down to zero.

When the timer reaches zero it issues an interrupt request, disables its interrupt level and continues counting. A subsequent high-to-low transition on pin 5 resets Timer 5 to its initial value. For another timer interrupt, the Timer 5 interrupt enable bit must be set again.

T35, T24—CASCADE TIMERS

These two bits cascade Timers 3 and 5 or 2 and 4.

Timers 2 and 3 are the lower bytes, while Timers 4 and 5 are the upper bytes. It T5C is set, then both Timers 3 and 5 can be preset and started by an external pulse.

When a high-to-low transition occurs, Timer 5 is preset to its saved value, but Timer 3 is always preset to all ones. If either CT2 or CT3 is set, then the corresponding timer pair is a 16-bit event counter.

A summary of the counter/timer control bits is given in Table 3.

NOTE:
Interrupt levels assigned to single counters are partly not occupied if event counters/timers are cascaded. Level 2 will be vacated if event counters/timers 2 and 4 are cascaded. Likewise, Level 7 will be vacated if event counters/timers 3 and 5 are cascaded.

Single event counters/timers generate an interrupt request on the transition from 01H to 00H, while cascaded ones generate it on the transition from 0001H to 0000H.

Port 1 Control Register

P17	P16	P15	P14	P13	P12	P11	P10

(4R) (4W)

Each bit in the Port 1 Control Register configures the direction of the corresponding pin. If the bit is high, the pin is an output, and if it is low the pin is an input. Every Port 1 pin has another function which is controlled by other registers. If that special function is disabled, the pin functions as a general I/O pin as specified by this register. The special functions for each pin are described below.

Port 10, 11—HANDSHAKE CONTROL

If byte handshake control is enabled for Port 2 by the Mode Register, then Port 10 is programmed as $\overline{\text{STB}}/\overline{\text{ACK}}$ handshake control input, and Port 11 is programmed as $\overline{\text{IBF}}/\overline{\text{OBF}}$ handshake control output.

If byte handshake mode is enabled for output on Port 2 $\overline{\text{OBF}}$ indicates that a character has been loaded into the Port 2 output buffer. When an external

Table 3. Event Counters/Timers Mode of Operation

Event Counter/ Timer	Function	Programming (Mode Word)	Clock Source
1	8-bit timer	—	Internal clock
2	8-bit timer	T24 = 0, CT2 = 0	Internal clock
	8-bit event counter	T24 = 0, CT2 = 1	P12 pin 37
2	8-bit timer	T35 = 0, CT3 = 0	Internal clock
	8-bit event counter	T35 = 0, CT3 = 1	P13 pin 36
4	8-bit timer	T24 = 0	Internal clock
5	8-bit timer, normal mode	T35 = 0, T5C = 0	Internal clock
	8-bit timer, retriggerable mode	T35 = 0, T5C = 1	Internal clock
2 and 4 cascaded	16-bit timer	T24 = 1, CT2 = 0	Internal clock
	16-bit event counter	T24 = 1, CT2 = 1	P12 pin 37
3 and 5 cascaded	16-bit timer, normal mode	T35 = 1, T5C = 0, CT3 = 0	Internal clock
	16-bit event counter, normal mode	T35 = 1, T5C = 0, CT3 = 1	P13 pin 36
	16-bit timer, retriggerable mode	T35 = 1, T5C = 1, CT3 = 0	Internal clock
	16-bit event counter, retriggerable mode	T35 = 1, T5C = 1, CT3 = 1	P13 pin 36

device reads the data, it acknowledges this operation by driving $\overline{ACK}$ low. $\overline{OBF}$ is set low by writing to Port 2 and is reset by $\overline{ACK}$.

If byte handshake mode is enabled for input on Port 2, $\overline{STB}$ is an input. $\overline{IBF}$ is driven low after $\overline{STB}$ goes low. On the rising edge of $\overline{STB}$ the data from Port 2 is latched.

$\overline{IBF}$ is reset high when Port 2 is read.

PORT 12, 13—COUNTER 2, 3 INPUT

If Timer 2 or Timer 3 is programmed as an event counter by the Mode Register, then Port 12 or Port 13 is the counter input for Event Counter 2 or 3, respectively.

PORT 14—BAUD RATE GENERATOR OUTPUT CLOCK

If test mode is enabled by the Mode Register and Command Register 2 baud rate select is greater than 2, then Port 14 is an output from the internal baud rate generator.

P14 in Port 1 control register must be set to 1 for the baud rate generator clock to be output. The baud rate generator clock is 64 × the serial bit rate excet at 19.2 Kbps when it is 32 × the bit rate.

PORT 15—TIMER 5 TRIGGER

If T5C is set in the Mode Register enabling a retriggerable timer, then Port 15 is the input which starts and reloads Timer 5.

A high-to-low transition on P15 (Pin 34) loads the timer with the slave register and starts the timer.

PORT 16—BREAK-IN DETECT

If Break-In Detect is enabled by BRKI in Command Register 1, then this input is used to sense a Break-In. If Port 16 is low while the serial transmitter is sending the last stop bit, then a Break-In condition is signaled.

PORT 17—PORT INTERRUPT SOURCE

If BITI in Command Register 1 is set, then a low-to-high transition on Port 17 generates an interrupt request on Priority Level 1.

Port 17 is edge triggered.

Interrupt Enable Register

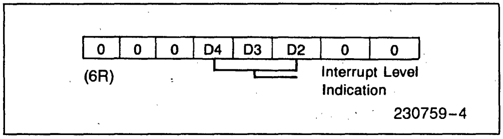
L7	L6	L5	L4	L3	L2	L1	L0

(5R) (5W = enable, 6W = disable)

Interrupts are enabled by writing to the Set Interrupts Register (5W). Interrupts are disabled by writing to the Reset Interrupts Register (6W). Each bit set by the Set Interrupts Register (5W) will enable that level interrupt, and each bit set in the Reset Interrupts Register (6W) will disable that level interrupt. The user can determine which interrupts are enabled by reading the Interrupt Enable Register (5R).

Priority		Source
Highest	L0	Timer 1
	L1	Timer 2 or Port Interrupt
	L2	External Interrupt (EXTINT)
	L3	Timer 3 or Timers 3 & 5
	L4	Receive Interrupt
	L5	Transmitter Interrupt
	L6	Timer 4 or Timers 2 & 4
Lowest	L7	Timer 5 or Port 2 Handshaking

Interrupt Address Register



Reading the interrupt address register transfers an identifier for the currently requested interrupt level on the system data bus. This identifier is the number of the interrupt level multiplied by 4. It can be used by the CPU as an offset address for interrupt handling. Reading the interrupt address register has the same effect as a hardware interrupt acknowledge $\overline{INTA}$; it clears the interrupt request pin (INT) and indicates an interrupt acknowledgement to the interrupt controller.

Receiver and Transmitter Buffer

D7	D6	D5	D4	D3	D2	D1	D0

(7R) (7W)

Both the receiver and transmitter in the MUART are double buffered. This means that the transmitter and receiver have a shift register and a buffer register. The buffer registers are directly addressable by reading or writing to register seven. After the receiver buffer is full, the RBF bit in the status register is set.

Reading the receive buffer clears the RBF status bit. The transmit buffer should be written to only if the TBE bit in the status register is set. Bytes written to the transmit buffer are held there until the transmit shift register is empty, assuming $\overline{CTS}$ is low. If the transmit buffer and shift register are empty, writing to the transmit buffer immediately transfers the byte to the transmit shift register. If a serial character length is less than 8 bits, the unused most significant bits are set to zero when reading the receive buffer, and are ignored when writing to the transmit buffer.

Port 1

D7	D6	D5	D4	D3	D2	D1	D0

(8R) (8W)

Writing to Port 1 sets the data in the Port 1 output latch. Writing to an input pin does not affect the pin, but the data is stored and will be output if the direction of the pin is changed later. If the pin is used as a control signal, the pin will not be affected, but the data is stored. Reading Port 1 transfers the data in Port 1 onto the data bus.

Port 2

D7	D6	D5	D4	D3	D2	D1	D0

(9R) (9W)

Writing to Port 2 sets the data in the Port 2 output latch. Writing to an input pin does not affect the pin, but it does store the data in the latch. Reading Port 2 puts the input pins onto the bus or the contents of the output latch for output pins.

Timer 1-5

D7	D6	D5	D4	D3	D2	D1	D0

($0A_{16}$–OE_{16}R) ($0A_{16}$–OE_{16}W)

Reading Timer N puts the contents of the timer onto the data bus. If the counter changes while $\overline{RD}$ is low, the value on the data bus will not change. If two timers are cascaded, reading the high-order byte will cause the low-order byte to be latched. Reading the low-order byte will unlatch them both. Writing to either timer or decascading them also clears the latch condition. Writing to a timer sets the starting value of that timer. If two timers are cascaded, writing to the high-order byte presets the low-order byte to all ones. Loading only the high-order byte with a value of X leads to a count of X *256 + 255. Timers count down continuously. If the interrupt is enabled, it occurs when the counter changes from 1 to 0.

The timer/counter interrupts are automatically disabled when the interrupt request is generated.

Status Register

INT	RBF	TBE	TRE	BD	PE	OE	FE

($0F_{16}$R)

Reading the status register gates its contents onto the data bus. It holds the operational status of the serial interface as well as the status of the interrupt in INT. The status register can be read at any time. The flags are stable and well defined at all instants.

FE—FRAMING ERROR, TRANSMISSION MODE

Bit 0 can be used in two modes. Normally, FE indicates framing error which can be changed to transmission mode indication by setting the TME bit in the modification register.

If transmission mode is disabled (in Modification Register), then FE indicates a framing error. A framing error is detected during the *first* stop bit. The error is reset by reading the Status Register or by a chip reset. A framing error does not inhibit the loading of the Receiver Buffer. If RxD remains low, the receiver will assemble the next character. The false stop bit is treated as the next start bit, and no high-to-low transition on RxD is required to synchronize the receiver.

When the TME bit in the Modification Register is set, FE is used to indicate that the transmitter was active during the reception of a character, thus indicating that the character received was transmitted by its own transmitter. FE is reset when the transmitter is not active during the reception of character. Reading the status register will not reset the FE bit in the transmission mode.

OE—OVERRUN ERROR

If the user does not read the character in the Receiver Buffer before the next character is received and transferred to this register, then the OE bit is set. The OE flag is set during the reception of the first stop bit and is cleared when the Status Register is read or when a hardware or software reset occurs. The first character received in this case will be lost.

PE—PARITY ERROR

This bit indicates a parity error has occurred during the reception of a character. A parity error is present if value of the parity bit in the received character is different from the one expected according to command word 2 bits 6 EP. The parity bit is expected and checked only if it is enabled by command word 2 bit 7 PEN.

A parity error is set during the first stop bit and is reset by reading the Status Register or by a chip reset.

BD—BREAK/BREAK-IN

The BD bit flags whether a break character has been received, or a Break-In condition exists on the transmission line. Command Register 1 Bit 3 (BRKI) enables the Break-In Detect function.

Whenever a break character has been received, Status Register Bit 3 will be set and in addition an interrupt request on Level 4 is generated. The receiver will be idled. It will be started again with the next high-to-low transition at pin RxD.

The break character received will not be loaded into the receiver buffer register.

If Break-In Detection is enabled and a Break-In condition occurs, Status Register Bit 3 will be set and in addition an interrupt request on Level 5 is generated.

The BD status bit will be reset on reading the status register or on a hardware or software reset. For more information on Break/Break-In, refer to the "Serial Asynchronous Communication" section of AP-153 under "Receive Break Detect" and "Break-In Detect."

TRE—TRANSMIT REGISTER EMPTY

When TRE is set the transmit register is empty and an interrupt request is generated on Level 5 if enabled. When TRE equals 0 the transmit register is in the process of sending data. TRE is set by a chip reset and when the last stop bit has left the transmitter. It is reset when a character is loaded into the Transmitter Register. If $\overline{CTS}$ is low, the Transmitter Register will be loaded during the transmission of the start bit. If $\overline{CTS}$ is high at the end of a character, TRE will remain high and no character will be loaded into the Transmitter Register until $\overline{CTS}$ goes low. If the transmitter was inactive before a character is loaded into the Transmitter Buffer, the Transmitter Register will be empty temporarily while the buffer is full. However, the data in the buffer will be transferred to the transmitter register immediately and TRE will be cleared while TBE is set.

TBE—TRANSMITTER BUFFER EMPTY

TBE indicates the Transmitter Buffer is empty and is ready to accept a character. TBE is set by a chip reset or the transfer of data to the Transmitter Register, and is cleared when a character is written to the transmitter buffer. When TBE is set, an interrupt request is generated on Level 5 if enabled.

RBF—RECEIVER BUFFER FULL

RBF is set when the Receiver Buffer has been loaded with a new character during the sampling of the first stop bit. RBF is cleared by reading the receiver buffer or by a chip reset.

INT—INTERRUPT PENDING

The INT bit reflects the state of the INT Pin (Pin 15) and indicates an interrupt is pending. It is reset by $\overline{INTA}$ or by reading the Interrupt Address Register if only one interrupt is pending and by a chip reset.

FE, OE, PE, RBF, and Break Detect all generate a Level 4 interrupt when the receiver samples the first stop bit. TRE, TBE, and Break-In Detect generate a Level 5 interrupt. TRE generates an interrupt when TBE is set and the Transmitter Register finished transmitting. The Break-In Detect interrupt is issued at the same time as TBE or TRE.

MODIFICATION REGISTER

0	RS4	RS3	RS2	RS1	RS0	TME	DSC

($0F_{16}$W)

DSC—DISABLE START BIT CHECK

DSC disables the receivers start bit check. In this state the receiver will not be reset if RxD is not low at the center of the start bit.

TME—TRANSMISSION MODE ENABLE

TME enables transmission mode and disables framing error detection. For information on transmission mode see the description of the framing error bit in the status register.

RS0, RS1, RS2, RS3, RS4—RECEIVER SAMPLE TIME

The number in RSn alters when the receiver samples RxD. The receiver sample time can be modified only if the receiver is *not* clocked by RxC.

NOTE:
The modification register cannot be read. Reading from address 0FH, 8086: 1EH gates the contents of the status register onto the data bus.

A hardware reset (reset, Pin 12) resets all modification register bits to 0, i.e.:

- The start bit check is enabled.
- Status Register Bit 0 (FE) indicates framing error.
- The sampling time of the serial receiver is the bit center.

A software reset (Command Word 3, RST) does not affect the modification register.

Hardware Reset

A reset signal on pin RESET (HIGH level) forces the device 8256 into a well-defined initial state. This state is characterized as follows:

1) Command registers 1, 2 and 3, mode register, Port 1 control register, and modification register are reset. Thus, all bits of the parallel interface are set to be intputs and event counters/timers are configured as independent 8-bit timers.
2) Status register bits are reset with the exception of bits 4 and 5. Bits 4 and 5 are set indicating that both transmitter register and transmitter buffer register are empty.
3) The interrupt mask, interrupt request, and interrupt service register bits are reset and disable all requests. As a consequence, interrupt signal INT IS INACTIVE (LOW).
4) The transmit data output is set to the marking state (HIGH) and the receiver section is disabled until it is enabled by Command Register 3 Bit 6.
5) The start bit will be checked at sampling time. The receiver will return to start bit search mode if input RxD is not LOW at this time.
6) Status Register Bit 0 implies framing error.
7) The receiver samples input RxD at bit center.

Reset has no effect on the contents of receiver buffer register, transmitter buffer register, the intermediate latches of parallel ports, and event counters/timers, respectively.

RS4	RS3	RS2	RS1	RS0	Point of Time Between Start of Bit and End of Bit Measured in Steps of 1/32 Bit Length
0	1	1	1	1	1 (Start of Bit)
0	1	1	1	0	2
0	1	1	0	1	3
0	1	1	0	0	4
0	1	0	1	1	5
0	1	0	1	0	6
0	1	0	0	1	7
0	1	0	0	0	8
0	0	1	1	1	9
0	0	1	1	0	10
0	0	1	0	1	11
0	0	1	0	0	12
0	0	0	1	1	13
0	0	0	1	0	14
0	0	0	0	1	15
0	0	0	0	0	16 (Bit center)
1	1	1	1	1	17
1	1	1	1	0	18
1	1	1	0	1	19
1	1	1	0	0	20
1	1	0	1	1	21
1	1	0	1	0	22
1	1	0	0	1	23
1	1	0	0	0	24
1	0	1	1	1	25
1	0	1	1	0	26
1	0	1	0	1	27
1	0	1	0	0	28
1	0	0	1	1	29
1	0	0	1	0	30
1	0	0	0	1	31
1	0	0	0	0	32 (End of Bit)

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.*

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS $T_A = 0°C$ to $70°C$, $V_{CC} = +5.0V \pm 10\%$

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} + 0.5	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_{IL}	Input Leakage		10 −10	μA μA	$V_{IN} = V_{CC}$ V_{IN} = 0V
I_{LO}	Output Leakage		10 −10	μA μA	$V_{OUT} = V_{CC}$ V_{OUT} = 0.45V
I_{CC}	V_{CC} Supply Current		160	mA	
C_{IN}	Input Capacitance		10	pF	f_c = 1 MHz(1)
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured Pins Returned to V_{SS}(1)

NOTE:
1. Sampled, not 100% tested. T_A = 25°C.

A.C. CHARACTERISTICS $T_A = 0°C$ to $70°C$, $V_{CC} = +5.0V \pm 10\%$, GND = 0V

Symbol	Parameter	8256AH		Units
		Min	Max	
BUS PARAMETERS				
t_{LL}	ALE Pulse Width	50		ns
t_{CSL}	$\overline{CS}$ to ALE Setup Time	0		ns
t_{AL}	Address to ALE Setup Time	20		ns
t_{LA}	Address Hold Time after ALE	25		ns
t_{LC}	ALE to $\overline{RD}/\overline{WR}$	20		ns
t_{CC}	$\overline{RD}$, $\overline{WR}$, $\overline{INTA}$ Pulse Width	200		ns
t_{RD}	Data Valid from $\overline{RD}$(1)		120	ns

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = +5.0V ±10%, GND = 0V (Continued)

Symbol	Parameter	8256AH		Units
		Min	Max	
BUS PARAMETERS (Continued)				
t_{DF}	Data Float after $\overline{\text{RD}}$ (2)		50	ns
t_{DW}	Data Valid to $\overline{\text{WR}}$	150		ns
t_{WD}	Data Valid after $\overline{\text{WR}}$	50		ns
t_{CL}	$\overline{\text{RD}}/\overline{\text{WR}}$ Control to Latch Enable	25		ns
t_{LDR}	ALE to Data Valid		150	ns
t_{RST}	Reset Pulse Width	300		ns
t_{RV}	Recovery Time between $\overline{\text{RD}}/\overline{\text{WR}}$	500		ns
TIMER/COUNTER PARAMETERS				
t_{CPI}	Counter Input Cycle Time (P12, P13)	2.2		μs
t_{CPWH}	Counter Input Pulse Width High	1.1		μs
t_{CPWL}	Counter Input Pulse Width Low	1.1		μs
t_{TPI}	Counter Input ↑ to INT ↑ at Terminal Count		2.75	μs
t_{TIH}	LOAD Pulse High Time Counter 5	1.1		μs
t_{TIL}	LOAD Pulse Low Time Counter 5	1.1		μs
t_{PP}	Counter 5 Load before Next Clock Pulse on P13	1.1		μs
t_{CR}	External Count Clock ↑ to $\overline{\text{RD}}$ ↓ to Ensure Clock is Reflected in Count	2.2		μs
t_{RC}	$\overline{\text{RD}}$ ↑ to External Count Clock ↑ to Ensure Clock is not Reflected in Count	0		ns
t_{CW}	External Count Clock ↑ to $\overline{\text{WR}}$ ↑ to Ensure Count Written is Not Decremented	2.2		μs
t_{WC}	$\overline{\text{WR}}$ ↑ to External Count Clock to Ensure Count Written is Decremented	0		ns
INTERRUPT PARAMETERS				
t_{DEX}	EXTINT ↑ to INT ↑		200	ns
t_{DPI}	Interrupt Request on P17 ↑ to INT ↑		$2t_{CY}+500$	ns
t_{PI}	Pulse Width of Interrupt Request on P17	$t_{CY}+100$		ns
t_{HEA}	$\overline{\text{INTA}}$ ↑ or $\overline{\text{RD}}$ ↑ to EXTINT ↓	30		ns
t_{HIA}	$\overline{\text{INTA}}$ ↑ or $\overline{\text{RD}}$ ↑ to INT ↓		300	μs
SERIAL INTERFACE AND CLOCK PARAMETERS				
t_{CY}	Clock Period	195	1000	ns
t_{CLKH}	Clock High Pulse Width	65		ns
t_{CLKL}	Clock Low Pulse Width	65		ns
t_R	Clock Rise Time		20	ns
t_F	Clock Fall Time		20	ns

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = +5.0V ±10%, GND = 0V (Continued)

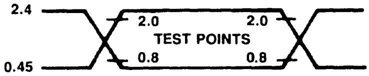
Symbol	Parameter	8256AH		Units
		Min	Max	
SERIAL INTERFACE AND CLOCK PARAMETERS (Continued)				
t_{SCY}	Serial Clock Period (4)	975		ns
t_{SPD}	Serial Clock High (4)	350		ns
t_{SPW}	Serial Clock Low (4)	350		ns
t_{STD}	Internal Status Update Delay from Center of Stop Bit (5)		300	ns
t_{DTX}	$\overline{\text{TxC}}$ to TxD Data Valid		300	ns
t_{IRBF}	INT Delay from Center of First Stop Bit		$2t_{CY}+500$	ns
t_{ITBE}	INT Delay from Falling Edge of Transmit Clock at End of Start Bit		$2t_{CY}+500$	ns
t_{CTS}	Pulse Width for Single Character Transmission	(6)		
PARALLEL I/O PORT PARAMETERS				
t_{WP}	$\overline{\text{WR}}$ ↑ to P1/P2 Data Valid		0	ns
t_{PR}	P1/P2 Data Stable before $\overline{\text{RD}}$ ↓ (7)	300		ns
t_{RP}	P1/P2 Data Hold Time	50		ns
t_{AK}	$\overline{\text{ACK}}$ Pulse Width	150		ns
t_{ST}	Strobe Pulse Width	t_{SIB}		ns
t_{PS}	Data Setup to $\overline{\text{STB}}$ ↑	50		ns
t_{PH}	Data Hold after $\overline{\text{STB}}$ ↑	50		ns
t_{WOB}	$\overline{\text{WR}}$ ↑ to $\overline{\text{OBF}}$ ↑		250	ns
t_{AOB}	$\overline{\text{ACK}}$ ↓ to $\overline{\text{OBF}}$ ↓		250	ns
t_{SIB}	$\overline{\text{STB}}$ ↓ to $\overline{\text{IBF}}$ ↓		250	ns
t_{RI}	$\overline{\text{RD}}$ ↑ to $\overline{\text{IBF}}$ ↑		250	ns
t_{SIT}	$\overline{\text{STB}}$ ↑ to $\overline{\text{INT}}$ ↑		$2t_{CY}$ $+500$	ns
t_{AIT}	$\overline{\text{ACK}}$ ↑ to $\overline{\text{INT}}$ ↑		$2t_{CY}$ $+500$	ns
t_{AED}	$\overline{\text{OBF}}$ ↓ to $\overline{\text{ACK}}$ ↓ Delay	0		ns

NOTES:

1. C_L = pF all outputs.
2. Measured from logic "one" or "zero" to 1.5V at C_L = 150 pF.
3. P12, P13 are external clock inputs.
4. Note that RxC may be used as an input only in 1× mode, otherwise it will be an output.
5. The center of the Stop Bit will be the receiver sample time, as programmed by the modification register.
6. 1/16th bit length for 32×, 64×; 100 ns for 1×.
7. To ensure t_{RD} spec is met.

WAVEFORMS

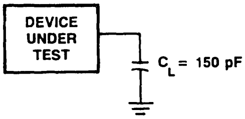
A.C. TESTING INPUT, OUTPUT WAVEFORM



230759-5

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".

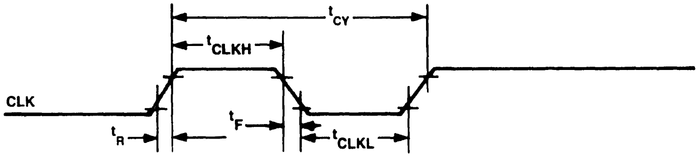
A.C. TESTING LOAD CIRCUIT



230759-6

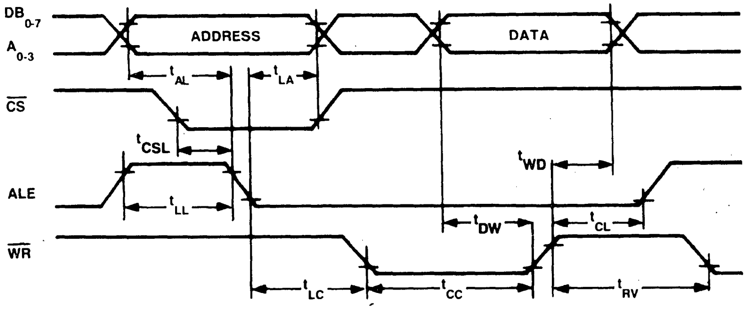
C_L = 150 pF
C_L Includes Jig Capacitance

SYSTEM CLOCK



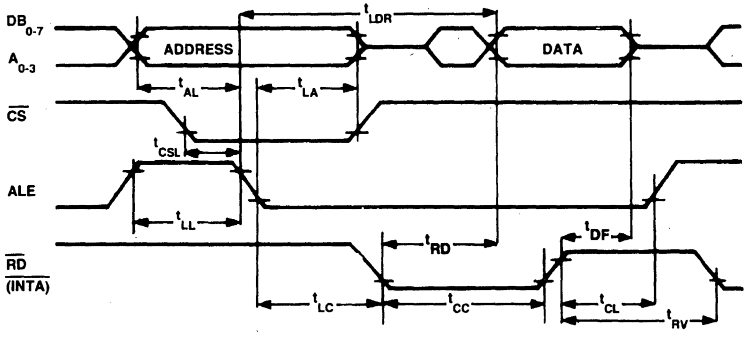
230759-7

WRITE CYCLE



230759-8

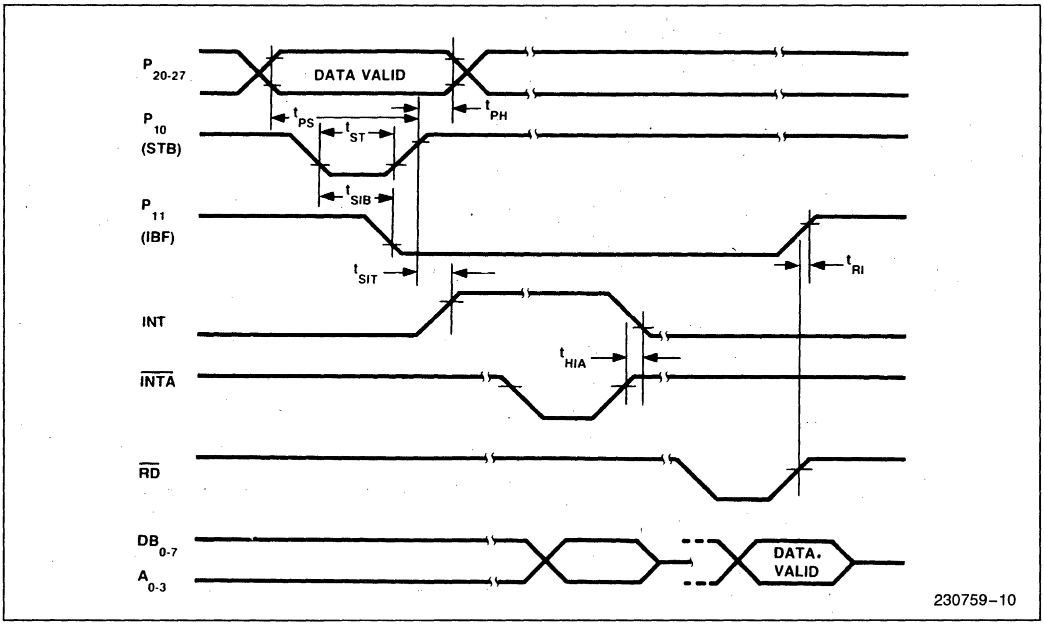
READ CYCLE



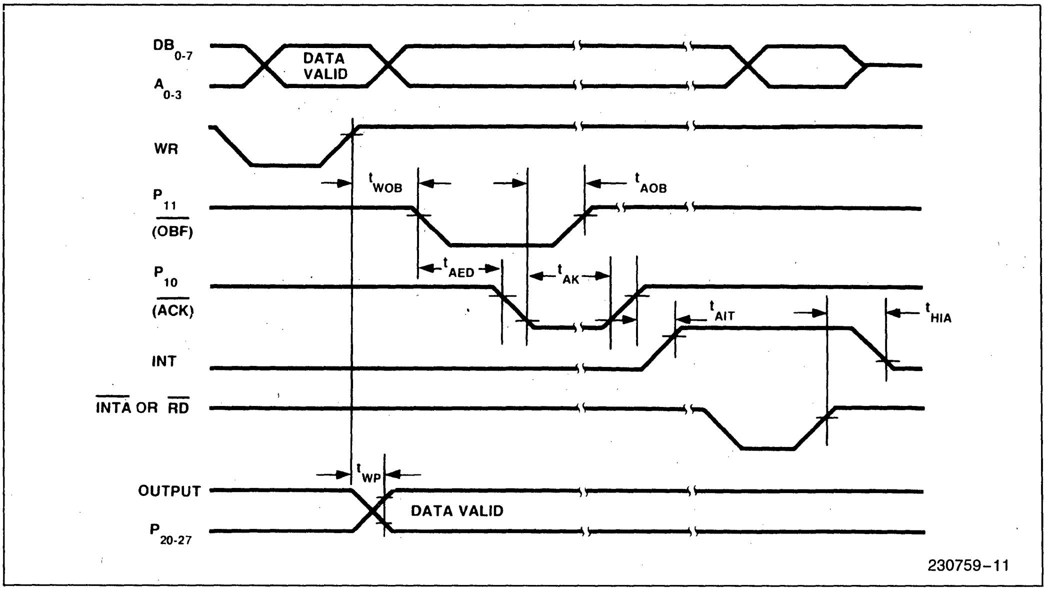
230759-9

WAVEFORMS (Continued)

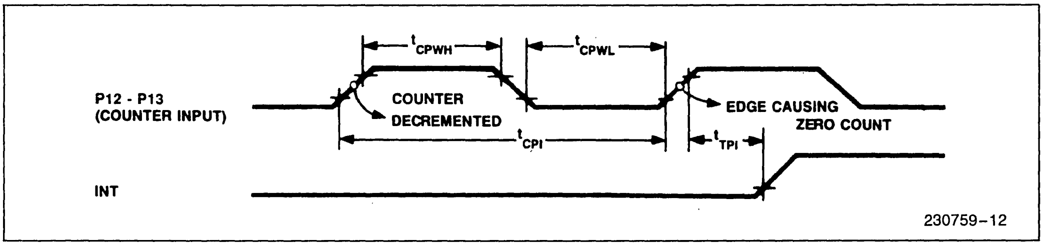
PARALLEL PORT HANDSHAKING—INPUT MODE



PARALLEL PORT HANDSHAKING—OUTPUT MODE

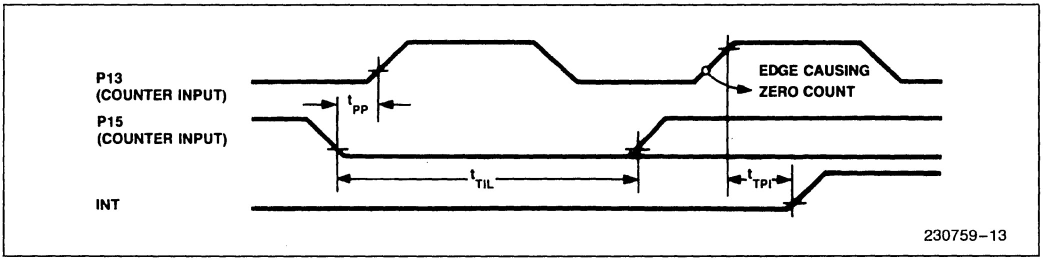


COUNT PULSE TIMINGS



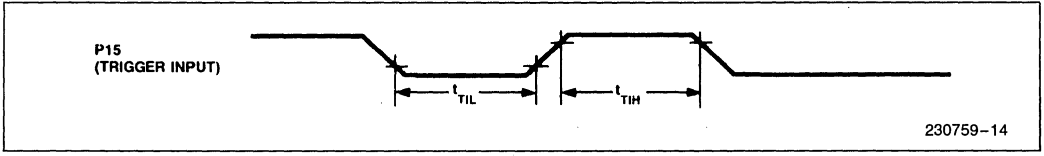
230759-12

LOADING TIMER (OR CASCADED COUNTER/TIMER 3 AND 5)



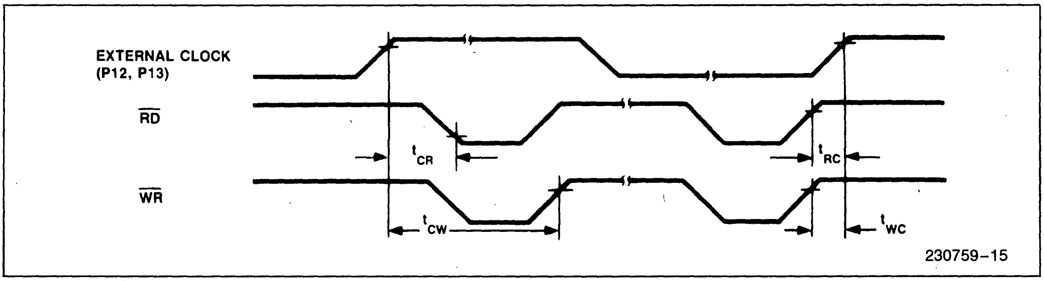
230759-13

TRIGGER PULSE FOR TIMER 5 (CASCADED EVENT COUNTER/TIMER 3 AND 5)



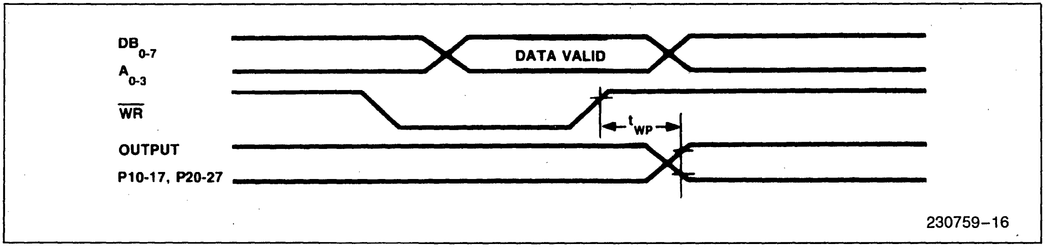
230759-14

COUNTER TIMER TIMING



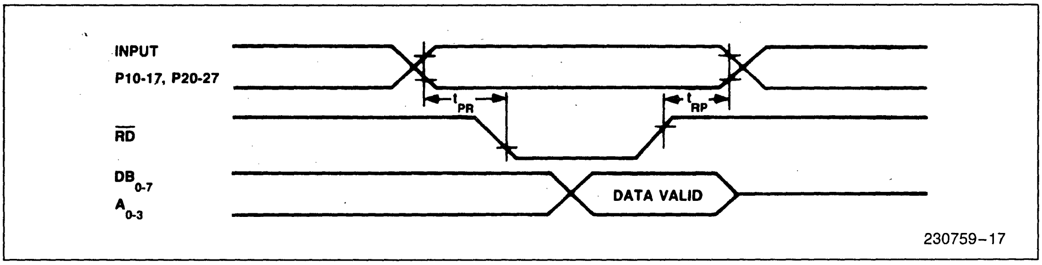
230759-15

OUTPUT FROM PORT 1 AND PORT 2

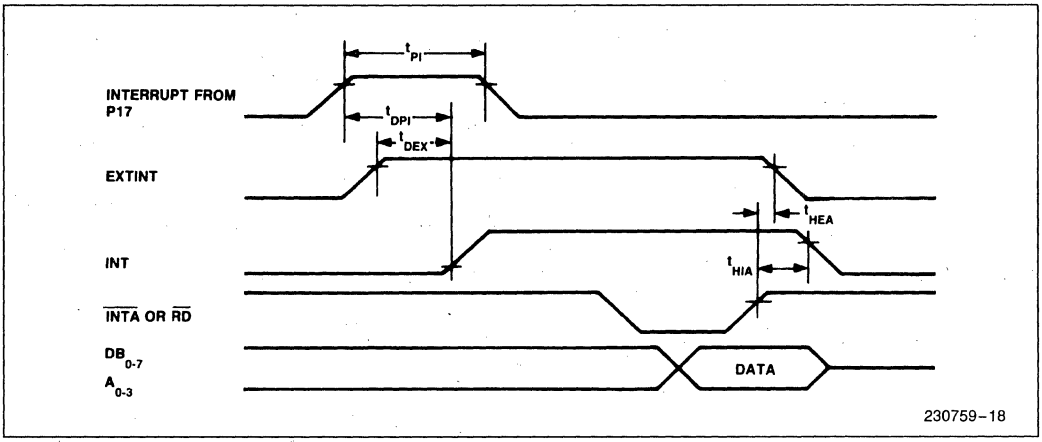


230759-16

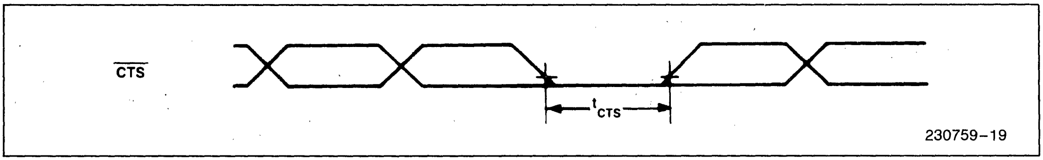
INPUT FROM PORT 1 AND PORT 2



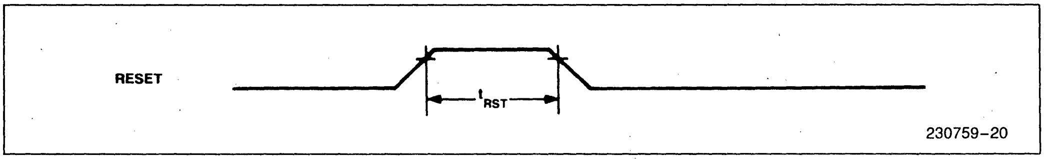
INTERRUPT TIMING



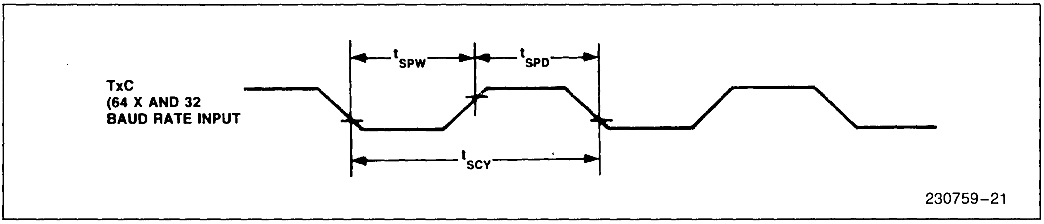
$\overline{CTS}$ FOR SINGLE CHARACTER TRANSMISSION



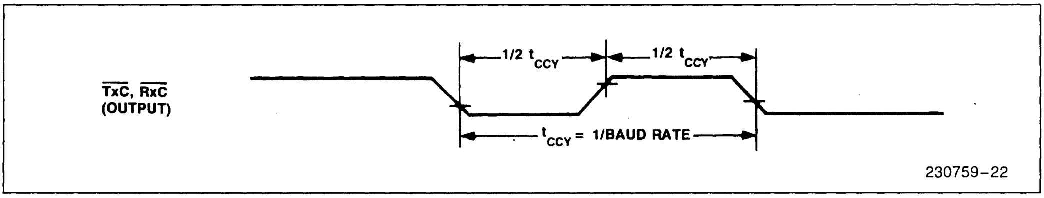
RESET TIMING



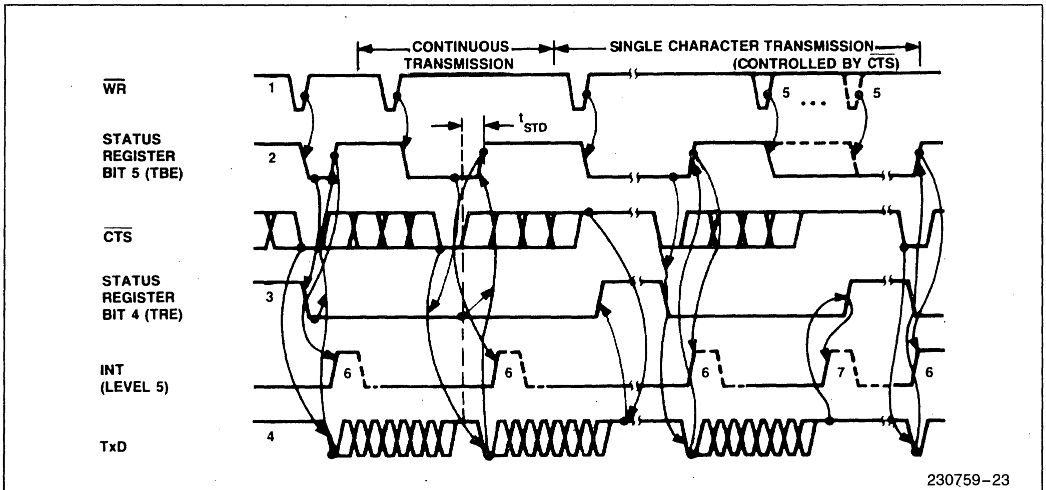
EXTERNAL BAUD RATE CLOCK FOR SERIAL INTERFACE



TRANSMITTER AND RECEIVER CLOCK FROM INTERNAL CLOCK SOURCE



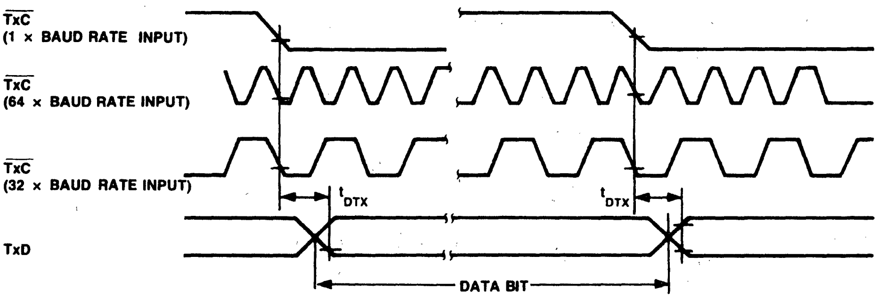
TRANSMISSION OF CHARACTERS ON SERIAL INTERFACE


CONTINUOUS TRANSMISSION
SINGLE CHARACTER TRANSMISSION (CONTROLLED BY $\overline{CTS}$)
$\overline{WR}$
STATUS REGISTER BIT 5 (TBE)
$\overline{CTS}$
STATUS REGISTER BIT 4 (TRE)
INT (LEVEL 5)
TxD
1
2
3
4
5 ... 5
6
6
6
7
6
t_{STD}
230759-23

NOTES:
1. Load transmitter buffer register.
2. Transmitter buffer register is empty.
3. Transmitter register is empty.
4. Character format for this example: 7 Data Bits with Parity Bit and 2 Stop Bits.
5. Loading of transmitter buffer register must be complete before $\overline{CTS}$ goes low.
6. Interrupt due to transmitter buffer register empty.
7. Interrupt due to transmitter register empty.

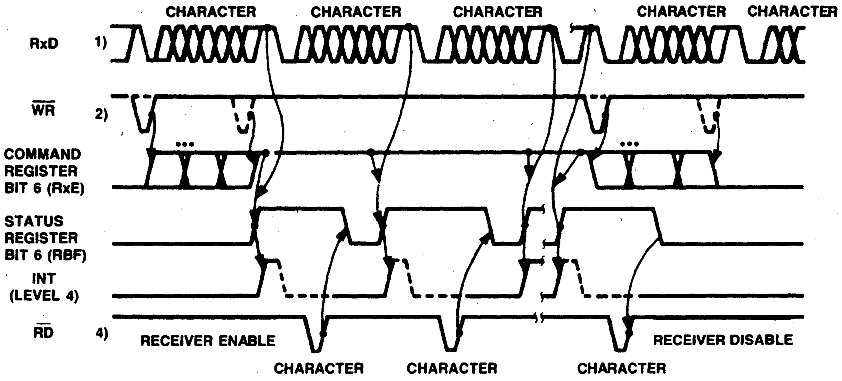
No status bits are altered when $\overline{RD}$ is active.

DATA BIT OUTPUT ON SERIAL INTERFACE



230759-24

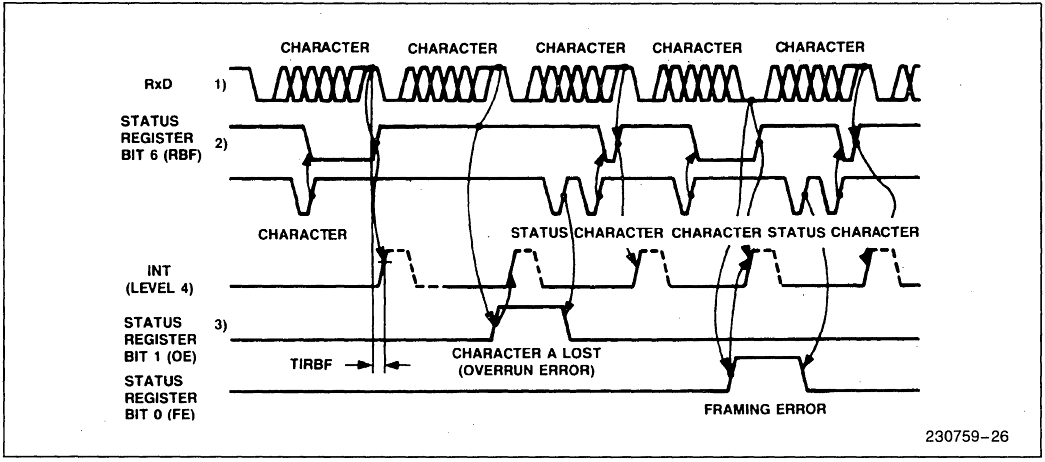
CONTINUOUS RECEPTION OF CHARACTERS ON SERIAL INTERFACE WITHOUT ERROR CONDITION



230759-25

NOTES:
1. Character format for this example: 6 data bits with parity bit and one stop bit.
2. Set or reset bit 6 of command register 3 (enable receiver).
3. Receiver buffer located.
4. Read receiver buffer register.

ERROR CONDITIONS DURING RECEPTION OF CHARACTERS ON THE SERIAL INTERFACE



NOTES:

1. Character format for this example: 6 data bits without parity and one stop bit.
2. Receiver buffer register loaded.
3. Overrun error.
4. Framing error.
5. Interrupt from receiver buffer register loading.
6. Interrupt from overrun error.
7. Interrupt from framing error and loading receiver buffer register.

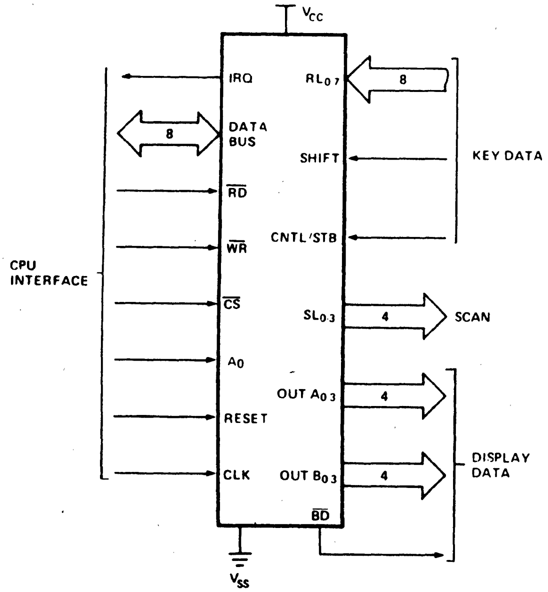
No status bits are altered when $\overline{RD}$ is active.

8279/8279-5
PROGRAMMABLE KEYBOARD/DISPLAY INTERFACE

- **Simultaneous Keyboard Display Operations**
- **Scanned Keyboard Mode**
- **Scanned Sensor Mode**
- **Strobed Input Entry Mode**
- **8-Character Keyboard FIFO**
- **2-Key Lockout or N-Key Rollover with Contact Debounce**
- **Dual 8- or 16-Numerical Display**
- **Single 16-Character Display**
- **Right or Left Entry 16-Byte Display RAM**
- **Mode Programmable from CPU**
- **Programmable Scan Timing**
- **Interrupt Output on Key Entry**
- **Available in EXPRESS**
  — Standard Temperature Range
  — Extended Temperature Range

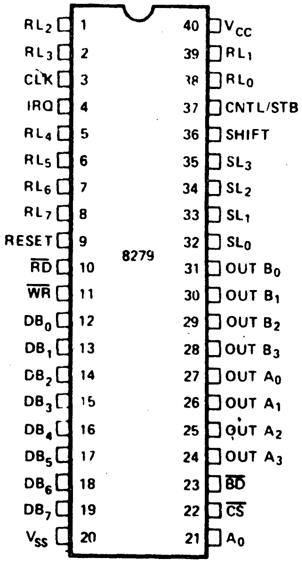
The Intel® 8279 is a general purpose programmable keyboard and display I/O interface device designed for use with Intel® microprocessors. The keyboard portion can provide a scanned interface to a 64-contact key matrix. The keyboard portion will also interface to an array of sensors or a strobed interface keyboard, such as the hall effect and ferrite variety. Key depressions can be 2-key lockout or N-key rollover. Keyboard entries are debounced and strobed in an 8-character FIFO. If more than 8 characters are entered, overrun status is set. Key entries set the interrupt output line to the CPU.

The display portion provides a scanned display interface for LED, incandescent, and other popular display technologies. Both numeric and alphanumeric segment displays may be used as well as simple indicators. The 8279 has 16x8 display RAM which can be organized into dual 16x4. The RAM can be loaded or interrogated by the CPU. Both right entry, calculator and left entry typewriter display formats are possible. Both read and write of the display RAM can be done with auto-increment of the display RAM address.



290123-1

Figure 1. Logic Symbol



290123-2

Figure 2. Pin Configuration

September 1987
Order Number: 290123-002

HARDWARE DESCRIPTION

The 8279 is packaged in a 40 pin DIP. The following is a functional description of each pin.

Table 1. Pin Description

Symbol	Pin No.	Name and Function
DB_0–DB_7	19–12	**BI-DIRECTIONAL DATA BUS:** All data and commands between the CPU and the 8279 are transmitted on these lines.
CLK	3	**CLOCK:** Clock from system used to generate internal timing.
RESET	9	**RESET:** A high signal on this pin resets the 8279. After being reset the 8279 is placed in the following mode: 1) 16 8-bit character display—left entry. 2) Encoded scan keyboard—2 key lockout. Along with this the program clock prescaler is set to 31.
CS	22	**CHIP SELECT:** A low on this pin enables the interface functions to receive or transmit.
A_0	21	**BUFFER ADDRESS:** A high on this line indicates the signals in or out are interpreted as a command or status. A low indicates that they are data.
$\overline{RD}$, $\overline{WR}$	10–11	**INPUT/OUTPUT READ AND WRITE:** These signals enable the data buffers to either send data to the external bus or receive it from the external bus.
IRQ	4	**INTERRUPT REQUEST:** In a keyboard mode, the interrupt line is high when there is data in the FIFO/Sensor RAM. The interrupt line goes low with each FIFO/Sensor RAM read and returns high if there is still information in the RAM. In a sensor mode, the interrupt line goes high whenever a change in a sensor is detected.
V_{SS}, V_{CC}	20, 40	**GROUND AND POWER SUPPLY PINS.**
SL_0–SL_3	32–35	**SCAN LINES:** Scan lines which are used to scan the key switch or sensor matrix and the display digits. These lines can be either encoded (1 of 16) or decoded (1 of 4).
RL_0–RL_7	38, 39, 1, 2, 5–8	**RETURN LINE:** Return line inputs which are connected to the scan lines through the keys or sensor switches. They have active internal pullups to keep them high until a switch closure pulls one low. They also serve as an 8-bit input in the Strobed Input mode.
SHIFT	36	**SHIFT:** The shift input status is stored along with the key position on key closure in the Scanned Keyboard modes. It has an active internal pullup to keep it high until a switch closure pulls it low.
CNTL/STB	37	**CONTROL/STROBED INPUT MODE:** For keyboard modes this line is used as a control input and stored like status on a key closure. The line is also the strobe line that enters the data into the FIFO in the Strobed Input mode. (Rising Edge). It has an active internal pullup to keep it high until a switch closure pulls it low.
OUT A_0–OUT A_3 OUT B_0–OUT B_3	27-24 31–28	**OUTPUTS:** These two ports are the outputs for the 16 x 4 display refresh registers. The data from these outputs is synchronized to the scan lines (SL_0–SL_3) for multiplexed digit displays. The two 4 bit ports may be blanked independently. These two ports may also be considered as one 8-bit port.
$\overline{BD}$	23	**BLANK DISPLAY:** This output is used to blank the display during digit switching or by a display blanking command.

FUNCTIONAL DESCRIPTION

Since data input and display are an integral part of many microprocessor designs, the system designer needs an interface that can control these functions without placing a large load on the CPU. The 8279 provides this function for 8-bit microprocessors.

The 8279 has two sections: keyboard and display. The keyboard section can interface to regular typewriter style keyboards or random toggle or thumb switches. The display section drives alphanumeric displays or a bank of indicator lights. Thus the CPU is relieved from scanning the keyboard or refreshing the display.

The 8279 is designed to directly connect to the microprocessor bus. The CPU can program all operating modes for the 8279. These modes include:

Input Modes

- Scanned Keyboard—with encoded (8 x 8 key keyboard) or decoded (4 x 8 key keyboard) scan lines. A key depression generates a 6-bit encoding of key position. Position and shift and control status are stored in the FIFO. Keys are automatically debounced with 2-key lockout or N-key rollover.
- Scanned Sensor Matrix—with encoded (8 x 8 matrix switches) or decoded (4 x 8 matrix switches) scan lines. Key status (open or closed) stored in RAM addressable by CPU.
- Strobed Input—Data on return lines during control line strobe is transferred to FIFO.

Output Modes

- 8 or 16 character multiplexed displays that can be organized as dual 4-bit or single 8-bit ($B_0 = D_0$, $A_3 = D_7$).
- Right entry or left entry display formats.

Other features of the 8279 include:

- Mode programming from the CPU.
- Clock Prescaler
- Interrupt output to signal CPU when there is keyboard or sensor data available.
- An 8 byte FIFO to store keyboard information.
- 16 byte internal Display RAM for display refresh. This RAM can also be read by the CPU.

PRINCIPLES OF OPERATION

The following is a description of the major elements of the 8279 Programmable Keyboard/Display interface device. Refer to the block diagram in Figure 3.

I/O Control and Data Buffers

The I/O control section uses the $\overline{CS}$, A_0, $\overline{RD}$ and $\overline{WR}$ lines to control data flow to and from the various internal registers and buffers. All data flow to and from the 8279 is enabled by $\overline{CS}$. The character of the information, given or desired by the CPU, is identified by A_0. A logic one means the information is a command or status. A logic zero means the information is data. $\overline{RD}$ and $\overline{WR}$ determine the direction of data flow through the Data Buffers. The Data Buffers are bi-directional buffers that connect the internal bus to the external bus. When the chip is not selected ($\overline{CS} = 1$), the devices are in a high impedance state. The drivers input during $\overline{WR} \bullet \overline{CS}$ and output during $\overline{RD} \bullet \overline{CS}$.

Control and Timing Registers and Timing Control

These registers store the keyboard and display modes and other operating conditions programmed by the CPU. The modes are programmed by presenting the proper command on the data lines with $A_0 = 1$ and then sending a $\overline{WR}$. The command is latched on the rising edge of $\overline{WR}$. The command is then decoded and the appropriate function is set. The timing control contains the basic timing counter chain. The first counter is a ÷ N prescaler that can be programmed to yield an internal frequency of 100 kHz which gives a 5.1 ms keyboard scan time and a 10.3 ms debounce time. The other counters divide down the basic internal frequency to provide the proper key scan, row scan, keyboard matrix scan, and display scan times.

Scan Counter

The scan counter has two modes. In the encoded mode, the counter provides a binary count that must be externally decoded to provide the scan lines for the keyboard and display. In the decoded mode, the scan counter decodes the least significant 2 bits and provides a decoded 1 of 4 scan. Note that when the keyboard is in decoded scan, so is the display. This means that only the first 4 characters in the Display RAM are displayed.

In the encoded mode, the scan lines are active high outputs. In the decoded mode, the scan lines are active low outputs.

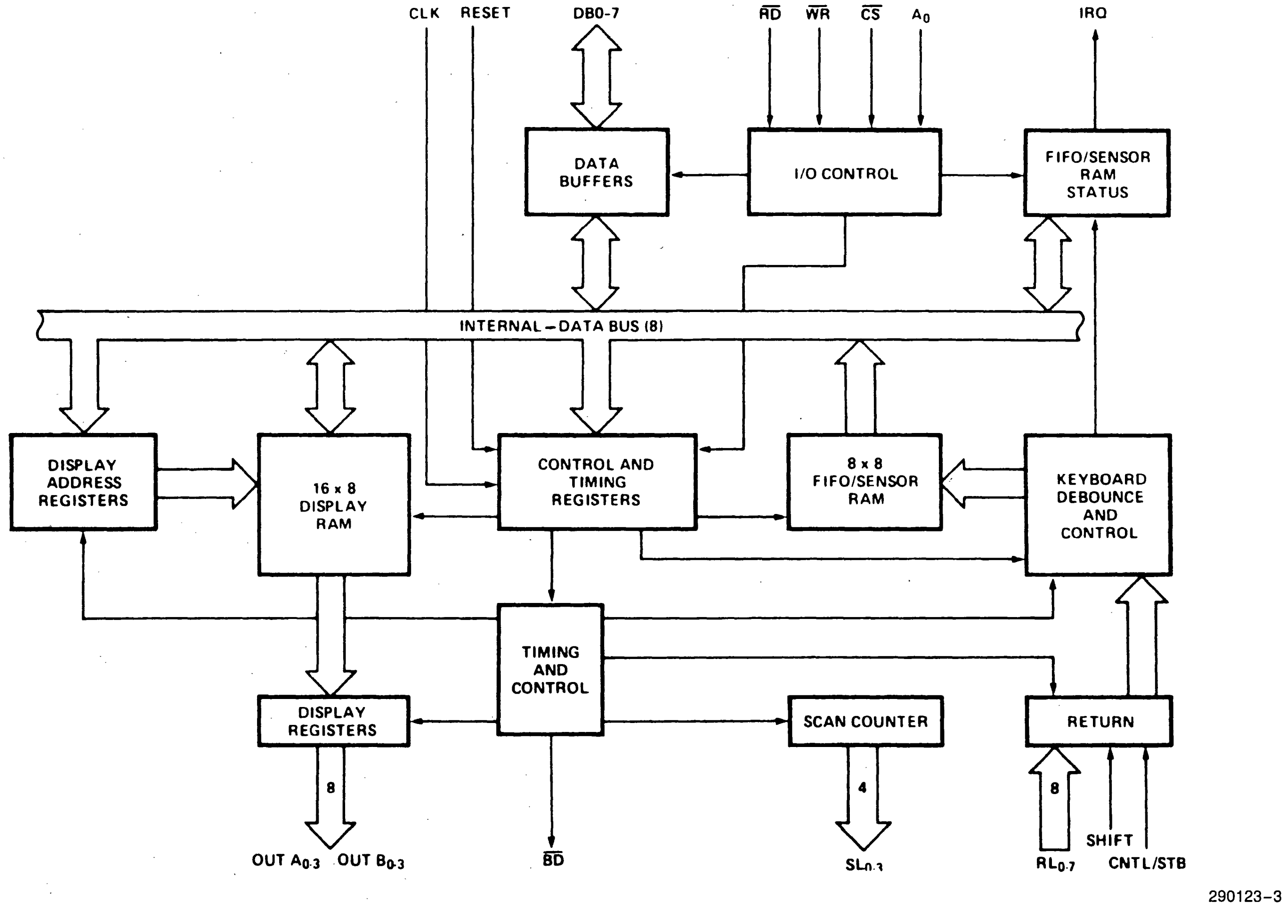


Figure 3. Internal Block Diagram

Return Buffers and Keyboard Debounce and Control

The 8 return lines are buffered and latched by the Return Buffers. In the keyboard mode, these lines are scanned, looking for key closures in that row. If the debounce circuit detects a closed switch, it waits about 10 ms to check if the switch remains closed. If it does, the address of the switch in the matrix plus the status of SHIFT and CONTROL are transferred to the FIFO. In the scanned Sensor Matrix modes, the contents of the return lines is directly transferred to the corresponding row of the Sensor RAM (FIFO) each key scan time. In Strobed Input mode, the contents of the return lines are transferred to the FIFO on the rising edge of the CNTL/STB line pulse.

FIFO/Sensor RAM and Status

This block is a dual function 8 x 8 RAM. In Keyboard or Strobed Input modes, it is a FIFO. Each new entry is written into successive RAM positions and each is then read in order of entry. FIFO status keeps track of the number of characters in the FIFO and whether it is full or empty. Too many reads or writes will be recognized as an error. The status can be read by an $\overline{RD}$ with $\overline{CS}$ low and A_0 high. The status logic also provides an IRQ signal when the FIFO is not empty. In Scanned Sensor Matrix mode, the memory is a Sensor RAM. Each row of the Sensor RAM is loaded with the status of the corresponding row of sensor in the sensor matrix. In this mode, IRQ is high if a change in a sensor is detected.

Display Address Registers and Display RAM

The Display Address Registers hold the address of the word currently being written or read by the CPU and the two 4-bit nibbles being displayed. The read/write addresses are programmed by CPU command. They also can be set to auto increment after each read or write. The Display RAM can be directly read by the CPU after the correct mode and address is set. The addresses for the A and B nibbles are automatically updated by the 8279 to match data entry by the CPU. The A and B nibbles can be entered independently or as one word, according to the mode that is set by the CPU. Data entry to the display can be set to either left or right entry. See Interface Considerations for details.

SOFTWARE OPERATION

8279 Commands

The following commands program the 8279 operating modes. The commands are sent on the Data Bus with $\overline{CS}$ low and A_0 high and are loaded to the 8279 on the rising edge of $\overline{WR}$.

Keyboard/Display Mode Set

	MSB							LSB
Code:	0	0	0	D	D	K	K	K

Where DD is the Display Mode and KKK is the Keyboard Mode.

DD

0	0	8 8-bit character display—Left entry
0	1	16 8-bit character display—Left entry*
1	0	8 8-bit character display—Right entry
1	1	16 8-bit character display—Right entry

For description of right and left entry, see Interface Considerations. Note that when decoded scan is set in keyboard mode, the display is reduced to 4 characters independent of display mode set.

KKK

0	0	0	Encoded Scan Keyboard—2 Key Lockout*
0	0	1	Decoded Scan Keyboard—2-Key Lockout
0	1	0	Encoded Scan Keyboard—N-Key Rollover
0	1	1	Decoded Scan Keyboard—N-Key Rollover
1	0	0	Encoded Scan Sensor Matrix
1	0	1	Decoded Scan Sensor Matrix
1	1	0	Strobed Input, Encoded Display Scan
1	1	1	Strobed Input, Decoded Display Scan

*Default after reset.

Program Clock

Code:	0	0	1	P	P	P	P	P

All timing and multiplexing signals for the 8279 are generated by an internal prescaler. This prescaler divides the external clock (pin 3) by a programmable integer. Bits PPPPP determine the value of this integer which ranges from 2 to 31. Choosing a divisor that yields 100 kHz will give the specified scan and debounce times. For instance, if Pin 3 of the 8279 is being clocked by a 2 MHz signal, PPPPP should be set to 10100 to divide the clock by 20 to yield the proper 100 kHz operating frequency.

Read FIFO/Sensor RAM

Code:	0	1	0	AI	X	A	A	A	X = Don't Care

The CPU sets the 8279 for a read of the FIFO/Sensor RAM by first writing this command. In the Scan Keyboard Mode, the Auto-Increment flag (AI) and the RAM address bits (AAA) are irrelevant. The 8279 will automatically drive the data bus for each subsequent read (A_0 = 0) in the same sequence in which the data first entered the FIFO. All subsequent reads will be from the FIFO until another command is issued.

In the Sensor Matrix Mode, the RAM address bits AAA select one of the 8 rows of the Sensor RAM. If the AI flag is set (AI = 1), each successive read will be from the subsequent row of the sensor RAM.

Read Display RAM

Code:	0	1	1	AI	A	A	A	A

The CPU sets up the 8279 for a read of the Display RAM by first writing this command. The address bits AAAA select one of the 16 rows of the Display RAM. If the AI flag is set (A1 = 1), this row address will be incremented after each following read *or write* to the Display RAM. Since the same counter is used for both reading and writing, this command sets the next read or *write* address and the sense of the Auto-Increment mode for both operations.

Write Display RAM

Code:	1	0	0	AI	A	A	A	A

The CPU sets up the 8279 for a write to the Display RAM by first writing this command. After writing the command with A_0 = 1, all subsequent writes with A_0 = 0 will be to the Display RAM. The addressing and Auto-Increment functions are identical to those for the Read Display RAM. However, this command does not affect the source of subsequent Data Reads; the CPU will read from whichever RAM (Display of FIFO/Sensor) which was last specified. If, indeed, the Display RAM was last specified, the Write Display RAM will, nevertheless, change the next Read location.

Display Write Inhibit/Blanking

					A	B	A	B
Code:	1	0	1	X	IW	IW	BL	BL

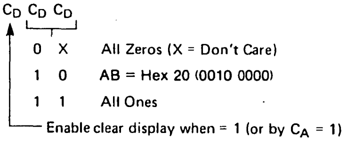
The IW Bits can be used to mask nibble A and nibble B in applications requiring separate 4-bit display ports. By setting the IW flag (IW = 1) for one of the ports, the port becomes marked so that entries to the Display RAM from the CPU do not affect that port. Thus, if each nibble is input to a BCD decoder, the CPU may write a digit to the Display RAM without affecting the other digit being displayed. It is important to note that bit B_0 corresponds to bit D_0 on the CPU bus, and that bit A_3 corresponds to bit D_7.

If the user wishes to blank the display, the BL flags are available for each nibble. The last Clear command issued determines the code to be used as a "blank." This code defaults to all zeros after a reset. Note that both BL flags must be set to blank a display formatted with a single 8-bit port.

Clear

Code:	1	1	0	C_D	C_D	C_D	C_F	C_A

The C_D bits are available in this command to clear all rows of the Display RAM to a selectable blanking code as follows:


C_D C_D C_D
0 X All Zeros (X = Don't Care)
1 0 AB = Hex 20 (0010 0000)
1 1 All Ones
Enable clear display when = 1 (or by C_A = 1)

290123-13

During the time the Display RAM is being cleared (~ 160 μs), it may not be written to. The most significant bit of the FIFO status word is set during this time. When the Display RAM becomes available again, it automatically resets.

If the C_F bit is asserted (C_F = 1), the FIFO status is cleared and the interrupt output line is reset. Also, the Sensor RAM pointer is set to row 0.

C_A, the Clear All bit, has the combined effect of C_D and C_F; it uses the C_D clearing code on the Display RAM and also clears FIFO status. Furthermore, it resynchronizes the internal timing chain.

End Interrupt/Error Mode Set

Code:

1	1	1	E	X	X	X	X	X = Don't care

For the sensor matrix modes this command lowers the IRQ line and enables further writing into RAM. (The IRQ line would have been raised upon the detection of a change in a sensor value. This would have also inhibited further writing into the RAM until reset).

For the N-key rollover mode—if the E bit is programmed to "1" the chip will operate in the special Error mode. (For further details, see Interface Considerations Section.)

Status Word

The status word contains the FIFO status, error, and display unavailable signals. This word is read by the CPU when A_0 is high and $\overline{CS}$ and $\overline{RD}$ are low. See Interface Considerations for more detail on status word.

Data Read

Data is read when A_0, $\overline{CS}$ and $\overline{RD}$ are all low. The source of the data is specified by the Read FIFO or Read Display commands. The trailing edge of $\overline{RD}$ will cause the address of the RAM being read to be incremented if the Auto-Increment flag is set. FIFO reads always increment (if no error occurs) independent of AI.

Data Write

Data that is written with A_0, $\overline{CS}$ and $\overline{WR}$ low is always written to the Display RAM. The address is specified by the latest Read Display or Write Display command. Auto-Incrementing on the rising edge of $\overline{WR}$ occurs if AI is set by the latest display command.

INTERFACE CONSIDERATIONS

Scanned Keyboard Mode, 2-Key Lockout

There are three possible combinations of conditions that can occur during debounce scanning. When a key is depressed, the debounce logic is set. Other depressed keys are looked for during the next two scans. If none are encountered, it is a single key depression and the key position is entered into the FIFO along with the status of CNTL and SHIFT lines. If the FIFO was empty, IRQ will be set to signal the CPU that there is an entry in the FIFO. If the FIFO was full, the key will not be entered and the error flag will be set. If another closed switch is encountered, no entry to the FIFO can occur. If all other keys are released before ths one, then it will be entered to the FIFO. If this key is released before any other, it will be entirely ignored. A key is entered to the FIFO only once per depression, no matter how many keys were pressed along with it or in what order they were released. If two keys are depressed within the debounce cycle, it is a simultaneous depression. Neither key will be recognized until one key remains depressed alone. The last key will be treated as a single key depression.

Scanned Keyboard Mode, N-Key Rollover

With N-key Rollover each key depression is treated independently from all others. When a key is depressed, the debounce circuit waits 2 keyboard scans and then checks to see if the key is still down. If it is , the key is entered into the FIFO. Any number of keys can be depressed and another can be recognized and entered into the FIFO. If a simultaneous depression occurs, the keys are recognized and entered according to the order the keyboard scan found them.

Scanned Keyboard—Special Error Modes

For N-key rollover mode the user can program a special error mode. This is done by the "End Interrupt/Error Mode Set" command. The debounce cycle and key-validity check are as in normal N-key mode. If during a *single debounce cycle*, two keys are found depressed, this is considered a simultaneous multiple depression, and sets an error flag. This flag will prevent any further writing into the FIFO and will set interrupt (if not yet set). The error flag could be read in this mode by reading the FIFO STATUS word. (See "FIFO STATUS" for further details.) The error flag is reset by sending the normal CLEAR command with CF = 1.

Sensor Matrix Mode

In Sensor Matrix mode, the debounce logic is inhibited. The status of the sensor switch is inputted directly to the Sensor RAM. In this way the Sensor RAM keeps an image of the state of the switches in the sensor matrix. Although debouncing is not provided, this mode has the advantage that the CPU knows how long the sensor was closed and when it was released. A keyboard mode can only indicate a validated closure. To make the software easier, the designer should functionally group the sensors by row since this is the format in which the CPU will read them.

The IRQ line goes high if any sensor value change is detected at the end of a sensor matrix scan. The IRQ line is cleared by the first data read operation if the Auto-Increment flag is set to zero, or by the End Interrupt command if the Auto-Increment flag is set to one.

NOTE:
Multiple changes in the matrix Addressed by ($SL_{0-3} = 0$) may cause multiple interrupts. ($SL_0 = 0$ in the Decoded Mode.) Reset may cause the 8279 to see multiple changes.

Data Format

In the Scanned Keyboard mode, the character entered into the FIFO corresponds to the position of the switch in the keyboard plus the status of the CNTL and SHIFT lines (non-inverted). CNTL is the MSB of the character and SHIFT is the next most significant bit. The next three bits are from the scan counter and indicate the row the key was found in. The last three bits are from the column counter and indicate to which return line the key was connected.

MSB LSB

CNTL	SHIFT	SCAN	RETURN

SCANNED KEYBOARD DATA FORMAT

In Sensor Matrix mode, the data on the return lines is entered directly in the row of the Sensor RAM that corresponds to the row in the matrix being scanned. Therefore, each switch position maps directly to a Sensor RAM position. The SHIFT and CNTL inputs are ignored in this mode. Note that switches are not necessarily the only thing that can be connected to the return lines in this mode. Any logic that can be triggered by the scan lines can enter data to the return line inputs. Eight multiplexed input ports could be tied to the return lines and scanned by the 8279.

MSB LSB

RL_7	RL_6	RL_5	RL_4	RL_3	RL_2	RL_1	RL_0

In Strobed Input mode, the data is also entered to the FIFO from the return lines. The data is entered by the rising edge of a CNTL/STB line pulse. Data can come from another encoded keyboard or simple switch matrix. The return lines can also be used as a general purpose strobed input.

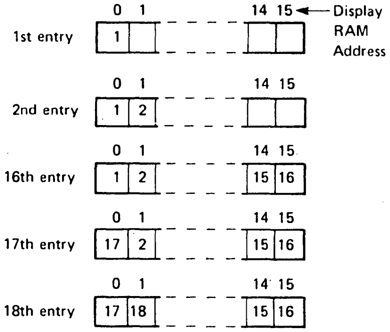
MSB LSB

RL_7	RL_6	RL_5	RL_4	RL_3	RL_2	RL_1	RL_0

Display

Left Entry

Left Entry mode is the simplest display format in that each display position directly corresponds to a byte (or nibble) in the Display RAM. Address 0 in the RAM is the left-most display character and address 15 (or address 7 in 8 character display) is the right most display character. Entering characters from position zero causes the display to fill from the left. The 17th (9th) character is entered back in the left most position and filling again proceeds from there.

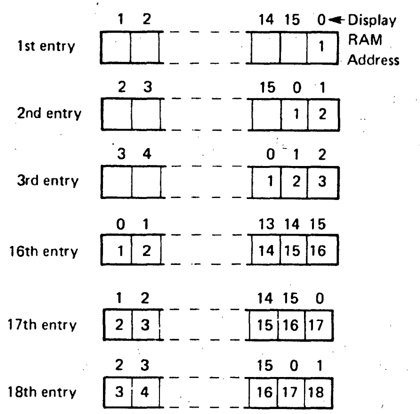


290123-14

Left Entry Mode (Auto Increment)

Right Entry

Right entry is the method used by most electronic calculators. The first entry is placed in the right most display character. The next entry is also placed in the right most character after the display is shifted left one character. The left most character is shifted off the end and is lost.



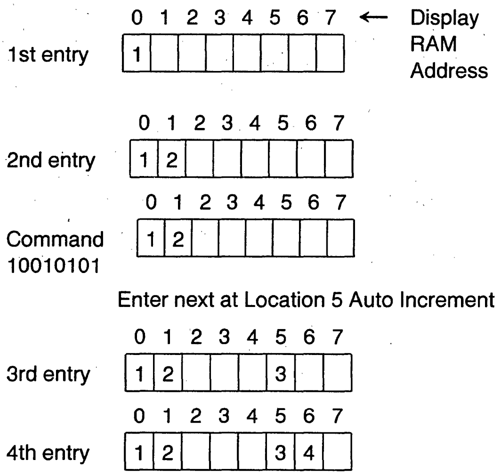
290123-15

Right Entry Mode (Auto Increment)

Note that now the display position and register address do not correspond. Consequently, entering a character to an arbitrary position in the Auto Increment mode may have unexpected results. Entry starting at Display RAM address 0 with sequential entry is recommended.

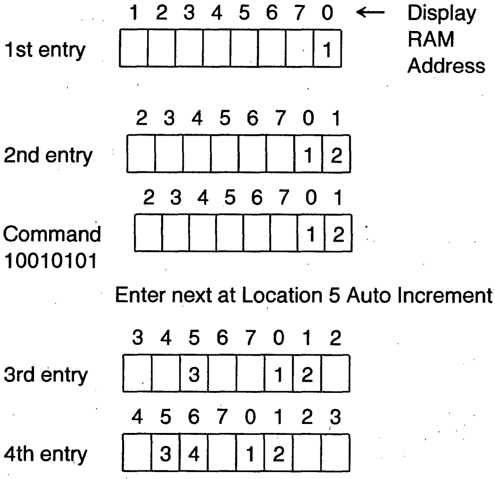
Auto Increment

In the Left Entry mode, Auto Incrementing causes the address where the CPU will next write to be incremented by one and the character appears in the next location. With non-Auto Incrementing the entry is both to the same RAM address and display position. Entry to an arbitrary address in the Auto Increment mode has no undesirable side effects and the result is predictable:



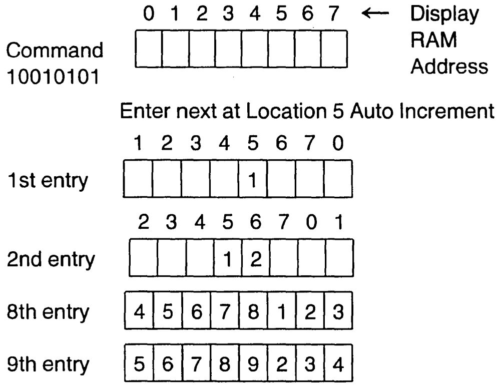
LEFT ENTRY MODE (AUTO INCREMENT)

In the Right Entry mode, Auto Incrementing and non Incrementing have the same effect as in the Left Entry except if the address sequence is interrupted.



RIGHT ENTRY MODE (AUTO INCREMENT)

Starting at an arbitrary location operates as shown below:



RIGHT ENTRY MODE
(AUTO INCREMENT)

Entry appears to be from the intial entry point.

8/16 Character Display Formats

If the display mode is set to an 8 character display, the on duty-cycle is double what it would be for a 16 character display (e.g., 5.1 ms scan time for 8 characters vs. 10.3 ms for 16 characters with 100 kHz internal frequency).

G. FIFO Status

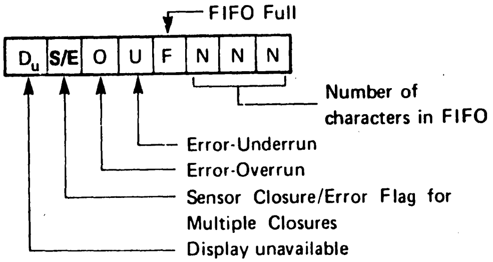
FIFO status is used in the Keyboard and Strobed Input modes to indicate the number of characters in the FIFO and to indicate whether an error has occurred. There are two types of errors possible: overrun and underrun. Overrun occurs when the entry of another character into a full FIFO is attempted. Underrun occurs when the CPU tries to read an empty FIFO.

The FIFO status word also has a bit to indicate that the Display RAM was unavailable because a Clear Display or Clear All command had not completed its clearing operation.

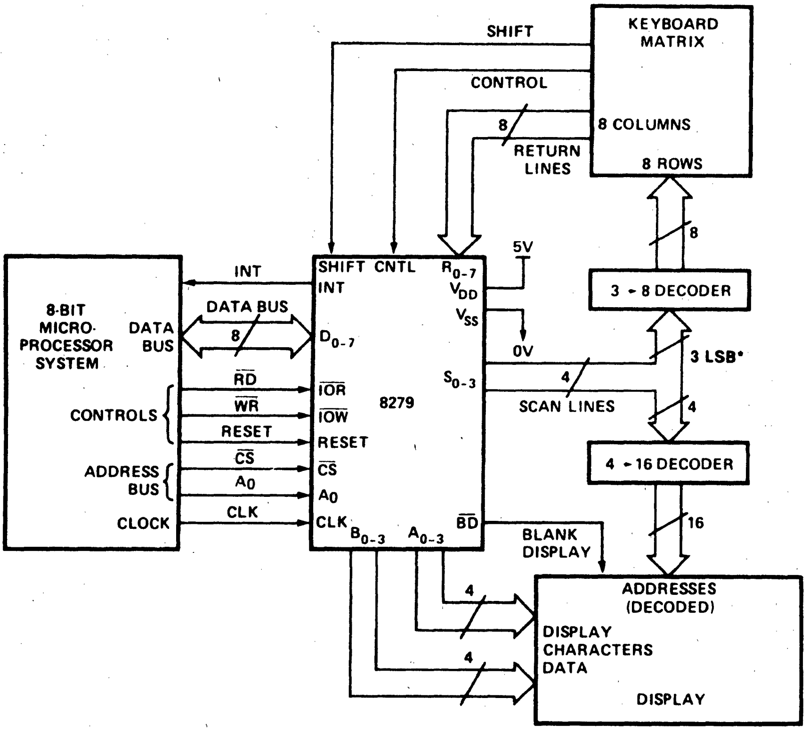
In a Sensor Matrix mode, a bit is set in the FIFO status word to indicate that at least one sensor closure indication is contained in the Sensor RAM.

In Special Error Mode the S/E bit is showing the error flag and serves as an indication to whether a simultaneous multiple closure error has occurred.

FIFO STATUS WORD


FIFO Full
D_u
S/E
O
U
F
N
N
N
Number of characters in FIFO
Error-Underrun
Error-Overrun
Sensor Closure/Error Flag for Multiple Closures
Display unavailable

290123-4



290123-5

*Do not drive the keyboard decoder with the MSB of the scan lines.

Figure 4. System Block Diagram

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature ....0°C to 70°C

Storage Temperature............. −65°C to 125°C

Voltage on any Pin with
Respect to Ground.............. −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 T_A = 0°C to 70°C V_{SS} = 0V (Note 3)*

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL1}	Input Low Voltage for Return Lines	−0.5	1.4	V	
V_{IL2}	Input Low Voltage for All Others	−0.5	0.8	V	
V_{IH1}	Input High Voltage for Return Lines	2.2		V	
V_{IH2}	Input High Voltage for All Others	2.0		V	
V_{OL}	Output Low Voltage		0.45	V	(Note 1)
V_{OH1}	Output High Voltage on Interrupt Line	3.5		V	(Note 2)
V_{OH2}	Other Outputs	2.4			I_{OH} = −400 μA 8279-5 −100 μA 8279
I_{IL1}	Input Current on Shift, Control and Return Lines		+10 −100	μA μA	$V_{IN} = V_{CC}$ V_{IN} = 0V
I_{IL2}	Input Leakage Current on All Others		±10	μA	$V_{IN} = V_{CC}$ to 0V
I_{OFL}	Output Float Leakage		±10	μA	$V_{OUT} = V_{CC}$ to 0.45V
I_{CC}	Power Supply Current		120	mA	
C_{IN}	Input Capacitance		10	pF	f_C = 1 MHz Unmeasured Pins Returned to V_{SS}(6)
C_{OUT}	Output Capacitance		20	pF	

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{SS} = 0V (Note 3)*

Bus Parameters

READ CYCLE

Symbol	Parameter	8279		8279-5		Unit
		Min	Max	Min	Max	
t_{AR}	Address Stable Before $\overline{READ}$	50		0		ns
t_{RA}	Address Hold Time for $\overline{READ}$	5		0		ns
t_{RR}	$\overline{READ}$ Pulse Width	420		250		ns
t_{RD}(4)	Data Delay from $\overline{READ}$		300		150	ns
t_{AD}(4)	Address to Data Valid		450		250	ns
t_{DF}	$\overline{READ}$ to Data Floating	10	100	10	100	ns
t_{RCY}	Read Cycle Time	1		1		μs
t_{AW}	Address Stable Before $\overline{WRITE}$	50		0		ns
t_{WA}	Address Hold Time for $\overline{WRITE}$	20		0		ns

A.C. CHARACTERISTICS (Continued)

WRITE CYCLE

Symbol	Parameter	8279		8279-5		Unit
		Min	Max	Min	Max	
t_{WW}	$\overline{\text{WRITE}}$ Pulse Width	400		250		ns
t_{DW}	Data Set Up Time for $\overline{\text{WRITE}}$	300		150		ns
t_{WD}	Data Hold Time for $\overline{\text{WRITE}}$	40		0		ns
t_{WCY}	Write Cycle Time	1		1		μs

OTHER TIMINGS

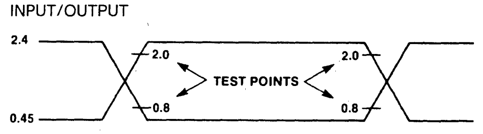
Symbol	Parameter	8279		8279-5		Unit
		Min	Max	Min	Max	
$t_{\phi W}$	Clock Pulse Width	230		120		ns
t_{CY}	Clock Period	500		320		ns

Keyboard Scan Time	5.1 ms
Keyboard Debounce Time	10.3 ms
Key Scan Time	80 μs
Display Scan Time	10.3 ms
Digit-on Time	480 μs
Blanking Time	160 μs
Internal Clock Cycle(5)	10 μs

NOTES:
1. 8279, I_{OL} = 1.6 mA; 8279-5, I_{OL} = 2.2 mA.
2. I_{OH} = −100 μA
3. 8279, V_{CC} = +5V ±5%; 8279-5, V_{CC} = +5V ±10%
4. 8279, C_L = 100 pF; 8279-5, C_L = 150 pF.
5. The Prescaler should be programmed to provide a 10 μs internal clock cycle.
6. Sampled not 100% tested. T_A = 25°C.

* For Extended Temperature EXPRESS, use M8279A electrical parameters.

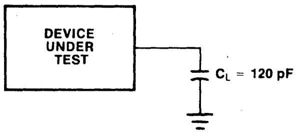
A.C. TESTING INPUT, OUTPUT WAVEFORM



290123-6

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

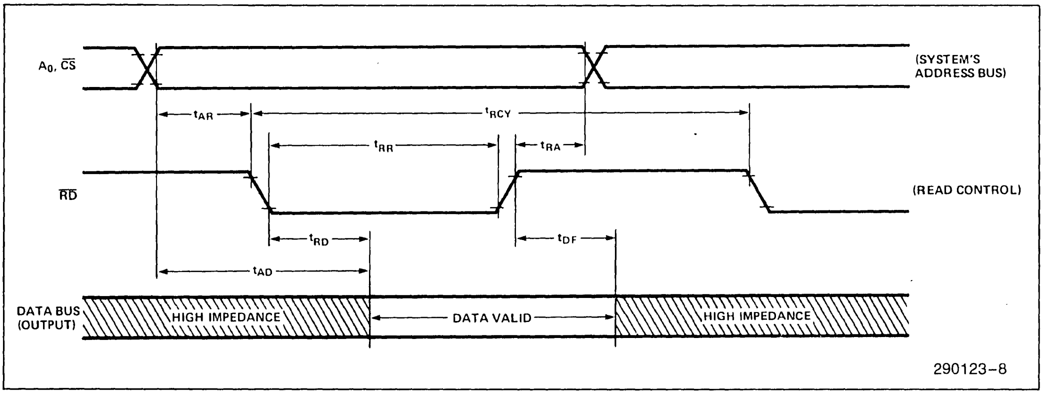


290123-7

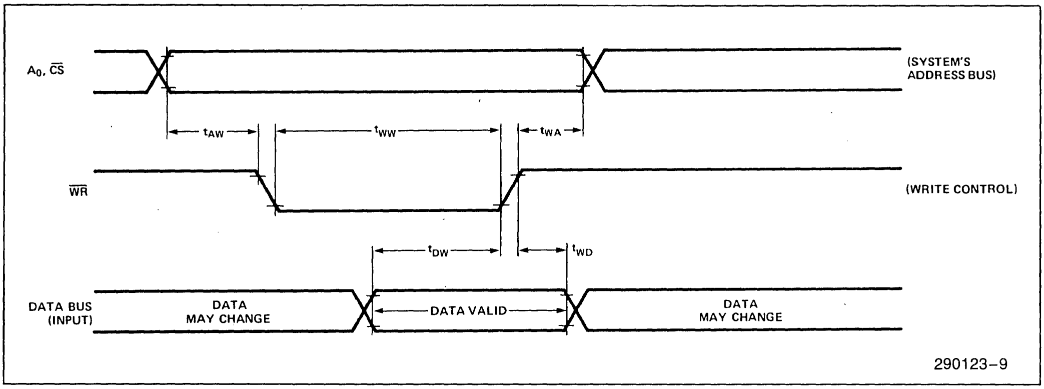
C_L = 120 pF
C_L Includes Jig Capacitance

WAVEFORMS

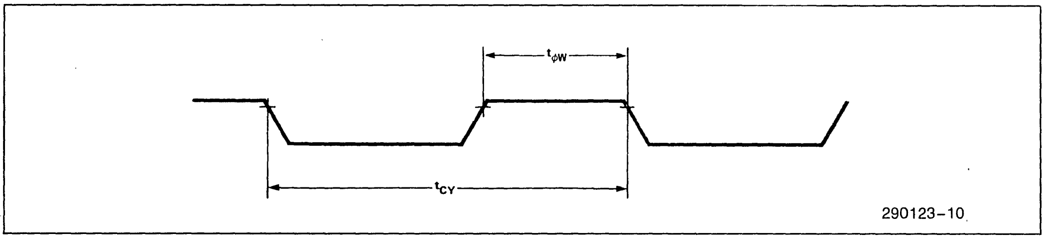
READ OPERATION



WRITE OPERATION

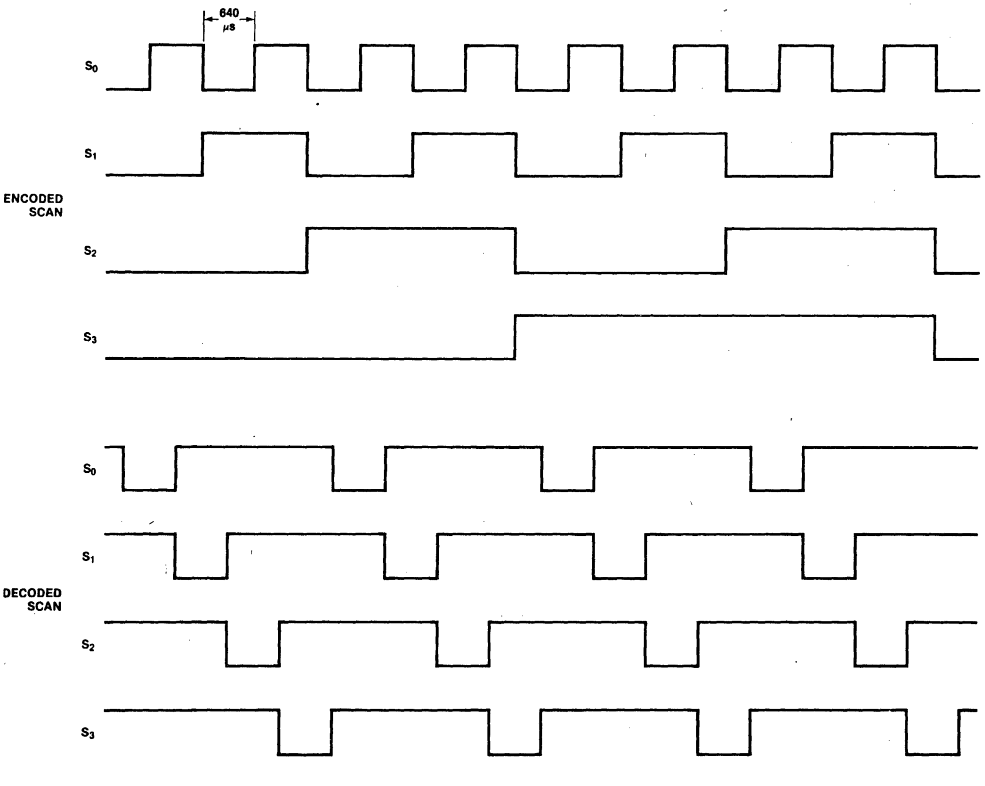


CLOCK INPUT



WAVEFORMS (Continued)

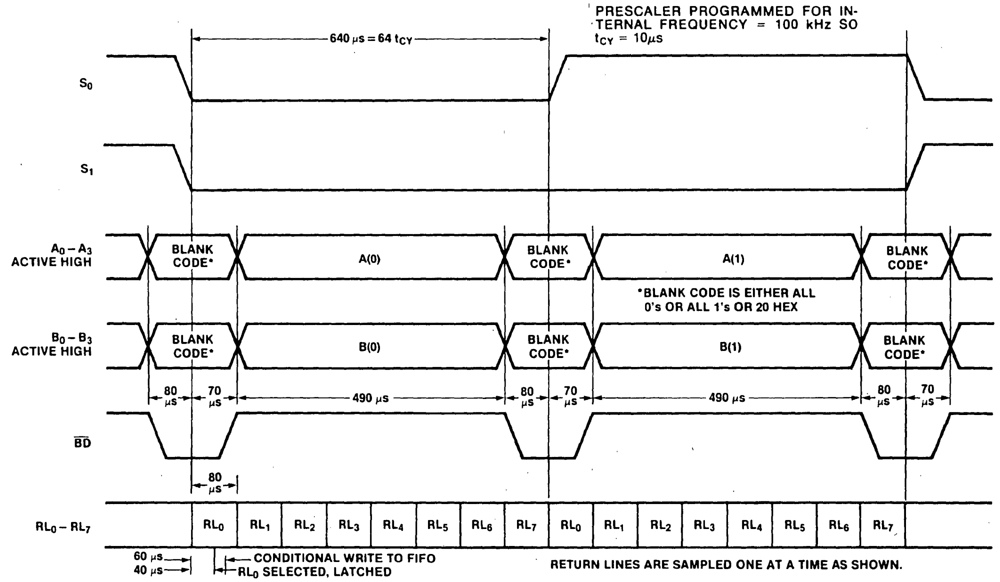
SCAN



290123–11

WAVEFORMS (Continued)

DISPLAY


PRESCALER PROGRAMMED FOR INTERNAL FREQUENCY = 100 kHz SO t_{CY} = 10μs
640 μs = 64 t_{CY}
S_0
S_1
$A_0 - A_3$ ACTIVE HIGH
BLANK CODE*
A(0)
A(1)
*BLANK CODE IS EITHER ALL 0's OR ALL 1's OR 20 HEX
$B_0 - B_3$ ACTIVE HIGH
B(0)
B(1)
80 μs
70 μs
490 μs
$\overline{BD}$
$RL_0 - RL_7$
RL_0 RL_1 RL_2 RL_3 RL_4 RL_5 RL_6 RL_7
60 μs
40 μs
CONDITIONAL WRITE TO FIFO
RL_0 SELECTED, LATCHED
RETURN LINES ARE SAMPLED ONE AT A TIME AS SHOWN.

290123–12

NOTE:
Shown is encoded scan left entry
S_2–S_3 are not shown but they are simply S_1 divided by 2 and 4.

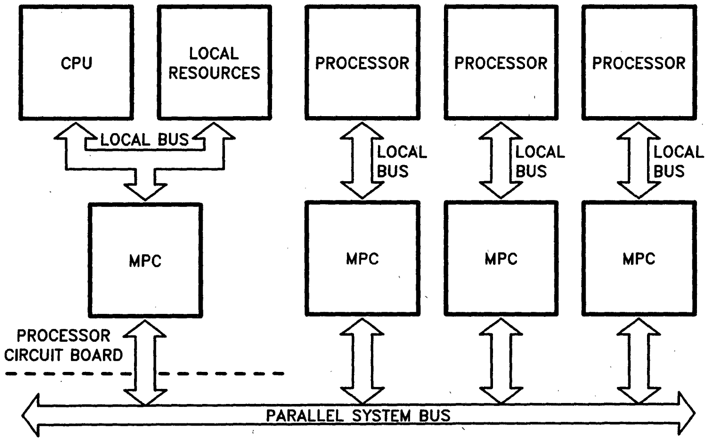
82389
MESSAGE PASSING COPROCESSOR
A MULTIBUS® II BUS INTERFACE CONTROLLER

- **Highly Integrated VLSI Device**
  - **—Single-Chip Interface for the Parallel System Bus (IEEE 1296)**
  - **—Interrupt Handling/Bus Arbitration Functions**
  - **—Dual-Buffer Input and Output DMA Capabilities**
  - **—Nine 32-Byte High Speed FIFOs**
- **Multiple Interface Support**
  - **—Complete Protocol Support of the PSB Bus (Message Passing)**
  - **—Processor Independent Interface (8, 16, or 32-Bit CPU)**
  - **—Low-Cost 8-Bit Microcontroller Interface**
  - **—Dual-Port Memory Interface**
- **High Performance Coprocessing Functions**
  - **—Offloads CPU for Communication and Bus Interfacing**
  - **—40 Megabytes/Sec Burst Transfer Speed**
  - **—Optimized for Real-Time Response (Max. 900 ns for 32-Byte Interrupt Packet)**
- **Compatible with Bus Arbiter Controller (BAC) and Message Interrupt Controller (MIC) Interface Designs**
- **CMOS Technology**
- **149 Pin PGA Package (15 x 15 Grid)**

The MPC 82389 is a highly integrated VLSI device that maximizes the performance of a Multibus® II based multiprocessor system. It integrates the functions of bus arbitration, packetizing data for transmit, error handling and interrupt control. Because of these integrated functions the host CPU can be offloaded to utilize the maximum bus performance and subsequently increase the system throughput. The MPC 82389 also supports geographic addressing by providing access to the local interconnect registers for reference and control.

The MPC 82389 is designed to interface with an 8, 16, or 32-bit processor and the Parallel System Bus performance is not affected by the CPU buswidth or its bandwidth. The data on the Parallel System Bus is burst transferred at the maximum bus speed of 40 Megabytes/second regardless of CPU bus performance. Such performance is possible due to decoupling of the CPU from the Parallel System Bus.

MULTIPROCESSOR ARCHITECTURE



290145-1

Figure 1-1

October 1988
Order Number: 290145-002

1.0 MPC 82389 INTRODUCTION

The Message Passing coprocessor 82389 is a highly integrated CMOS VLSI device to interconnect intelligent boards in a MULTIBUS II system environment. The parallel system bus of the MBII architecture definition however allows existence of intelligent and non-intelligent boards in the system.

This section of the data sheet describes the device in general including the definition of message passing protocol and the subsequent sections will contain the detailed features of the device. Please refer to the MPC User's Manual for more details.

1.1 MPC 82389 Functional Overview

The MPC 82389 is a Bus Interface Controller designed to offload the host CPU for interprocessor communication on the PSB network, and it's primary function is to support the communication protocol standard defined for the PSB bus (message passing). The device provides both the physical and data link support. By standardizing the signal interface (physical), it allows multiple vendors to offer standard add-on products for the user and at the same time it reduces costly overheads for the suppliers. The data link protocol is completely handled by the MPC 82389 including packetization after receiving data from the local interface, bus arbitration, burst transfer and error detection without the CPU intervention.

The PSB bus standard is defined for easy access and sharing of resources in a distributed processing environment. The MPC 82389 complements this standard by providing an optimized interface for the PSB bus usage at its maximum bandwidth.

The MPC 82389 also features three additional interfaces for use on a processor board.

Local Bus Interface for Host independent CPU. The CPU can be 8, 16 or 32 bits wide.

Interconnect Bus for interfacing to a low-cost microcontroller. The interconnect bus has a local address space which can be accessed by other agents on the PSB bus via the MPC.

Dual-Port Memory Interface to support an alternative communication approach which may coexist with the message passing method.

1.2 Major Operations of the MPC 82389

— Support of both unsolicited and solicited message transfers. This interprocessor communication protocol allows an intelligent agent on the PSB bus to communicate to another without any CPU intervention and at rates approaching the PSB bus bandwidth.

— Support of single cycle accesses by the host processor to memory and I/O locations resident on the PSB bus. Bus architecture, parity generation and error detection is completely handled by the MPC 82389 coprocessor.

— Support of accesses to local interconnect space by both the host processor and other agents on the PSB bus.

— Support of accesses by the host processor to interconnect location assigned to other PSB bus agents.

— Support of accesses to local, dual-port memory by other agents on the PSB bus.

1.3 Message Passing Protocol

The Multibus II architecture defines the data transfer protocol between agents on the PSB bus as Message Passing.

Message Passing allows the PSB agents to transfer variable amounts of data at rates approaching the maximum bus speed. The MPC 82389 fully supports the standardized data link protocol designed for the PSB and the entire handshaking between agents on the PSB bus is handled by the MPC 82389 without the CPU intervention.

There are two types of messages that can be transmitted from one PSB agent to another: **Unsolicited Messages** and **Solicited Messages**.

Unsolicited Messages—An unsolicited message is an intelligent interrupt also called virtual interrupt. This unsolicited message, as the name implies, is an asynchronous event to notify the receiving agent to prepare for the receipt of **Solicited Messages**. The message is in the form of a packet and it consists of information about the interrupt. By providing such intelligence the receiving agent's CPU do not have to poll for information, thus resulting in minimal latency.

Solicited Messages—The solicited messages are the actual data that are transmitted from one MPC to another. The data is once again broken into packets and these packets are transferred using the negotiation (handshaking) process which are synchronized by the MPC 82389 coprocessors.

1.4 Compatibility with BAC/MIC Interface Designs

The Bus Arbiter Controller (BAC) and Message Interrupt Contoller (MIC) were the first support components for the Parallel System Bus. The BAC implemented the full arbitration, requestor, and replier functions of the PSB. The MIC supported the transmit and receiving of minimum size unsolicited (interrupt) messages.

To ensure future compatibility with the MPC 82389 implementation, the BAC/MIC architecture was put on a module called the docket for direct incorporation onto the base-board. The PSB implementations for the MPC and the BAC/MIC docket are compatible in all respects. These implementations may coexist on the PSB bus of the same system. For the host and microcontroller interfaces, compatibility is maintained for message and interconnect space operations. It is, thus, possible to replace the BAC/MIC docket with the MPC at little impact to the board design.

Software initialization of the operating parameters for the MIC is possible through the Configuration Register to support host widths of 8, 16 or 32 bits. The MIC supports a host width of 8 bits only. Both implementations present similar software interfaces for the sending and receiving of interrupts. For details about the initialization procedures and interrupt protocols, please refer to the MPC Users's Manual, Part Number: 176526.

The MPC offers capabilities and performance far superior to the BAC/MIC implementation. Using the MPC, interrupt handling at the host interface can be improved by over an order of magnitude. Whereas the MIC can handle only minimum 4-byte interrupts, the MPC enables up to 28 bytes of data to be sent and received along with each interrupt. Further, the MPC has dedicated support for DMA based solicited message transfer.

2.0 MPC 82389 INTERFACES

The MPC 82389 features 4 interfaces: the local CPU bus for processor interface, the interconnect bus for 8-bit microcontroller interface, the Parallel System Bus interface and the dual-port memory interface.

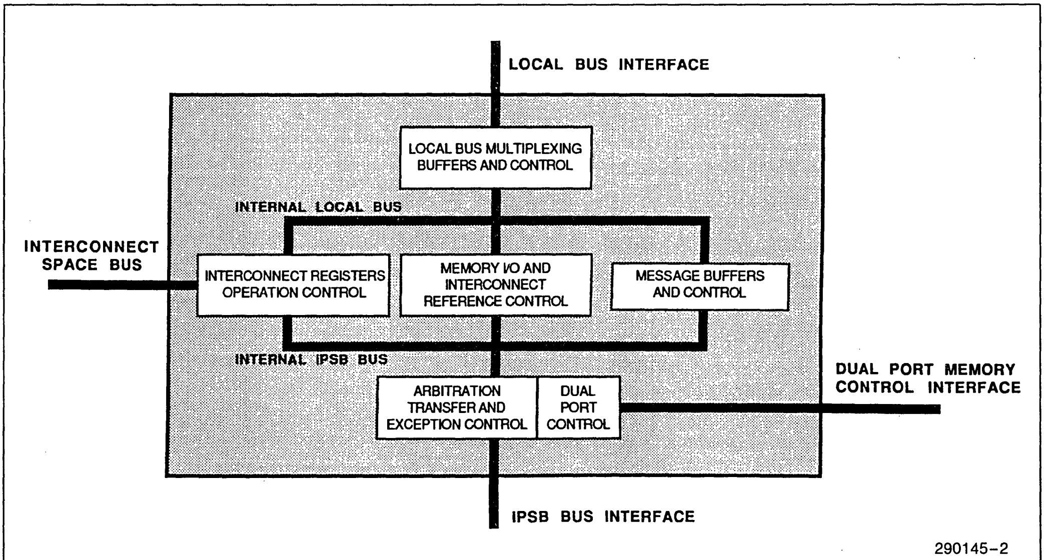

LOCAL BUS INTERFACE
LOCAL BUS MULTIPLEXING BUFFERS AND CONTROL
INTERNAL LOCAL BUS
INTERCONNECT SPACE BUS
INTERCONNECT REGISTERS OPERATION CONTROL
MEMORY I/O AND INTERCONNECT REFERENCE CONTROL
MESSAGE BUFFERS AND CONTROL
INTERNAL IPSB BUS
DUAL PORT MEMORY CONTROL INTERFACE
ARBITRATION TRANSFER AND EXCEPTION CONTROL
DUAL PORT CONTROL
IPSB BUS INTERFACE
290145-2

Figure 2-1. MPC Bus Interfaces

2.1 Local Bus

The local bus of the MPC 82389 is used to interface to a host processor. The CPU can be 8, 16, or 32 bits wide and the interface is processor independent.

The local bus interface supports direct references to memory, I/O and interconnect address space on the PSB bus. It also supports references to local interconnect space and the full message passing protocol. The entire local bus interface can be categorized into three sub-interfaces: register, reference and DMA.

2.1.1 REGISTER INTERFACE

The MPC 82389 local bus register interface is used for message operations and access to the interconnect space. These operations are asynchronous to the bus clock or interconnect bus operation.

2.1.2 REFERENCE INTERFACE

The MPC 82389 local bus reference interface supports direct references to memory, I/O and interconnect address space on the PSB bus. Memory and I/O references are initiated by the CPU to the MPC. The MPC responds by putting the CPU on hold while arbitrating for PSB bus access. The CPU is held in WAIT state until the operation is complete or a bus exception occurs on the PSB bus. The reference interface supports both read and write to the registers. The local interconnect address space is differentiated from the interconnect address on the PSB bus by the bit pattern stored in the slot address register of the MPC.

2.1.3 DMA INTERFACE

The DMA interface transfers data between local memory and the MPC 82389 during solicited message operations. The MPC provides both the input and output channels to the PSB bus. The number of transfers to or from the MPC is determined by the maximum size of the packet buffer (32-byte) or completion of the solicited transfer, whichever is less.

The DMA interface is designed to operate on either a read or write command to allow two-cycle operation or fly-by transfers. For two-cycle operation, the DMA uses a read operation to fetch data from the MPC and a write to put data into the MPC. Conversely, a fly-by read or write operation occurs correspondingly to memory write or read operation.

The DMA interface to the MPC performs best with aligned transfers. However, for compatibility with existing software, the MPC supports operations of arbitrary byte strings.

2.2 Parallel System Bus

The MPC 82389 provides a full 32-bit interface to the PSB bus and participates in arbitration, requestor control, replier control and error handling.

2.2.1 ARBITRATION

The MPC 82389 initials PSB bus access arbitration upon request generated inside the MPC. This request could be the result of a synchronized PSB bus reference request (memory, I/O or interconnect) or a message packet transmit request from the CPU. The PSB bus arbitration specification can be referred in the document, MPC User's Manual, Part Number: 176526.

2.2.3 REPLIER CONTROL

The MPC 82389 as a replier supports interconnect space reference and message reception. It gets into replier mode when a match is detected between the assigned slot ID and the address on the PSB bus. The interface space microcontroller is alerted of the replier mode condition. The address comparison is disabled when the MPC is the bus owner.

The MPC allows interconnect space to be locked from the PSB bus. This inhibits local bus interface access requests.

2.2.4 ERROR HANDLING

The MPC 82389 monitors errors generated during the transfer operation. It provides error checking on incoming interconnect references that match the slot ID. If an exception occurs on the PSB bus while an interconnect operation is in progress, the MPC provides for a graceful recovery.

2.3 Interconnect Bus

The Interconnect bus of the MPC 82389 has a simple 8-bit interface. A low-cost microcontroller can be interfaced to perform board configuration at startup and other tasks like local diagnostics.

The Interconnect space of an agent has a 512-byte register range. Within this space the microcontroller can store the local operating and configuration parameters associated with the agent. For example local diagnostics can be executed out of the microcontroller and the results posted in the Interconnect space. IEEE 1296 specifications require the first record in the Interconnect space to contain the board ID and Intel recommends other record types. Refer to Interconnect Interface Specification, Part Number: 149299.

The MPC 82389 provides the path to access the local Interconnect space. The references supported are:

1. CPU local bus to the local Interconnect space
2. CPU local bus to the Interconnect space of another agent on the PSB bus
3. From the PSB bus to the local Interconnect space

The local Interconnect accesses are identified as slot IDs 24–31 and the Interconnect accesses on the PSB bus are mapped as slot IDs 0–23.

The MPC participates in PSB bus handshake protocol, parity generation and checking and agent error generation for local Interconnect accesses from the PSB bus.

The Interconnect microcontroller is the master device on the bus and all other devices including the MPC are slaves.

2.4 Dual-Port Memory Interface

The MPC 82389 supports the dual-port memory interface for those designs that must coexist with the memory passing architecture.

The dual-port services supported are: Address recognition, PSB bus replier handshake, error checking, and bus parity generation and checking. A useful recovery mechanism is provided by the MPC should a bus exception error occur while a dual-port memory access is in progress. Although the MPC 82389 provides bus parity check it is the responsibility of the memory controller to generate and check data parity.

2.5 Basic Implementation with the MPC 82389

Figure 2-2 shows a basic implementation of the MPC 82389. Included in this implementation is the interconnect interface to a microcontroller, the CPU interface and the PSB bus interface.

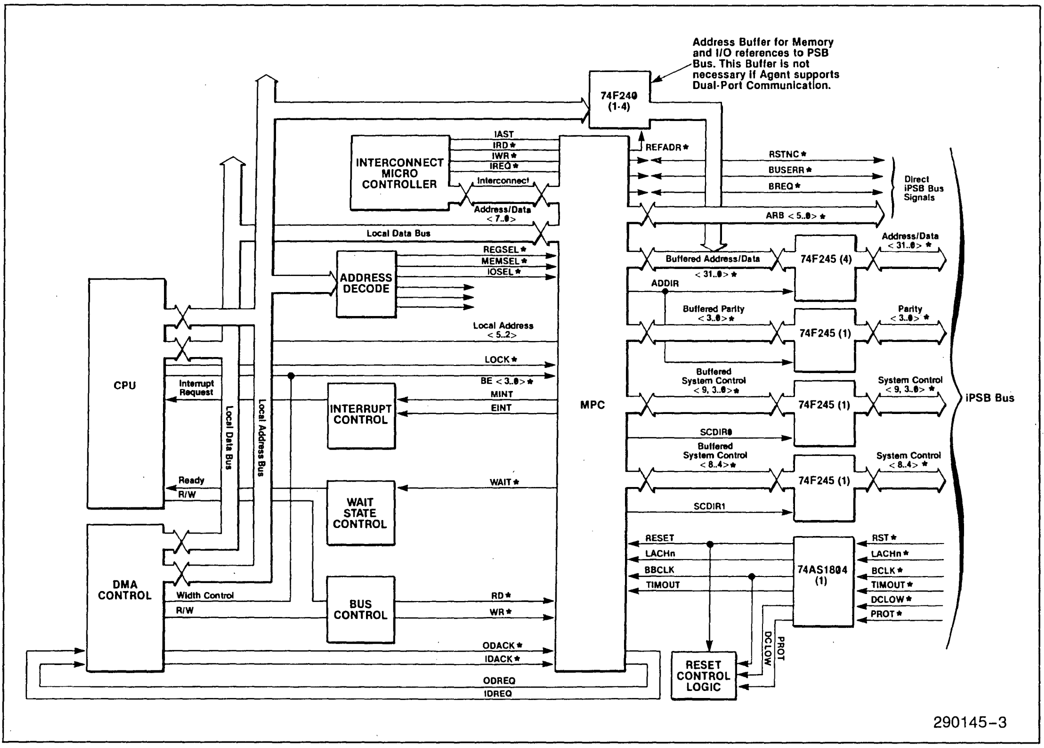


Figure 2-2. MPC Implementation to Support References

2.6 Implementation with Dual-Port Memory Interface

Figure 2-3 shows the logic required to implement dual-port operations.

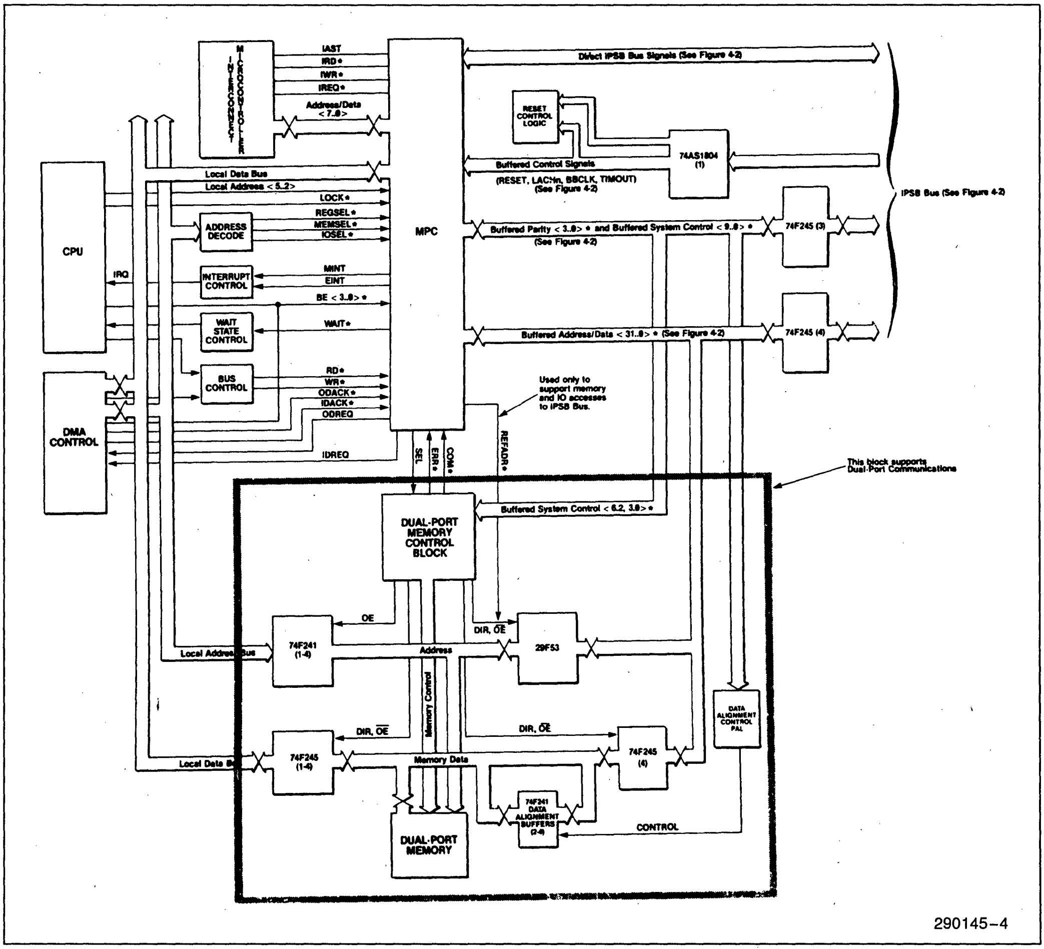


Figure 2-3. The MPC Implemented with Dual-Port Memory

3.0 MPC 82389 INTERPROCESSOR COMMUNICATION

A MULTIBUS II system can have up to 20 boards in the slot backplane. Slot zero must have a Central Services Module (CSM) which provides bus initialization and clocking. The remaining 19 slots can have a mix of intelligent and non-intelligent boards. The intelligent boards will typically communicate over the high speed PSB bus. Any processor based board may contain an MPC 82389 for high speed communication and the MPC is designed to support the system performance and the data transfer speed of the PSB bus. The MPC has an optimized PSB bus interface.

Message Passing over the PSB bus is completely handled by the MPC 82389 coprocessor without CPU intervention. The messages (data) are packetized in blocks of 32 bytes and burst transferred over the PSB bus.

The decoupling is achieved by using very high speed FIFOs of the MPC 82389. Nine 32-byte FIFOs are used in the MPC 82389. Five of these FIFOs are used for setting up the unsolicited messages (interrupts). One is used for output set up and the other four for input set up of up to four unsolicited messages. For data transmission (solicited messages) over the PSB, the MPC 82389 has two dedicated solicited output and input channels. With each channel, dual 32-byte FIFOs are used to pipeline data during output and input operation.

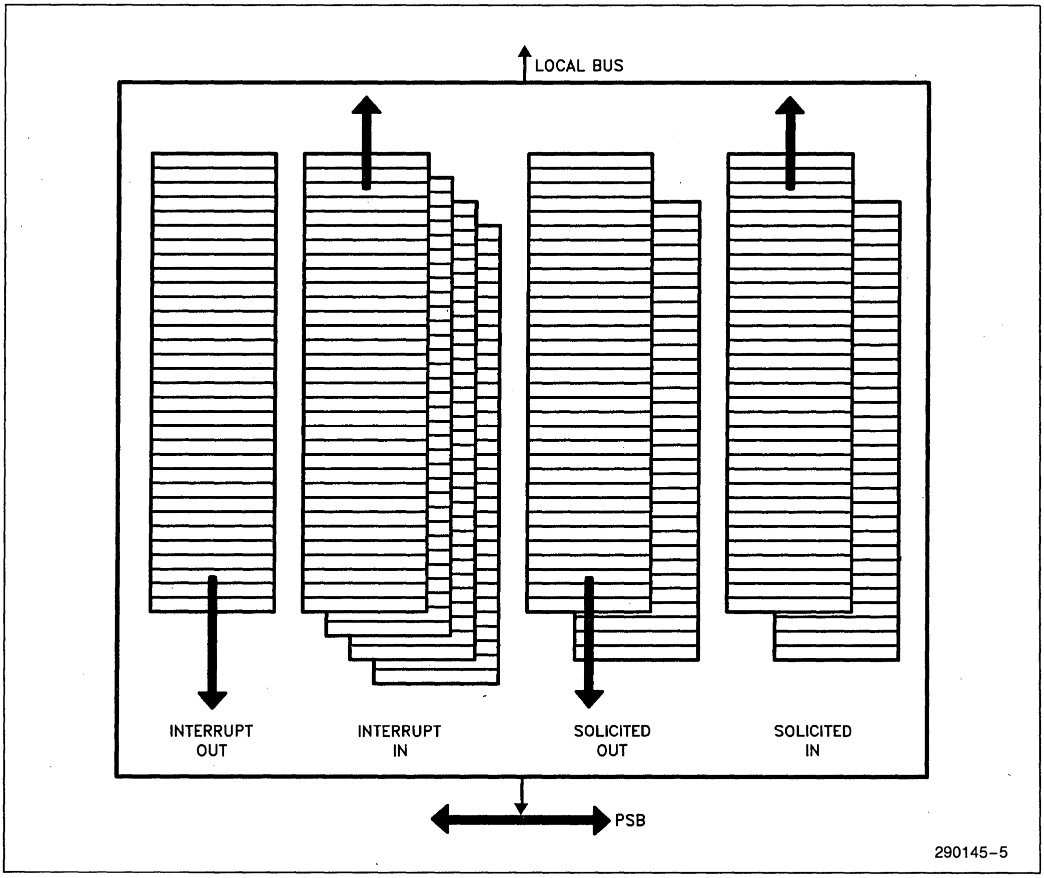


Figure 3-1. The MPC Uses Nine 32 Byte FIFOs to Couple the Local and System Buses

3.1 Communication Protocol

Any device with an interface to the PSB bus is termed as an agent. The agents communicate over the PSB bus and they completely offload the CPU for other tasks. Each agent is assigned an 8-bit address for identification.

The mechanism used for interprocessor communication over the PSB entails a standardized data link protocol, and a dedicated address space. The information which is packetized is transmitted to a dedicated address space instead of the target memory. This addressing scheme serves to decouple the CPU from the PSB bus. Packetization serves to limit the time an agent has an access over the bus, thus protecting against hogging of the bus by agents which require transmitting a large amount of data.

The data link protocol called message passing includes two kinds of messages: unsolicited and solicited.

3.1.1 UNSOLICITED MESSAGE

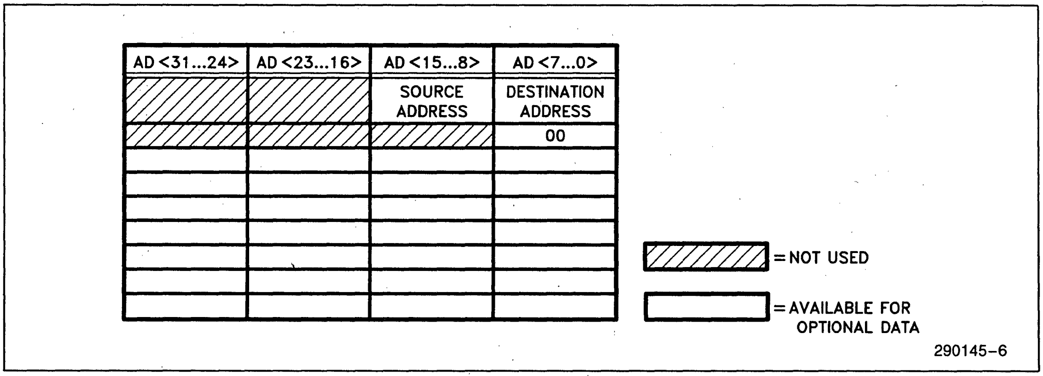
Unsolicited message is an intelligent interrupt. It is a virtual interrupt for the receiving agent and includes all the information required to service the interrupt. It takes only 900 ns to send an unsolicited message over the PSB bus, and since the receiving agent's processor does not have to poll for servicing the interrupt, this mechanism is fast and efficient.

The unsolicited message is sent as a packet consisting of up to 32 bytes, as shown in Figure 3-2. The address field is 8 bits long. Up to 255 agents can be addressed uniquely while one address is used for broadcast function.

The general format for unsolicited messages varies depending on the CPU bus width. There are other variations depending on whether the CPU is receiving or transmitting the message and the type of unsolicited message.

Unsolicited messages are asynchronous in nature. These unsolicited messages are used to set up solicited messages and contain Control and Command information.

An unsolicited message consisting of 32 bytes takes a maximum of 900 ns to transmit. The unsolicited message packet without the optional 28-byte data will take only 200 ns to transmit.



AD <31...24>	AD <23...16>	AD <15...8>	AD <7...0>
▨	▨	SOURCE ADDRESS	DESTINATION ADDRESS
▨	▨	▨	00

▨ = NOT USED

□ = AVAILABLE FOR OPTIONAL DATA

290145-6

Figure 3-2. General Interrupt Message on the PSB

AD<31..24>	AD<23..16>	AD<15..8>	AD<7..0>
		Source Add. Byte 1	Dest. Add. Byte 0
		ID Byte 3	Type Byte 2
Data, Byte 7	Data, Byte 6	Data, Byte 5	Data, Byte 4
Data, Byte 11	Data, Byte 10	Data, Byte 9	Data, Byte 8
Data, Byte 15	Data, Byte 14	Data, Byte 13	Data, Byte 12
Data, Byte 19	Data, Byte 18	Data, Byte 17	Data, Byte 16
Data, Byte 23	Data, Byte 22	Data, Byte 21	Data, Byte 20
Data, Byte 27	Data, Byte 26	Data, Byte 25	Data, Byte 24
Data, Byte 31	Data, Byte 30	Data, Byte 29	Data, Byte 28
Data, Byte 35 Solic. only	Data, Byte 34 Solic. only	Data, Byte 33 Solic. only	Data, Byte 32 Solic. only

Figure 3-3. General Message Packet Format on the PSB Bus

3.1.2 SOLICITED MESSAGE

Solicited message consists of the actual data to be transferred from one agent to another over the PSB bus. The data is packetized in blocks of 32 bytes for transfer. Up to 16 Megabytes of data may be transferred in a single solicited message.

The transfer of data is negotiated between the transmitting and receiving agents via unsolicited messages. By using the acknowledge response method through the unsolicited messages, the agents complete the transfer of data.

A solicited message contains one or more data packets. The packetization of solicited messages is handled by the MPC. The padding of header information at the transmitting end and the stripping of this information out of the packet is solely the MPC responsibility. The local CPU simply fills or empties the FIFO over the local bus. The MPC also handles the last packet fillers to maintain the 32-byte data packet format. If necessary, during output the bytes are padded and during input the padded bytes are stripped by the MPC.

For programming details consult MPC User's Manual, Part Number: 176526.

3.2 Bus Bandwidth

The advantages of decoupling the buses can be summarized in Figure 3-4. The effective speed performance numbers are also listed. The first advantage is that no resource is held in wait states while arbitration for another resource is occurring. The second advantage is that each transfer can occur at the full bandwidth of the associated bus.

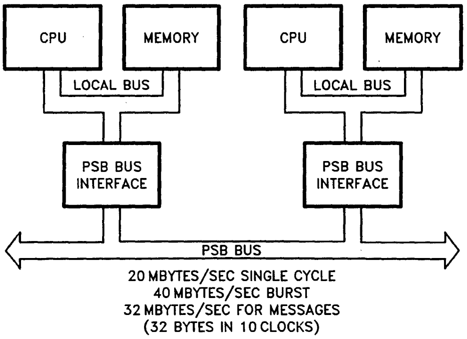


Figure 3-4. Message Passing Performance Example

4.0 MPC 82389 PIN DESCRIPTION

The MPC 82389 is packaged in a 149 pin package. The signals for the device are functionally divided by their associated interfaces as shown in Figure 4-1.

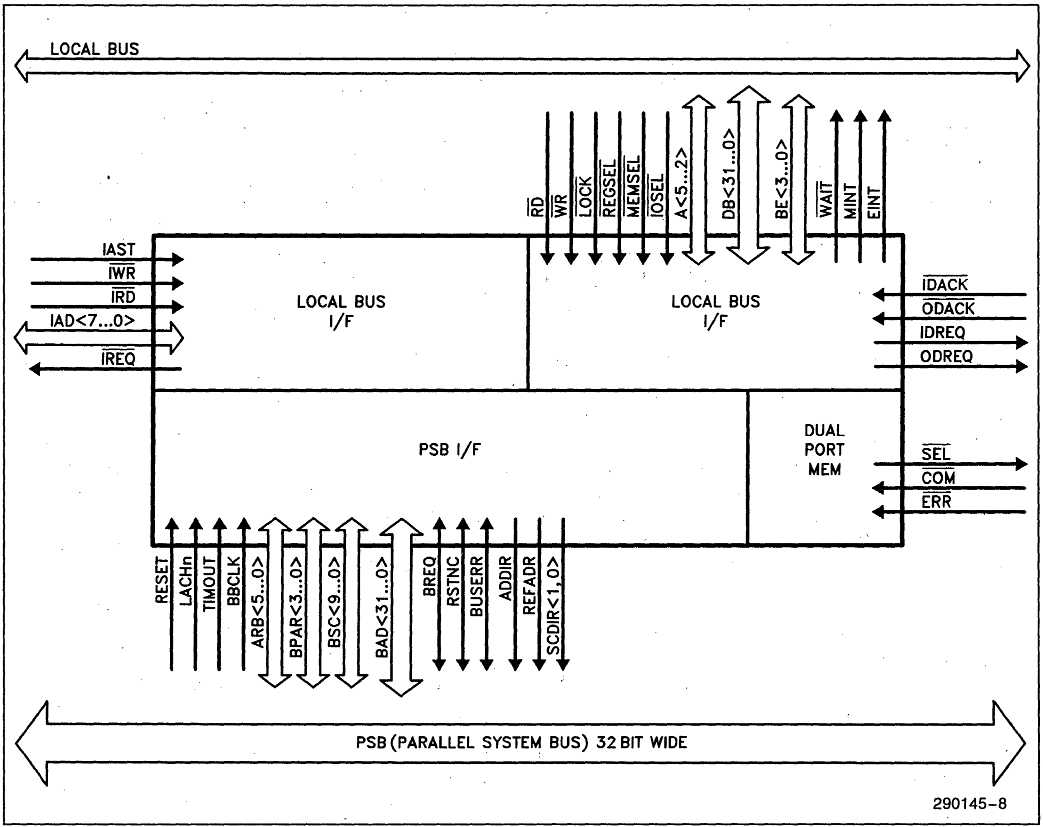


Figure 4-1. MPC Functional Blocks

4.1 PSB Bus Signals

This section describes each of the PSB bus signals that interface with the MPC. For complete descriptions of these signals, see the MULTIBUS II Architecture Specification, Part Number: 146077.

The PSB bus signals interfaced by the MPC 82389 fall into five groups, depending on function:

- Arbitration Operation Signal Group
- Address/Data Bus Signal Group
- System Control Signal Group
- Central Control Signal Group
- Exception Operation Signal Group

Unless otherwise stated, all PSB bus signals are synchronous to the bus clock.

4.1.1 ARBITRATION OPERATION SIGNAL GROUP

The MPC 82389 interfaces directly with the Arbitration Operation Signal Group of the PSB bus. These are all high-current drive, open-collector signals. Below is a description of each signal.

BREQ (Bus Request)
BREQ is a bidirectional open-collector signal that connects directly to the PSB bus. As an input to the MPC, it indicates that agents are awaiting access to the bus. In fair access mode, this inhibits the MPC from activating its own request. As an output, the MPC asserts BREQ to request access to the PSB bus.

ARB<5..0> (Arbitration)
ARB<5..0> are the arbitration signals for the PSB bus. At the MPC interface, these are bidirectional, open-collector signals that connect directly to the PSB bus. ARB<5..0> are used during normal operation to identify the mode and arbitration priority of an agent during an arbitration cycle to facilitate the arbitration process. During system initialization (while reset is active), the Central Services Module (CSM) drives these signals to initialize slot and arbitration IDs.

4.1.2 ADDRESS/DATA BUS SIGNAL GROUP

This signal group includes a 32-bit multiplexed address/data path that interfaces to the PSB address/data bus. The MPC also includes the byte parity signals present on the PSB bus BPAR<3..0>. All signals in this group interface with the PSB bus through bus tranceivers. For the MPC, this signal group also includes signals to control these bus transceivers (ADDIR and REFADR). These signals are described next.

BAD<31..0> (Buffered Address/Data)
BAD<31..0> are the 32 buffered, multiplexed address/data signals that are bidirectional and provide the interface to the PSB address/data bus. At the MPC, these lines should be connected to the equivalent PSB bus AD signals using 74F245 or equivalent transceivers.

BPAR<3..0> (Buffered Parity)
BPAR are four signals that provide parity for the 4 bytes of the BAD bus. These bidirectional lines connect to the PSB bus PAR<3..0> signals through a 74F245 or equivalent transceiver. These signals are used to receive byte parity for incoming operations and to drive byte parity for outgoing operations.

ADDIR (Address/Data Direction)
ADDIR is an output that provides direction control over the transceivers driving and receiving BAD<31..0> and BPAR<3..0>. In the high state, this signal causes the transceivers to place address/data information along with parity onto the PSB bus. In the low state, this signal causes address/data information and parity to be received from the PSB bus.

REFADR (Reference Address Enable)
REFADR is an output used to enable external address buffers. Asserting this signal places address information from the local bus onto BAD. The address path enabled by this signal is used for memory and I/O references to the PSB bus and is not used during message passing or for references to interconnect space on the PSB bus.

4.1.3 SYSTEM CONTROL SIGNAL GROUP

The MPC provides signals that are used to interface to the System Control Signal Group of the PSB bus. These signals are described next.

BSC<9..0> (Buffered System Control)
BSC<9..0> is a group of ten bidirectional signals that interface to the System Control Signal Group of the PSB bus through 74F245 or equivalent transceivers. Direction control of the transceivers is provided by SCDIR<1, 0> (discussed next). Agents on the PSB bus use the System Control Signal Group to define commands or report status, depending on the phase of the operation. See the MULTIBUS II Architecture Specification for more information on these signals.

SCDIR<1, 0> (System Control Direction)
SCDIR<1, 0> are output signals that provide direction control of the 74F245 transceivers driving and receiving BSC<9..0>. SCDIR0 provides control for BSC<9, 3..0>, while SCDIR1 provides control for

BSC<8..4>. When either signal is high, the corresponding five bits of the BSC signal group are driven onto the PSB bus. When either signal is low, the corresponding five bits on the PSB bus are driven onto the BSC signal group.

4.1.4 CENTRAL CONTROL SIGNAL GROUP

The MPC provides several signals that interface directly or through transceivers to the Central Control signal group of the PSB bus. These signals are described next.

BBCLK (Buffered Bus Clock)
BBCLK is buffered from the PSB bus BCLK signal. This signal should be connected to BCLK using a 74AS1804 or equivalent inverting buffer. This clock is used for all synchronous internal MPC timing.

LACHn (Latch n)
LACHn is an input signal used during initialization of slot and arbitration IDs (where "n" is the slot number). When the RESET signal is active, LACHn asserted indicates to an agent that a slot or arbitration ID is available and should be latched. LACHn is an active high input and should be connected to the LACHn signal on the PSB bus with a 74AS1804 or equivalent inverting buffer.

RESET
RESET is an input that, when asserted, places the MPC in a known state. Only the parts of the MPC involved with initialization of slot and arbitration IDs remain unaffected. RESET is an active high input and should be connected to the RST signal on the PSB bus with a 74AS1804 or equivalent inverting buffer.

Reset Condition
Table 4-1 summarizes the states of the signals while the RESET signal is active.

Table 4-1. Signal State During Reset

Signal	Reset State
BREQ, ARB<5..0>	Z
BAD<31..0>	Z
ADDIR	L
REFADR	H
BSC<9..0>	Z
SCDIR<1, 0>	L
BUSERR	Z(H)
RSTNC	L
SEL	H
D<31..0>	Z
WAIT	H
MINT, EINT	L
ODREQ, IDREQ	L

NOTES:
H — Electrical High State
L — Electrical Low State
Z — High Impedence

RSTNC (Reset Not Complete)
RSTNC is a bidirectional, open-collector signal with high-current drive. It connects directly to the PSB bus. As an input, RSTNC inhibits the MPC from initiating PSB bus operations. As an output, the MPC asserts RSTNC to prevent PSB bus operation until the agent is finished with initialization. The MPC asserts RSTNC whenever the RST signal is asserted by the Central Services Module (CSM). After the CSM deasserts RST and initialization of the local agent is complete, the interconnect microcontroller writes to a register within the MPC. The MPC then deasserts RSTNC.

4.1.5 EXCEPTION OPERATION SIGNAL GROUP

The MPC interfaces with both signals of the Exception Operation Signal Group (part of the PSB bus), as described below.

BUSERR (Bus Error)
BUSERR is a bidirectional, open-collector signal with high-current drive. It connects directly to the PSB bus. As an input, the MPC uses this signal to detect bus errors signaled by other agents. As an output, the MPC uses BUSERR to indicate parity errors detected on either the BAD or BSC signals and to indicate handshake protocol violations detected on the BSC signals.

TIMOUT (Time-Out)
TIMOUT is an input from the PSB bus used to detect a time-out condition signaled by the CSM. TIMOUT is an active high input to the MPC and must be connected to the TIMOUT signal of the PSB bus through a 74AS1804 or equivalent inverter buffer.

4.2 Dual-Port Memory Control Signals

The MPC provides the following signals to support dual-port memory.

SEL (Select)
The MPC asserts SEL to indicate that a dual-port memory access is in progress. The assertion of SEL initiates the dual-port operation and during memory reads, can be used to enable the dual-port data buffers onto the BAD bus. When the MPC completes the PSB bus handshake on the PSB bus, or if the MPC detects an exception, it deasserts SEL.

COM (Complete)
COM is an input to the MPC. The dual-port memory controller asserts COM to indicate it is ready to complete dual-port access. COM is assumed to be synchronous to the bus clock. The MPC asserts the Replier Ready (SC4) signal on the PSB bus on the bus clock after the memory controller has asserted COM. The memory controller cannot deassert COM until the end-of-transfer (EOT) handshake is complete on the PSB bus. This requires that the memory controller monitor the PSB bus for the EOT handshake.

ERR (Error)
ERR is asserted by the dual-port memory controller to signal a memory data parity error. ERR must be stable (high or low) whenever COM is asserted. The MPC responds to this signal by completing the replier handshake on the PSB bus using a "data error" agent error code. This signal may be asynchronous to the bus clock since it is qualified by the COM signal.

4.3 Local Bus Signals

The MPC provides five signal groups that together interface to the CPU's local bus. These local bus signal groups are:

- data
- address and select
- transfer control
- interrupt
- DMA control

All local bus signals are assumed to be asynchronous to the bus clock.

4.3.1 DATA BUS

The local data bus (D<31..0>) is a bidirectional group of signals that transfers data between the host CPU, DMA controller, or memory and the MPC. Although this is a 32-bit interface, the MPC provides control to allow operation with processors using 8-, 16-, or 32-bit data buses.

Not all processors use the same byte order when performing multiple data byte operations. For example, for a 16-bit write to memory, one processor may carry the least-significant byte on local bus bits D<7..0> and the most-significant byte on bits D<15..8>, while another processor may carry the least-significant byte on bits D<15..8> and the most-significant byte on bits D<7..0>. For a given agent, be sure to implement the processor interface to maintain consistent byte addressability with all other agents in the system.

4.3.2 ADDRESS AND SELECT SIGNALS

The address and status signals identify all MPC operations over the local bus.

A<5..2> (Address)
The address inputs select MPC registers for message and interconnect space operations. A1 and A0 are omitted to provide a consistent register address for all data bus width options. These signals are qualified by commands in the MPC (for example, RD or WR, defined in section 4.1.3.3). To the MPC, the state of A<5..2> must be stable within the specified setup and hold window. The address values defined by A<5..2>, and required to access MPC registers, are provided in, "Programming the Host Interface" of the MPC User's Manual, Part Number: 176526.

BE<3..0> (Byte Enable)
These input signals identify valid bytes for memory and I/O reference operations and also provide data path control for register and DMA operations. The assertion of a byte enable signal validates a particular byte on the data bus. Signals BE<3..0> correspond to data bytes 3 through 0 on the data bus (where byte 3 is D<31..24>). Only combinations supported by the PSB bus specification are valid. Valid combinations are summarized in Table 4-2. Values not shown in the table are illegal and will result in unpredictable operation. These signals are qualified by commands (for example, RD or WR) in the MPC and must be stable within the specified setup and hold window.

Operation with 32-bit local buses requires that all byte enable and data signals are used. For 16-bit local buses, BE2 and BE33 are deasserted, BE1 and BE2 are used to indicate which of the two bytes will contain valid data, and only D<15..0> are used. For 8-bit local bus operations, BE3 is asserted, BE2 is deasserted, and BE1 and BE0 are used to select which byte of the PSB bus will carry the valid data byte. This mode uses only D<7..0> (on the local bus). Note that during all read operations, the MPC drives D<31..0>.

Table 4-2. Valid Byte Enable Combinations

				Local Bus				PSB Bus†			
BE3	BE2	BE1	BE0	D31–D24	D23–D16	D15–D8	D7–D0	AD31–AD24	AD23–AD16	AD15–AD8	AD7–AD0
L	L	L	L	V3	V2	V1	V0	V3	V2	V1	V0
L	L	L	H	V3	V2	V1	x	V3	V2	V1	x
H	L	L	L	x	V2	V1	V0	x	V2	V1	V0
L	L	H	H	V3	V1	x	x	x	x	V3	V2
H	L	L	H	x	V2	V1	x	x	V2	V1	x
H	H	L	L	x	x	V1	V0	x	x	V1	V0
L	H	H	H	V3	x	x	x	x	x	V3	x
H	L	H	H	x	V2	x	x	x	x	x	V2
H	H	L	H	x	x	V1	x	x	x	V1	x
H	H	H	L	x	x	x	V0	x	x	x	V0
L	H	L	H	x	x	x	V0	x	x	V0	x
L	H	H	L	x	x	x	V0	x	x	x	V0

NOTES:
L — Electrical low state (active)
H — Electrical high state (inactive)
Vx — Valid data bytes
x — Active bytes with undefined data
† — For this PSB bus, these combinations apply to reference operations, not message space operations

MEMSEL (Memory Select)
This MPC input signal, when asserted, indicates to the MPC that the current operation is a memory reference to the PSB bus. It is qualified by the assertion of RD or WR (defined in section 4.3.3). The state of MEMSEL be must stable within the defined set-up and hold window. Additionally, for MEMSEL to be valid, the signals IOSEL, REGSEL, IDACK, and ODACK must not be active during the same setup and hold window. (IDACK and ODACK are defined later in section 4.3.5.)

IOSEL (I/O Select)
This input signal, when asserted, indicates to the MPC that the current operation is an I/O reference to the PSB bus. It is qualified by the assertion of RD or WR (defined in section 4.3.3). The state of IOSEL must be stable within the defined setup and hold window. Additionally, for IOSEL to be valid, the signals MEMSEL, REGSEL, IDACK, and ODACK must not be active during the same setup and hold window. (IDACK and ODACK are defined later in section 4.3.5.)

REGSEL (Register Select)

This input signal, when asserted, identifies an operation as an MPC-register access. The host CPU asserts REGSEL to set up the MPC for message or interconnect space operations and these are mapped as register operations. REGSEL is qualified by the assertion of RD or WR (defined in section 4.3.3). The state of REGSEL must be stable within the defined setup and hold window. Additionally, for REGSEL to be valid, the signals MEMSEL, IOSEL, IDACK, and ODACK must not be active during the same setup and hold window. (IDACK and ODACK are defined later in section 4.3.5).

LOCK
This input signal allows back-to-back operations to be performed on the PSB bus or to local interconnect space. When LOCK is asserted, any resource accessed by the operation (PSB bus or local interconnect space) is locked until LOCK is deasserted.

4.3.3 TRANSFER CONTROL SIGNALS

Transfer control to the MPC over the local bus is provided by two command signals (Read and Write) and a wait signal. This handshake provides fully interlocked (two-sided handshake) operation.

RD (Read)
This input signal, when asserted, generally initiates a read operation. The CPU asserts RD to initiate read operations of MPC registers. The CPU also asserts RD to initiate read operations of I/O and memory locations present on the PSB bus. The DMA controller asserts RD to qualify DMA cycles. In this last case, the MPC does not interpret RD as an indicator of the data transfer direction, but only to qualify the DMA acknowledge signal (see definitions for ODACK and IDACK). RD must transition cleanly, since it is used to latch other signals that define the parameters of the operation.

WR (Write)

This input signal, when asserted, generally initiates a write operation. The CPU asserts WR to initiate write operations of MPC registers. The CPU also asserts WR to initiate write operations of I/O and memory locations present on the PSB bus. The DMA controller asserts WR to qualify DMA cycles. In this last case, the MPC does not interpret WR as an indicator of the data transfer direction, but only to qualify the DMA acknowledge signal (see definitions for ODACK and IDACK). WR must transition cleanly, since it is used to qualify other signals that define the parameters of the operation.

WAIT

WAIT is an MPC output signal used to delay (or suspend) a local bus operation during an access to an asynchronous resource via the MPC. The MPC asserts WAIT to the local CPU for memory, I/O, and interconnect accesses to the PSB bus; and for local interconnect accesses. WAIT, when asserted, allows time for the accessed resource to become available. The MPC asserts WAIT after the CPU has asserted the command signal (RD or WR). On the PSB bus, the MPC deasserts WAIT after either the PSB bus EOT handshake or an exception has occurred. For accesses to local interconnect space, the MPC deasserts WAIT after the interconnect operation is complete.

4.3.4 INTERRUPT SIGNALS

Interrupt signals are used to inform the host CPU that the MPC requires service. The MPC provides two signals: one for message operations and one for reference errors.

MINT (Message Interrupt)

The MPC asserts this output signal for all message-related signaling to the host CPU. This includes the arrival of an unsolicited message, an available transmit FIFO buffer, the completion of a solicited transfer, and an error on message transfer.

EINT (Error Interrupt)

The MPC asserts this output signal to the CPU to indicate errors related to memory, I/O, or interconnect space operations (i.e., all except message operations). Internal registers in the MPC provide details of the error via interconnect space.

4.3.5 DMA CONTROL SIGNALS

The MPC provides several DMA control signals to support an external DMA controller. A DMA controller is required to support solicited message operations.

ODREQ (Output Channel DMA Request)

ODREQ is an output signal the MPC asserts to enable DMA transfer of data to the MPC (i.e., output to the PSB bus). This signal behaves as a normal DMA request line. For a solicited message output operation, the MPC asserts ODREQ when a solicited output packet buffer is empty and as long as the MPC is in the transfer phase (the Buffer Request unsolicited message has been sent). The DMA controller responds by performing DMA transfers to the MPC for transfer to the receiving agent.

IDREQ (Input Channel DMA Request)

The MPC asserts this output signal to enable DMA transfer of data from the MPC (i.e., input from the PSB bus). This signal behaves as a normal DMA request line. For a solicited message input operation, the MPC asserts IDREQ after a solicited input packet buffer is full and as long as the MPC is in the transfer phase. The DMA controller responds by performing DMA transfer from the MPC. IDREQ remains asserted until the packet is transferred to memory.

ODACK (Output Channel DMA Acknowledge)

ODACK is an input signal asserted by the DMA controller in response to the assertion of ODREQ by the MPC. The DMA controller asserts ODACK to set up the MPC for the DMA transfer from local memory (or the controller) to the MPC. The assertion of ODACK is qualified by the assertion of RD or WR by the MPC. The direction of data transfer with respect to the MPC is controlled by the request signal (IDREQ or ODREQ) and the acknowledge signal (IDACK or ODACK). The command signal (RD or WR) only qualifies the acknowledge signal (IDACK or ODACK). The state of ODACK must be stable within the defined setup and hold window. Additionally, for ODACK to be valid, the signals MEMSEL, IOSEL, REGSEL, and IDACK must not be active during the same setup and hold window.

IDACK (Input Channel DMA Acknowledge)

IDACK is an input signal asserted by the DMA controller in response to the assertion of IDREQ by the MPC. The DMA controller asserts IDACK to set up the MPC for the DMA transfer from the MPC to the DMA controller (or local memory). The assertion of IDACK is qualified by the assertion of RD or WR by the DMA controller. The direction of data transfer with respect to the MPC is controlled by the request signal (IDREQ or ODREQ) and the acknowledge signal (IDACK or ODACK). The command signal (RD or WR) only qualifies the acknowledge signal. The state of IDACK must be stable within the defined setup and hold window. Additionally, for IDACK to be valid, the signals MEMSEL, IOSEL, REGSEL, and ODACK must not be active during the same setup and hold window.

4.4 Interconnect Bus Signals

The interconnect bus signals provide a simple interface to a microcontroller for implementation of interconnect space. All interconnect bus signals are asynchronous to the bus clock and to the local bus signals.

IAD<7..0> (Interconnect Address/Data)
IAD<7..0> is an 8-bit, bidirectional, multiplexed address and data bus intended to interface directly to a microcontroller. In addition to the MPC, other interconnect registers can be connected to this bus.

IREQ (Interconnect Request)
The MPC asserts this output signal when an interconnect operation has been requested from either the local bus or the PSB bus. The MPC deasserts IREQ after the microcontroller has written to the Interconnect Reference Arbitration register.

IAST (Interconnect Address Strobe)
IAST is an input signal from the microcontroller and, when asserted, indicates that a valid address is on the interconnect bus. IAST may be directly connected to the ALE (Address Latch Enable or equivalent) output of most microcontrollers. IAST must provide clean transitions.

IRD (Interconnect Bus Read)
IRD is an input signal. The microcontroller asserts IRD to perform a read operation to one of the MPC interconnect interface registers. IRD must provide clean transitions. When IRD is asserted in conjunction with the IWR signal, all MPC outputs are disabled.

IWR (Interconnect Write)
IWR is an input signal. The microcontroller asserts IWR to perform a write operation to one of the MPC interconnect interface registers. IWR must provide clean transitions. When IWR is asserted in conjunction with the IRD signal, all MPC outputs are disabled.

4.5 Power and Ground Signals

The MPC requires supply voltage and ground connections at the pin numbers listed below.

V_{CC}	Ground
D4	J3
M4	N4
N8	N6
M12	N9
D12	N11
C7	N13
	K13
	F13
	C12
	D8
	C5
	C3

5.0 MPC 82389 MECHANICAL DATA

The MPC 82389 is packaged in a 149 lead pin grid array. The square package has a 15 x 15 grid layout with the outer 3 rows used along each edge.

5.1 Pin Assignment

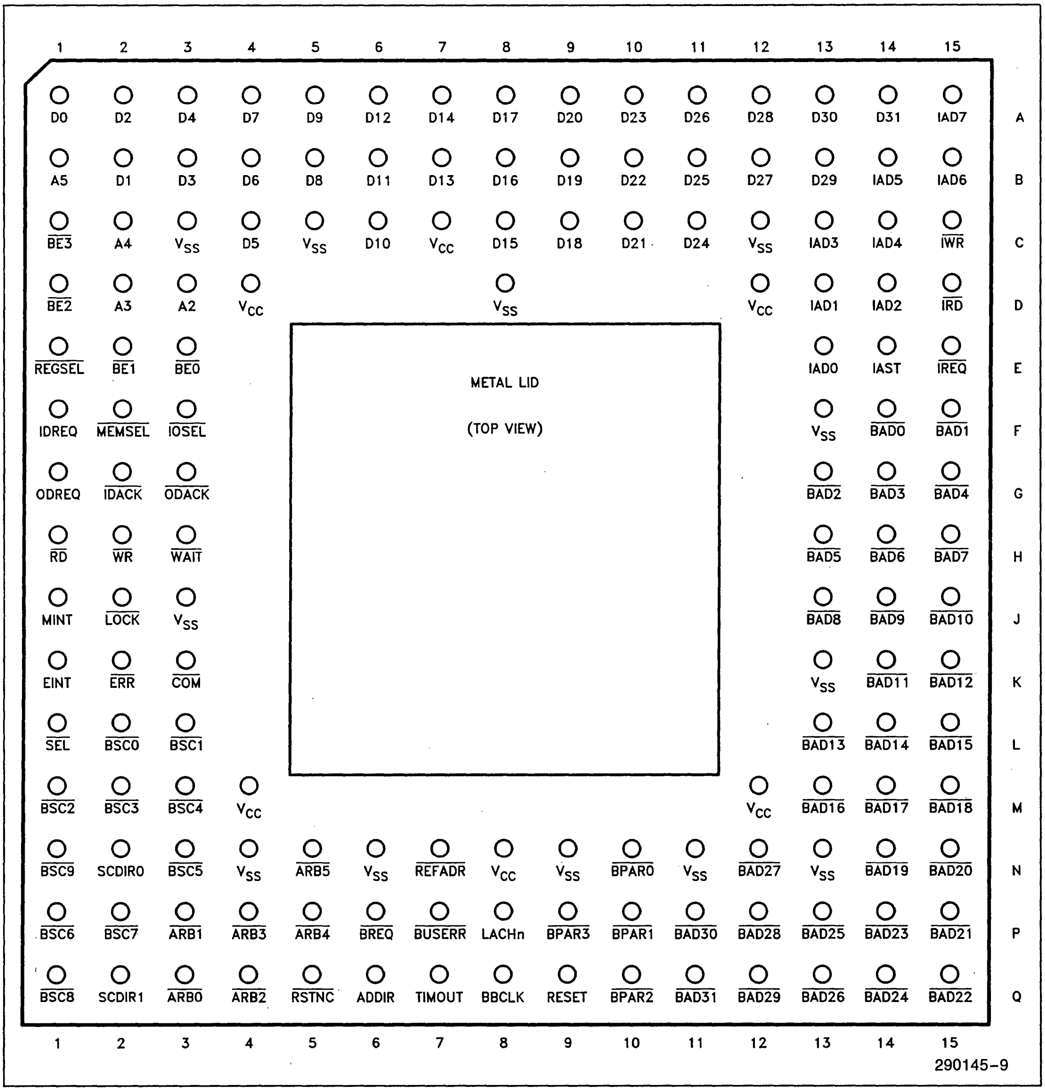
The MPC 82389 pinout as viewed from the top side of the component is shown in Figure 5-1. When viewed from the pin side, the component pin layout is shown in Figure 5-2.

To reduce possible noise problems on the board, V_{CC} and V_{SS} must be connected to multiple supplies. The board should be laid out with V_{CC} and Ground planes for power distribution and the components V_{CC} and V_{SS} must be connected to the appropriate power plane.

5.2 Package Dimensions

The 82389 is packaged in a 149-pin Ceramic Pin Grid Array (PGA). The pins are arranged 0.100 inch (2.54 mm) center-to-center, in a 15 x 15 matrix. Please refer to Figure 5-3 for case outlines.

A wide variety of sockets are available including the zero-insertion force socket for prototyping.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	D0	D2	D4	D7	D9	D12	D14	D17	D20	D23	D26	D28	D30	D31	IAD7
B	A5	D1	D3	D6	D8	D11	D13	D16	D19	D22	D25	D27	D29	IAD5	IAD6
C	$\overline{BE3}$	A4	V_{SS}	D5	V_{SS}	D10	V_{CC}	D15	D18	D21	D24	V_{SS}	IAD3	IAD4	$\overline{IWR}$
D	$\overline{BE2}$	A3	A2	V_{CC}				V_{SS}				V_{CC}	IAD1	IAD2	$\overline{IRD}$
E	$\overline{REGSEL}$	$\overline{BE1}$	$\overline{BE0}$										IAD0	IAST	$\overline{IREQ}$
F	IDREQ	$\overline{MEMSEL}$	$\overline{IOSEL}$										V_{SS}	$\overline{BAD0}$	$\overline{BAD1}$
G	ODREQ	$\overline{IDACK}$	$\overline{ODACK}$										$\overline{BAD2}$	$\overline{BAD3}$	$\overline{BAD4}$
H	$\overline{RD}$	$\overline{WR}$	$\overline{WAIT}$										$\overline{BAD5}$	$\overline{BAD6}$	$\overline{BAD7}$
J	MINT	$\overline{LOCK}$	V_{SS}										$\overline{BAD8}$	$\overline{BAD9}$	$\overline{BAD10}$
K	EINT	$\overline{ERR}$	$\overline{COM}$										V_{SS}	$\overline{BAD11}$	$\overline{BAD12}$
L	$\overline{SEL}$	$\overline{BSC0}$	$\overline{BSC1}$										$\overline{BAD13}$	$\overline{BAD14}$	$\overline{BAD15}$
M	$\overline{BSC2}$	$\overline{BSC3}$	$\overline{BSC4}$	V_{CC}								V_{CC}	$\overline{BAD16}$	$\overline{BAD17}$	$\overline{BAD18}$
N	$\overline{BSC9}$	SCDIR0	$\overline{BSC5}$	V_{SS}	$\overline{ARB5}$	V_{SS}	$\overline{REFADR}$	V_{CC}	V_{SS}	$\overline{BPAR0}$	V_{SS}	$\overline{BAD27}$	V_{SS}	$\overline{BAD19}$	$\overline{BAD20}$
P	$\overline{BSC6}$	$\overline{BSC7}$	$\overline{ARB1}$	$\overline{ARB3}$	$\overline{ARB4}$	$\overline{BREQ}$	$\overline{BUSERR}$	LACHn	$\overline{BPAR3}$	$\overline{BPAR1}$	$\overline{BAD30}$	$\overline{BAD28}$	$\overline{BAD25}$	$\overline{BAD23}$	$\overline{BAD21}$
Q	$\overline{BSC8}$	SCDIR1	$\overline{ARB0}$	$\overline{ARB2}$	$\overline{RSTNC}$	ADDIR	TIMOUT	BBCLK	RESET	$\overline{BPAR2}$	$\overline{BAD31}$	$\overline{BAD29}$	$\overline{BAD26}$	$\overline{BAD24}$	$\overline{BAD22}$

Figure 5-1. MPC 82389 Pinout—View from Top Side

(BOTTOM VIEW)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
A	IAD7	D31	D30	D28	D26	D23	D20	D17	D14	D12	D9	D7	D4	D2	D0
B	IAD6	IAD5	D29	D27	D25	D22	D19	D16	D13	D11	D8	D6	D3	D1	A5
C	$\overline{\text{IWR}}$	IAD4	IAD3	V_{SS}	D24	D21	D18	D15	V_{CC}	D10	V_{SS}	D5	V_{SS}	A4	$\overline{\text{BE3}}$
D	$\overline{\text{IRD}}$	IAD2	IAD1	V_{CC}				V_{SS}				V_{CC}	A2	A3	$\overline{\text{BE2}}$
E	$\overline{\text{IREQ}}$	IAST	IAD0										$\overline{\text{BE0}}$	$\overline{\text{BE1}}$	$\overline{\text{REGSEL}}$
F	$\overline{\text{BAD1}}$	$\overline{\text{BAD0}}$	V_{SS}										$\overline{\text{IOSEL}}$	$\overline{\text{MEMSEL}}$	IDREQ
G	$\overline{\text{BAD4}}$	$\overline{\text{BAD3}}$	$\overline{\text{BAD2}}$										$\overline{\text{ODACK}}$	$\overline{\text{IDACK}}$	ODREQ
H	$\overline{\text{BAD7}}$	$\overline{\text{BAD6}}$	$\overline{\text{BAD5}}$										$\overline{\text{WAIT}}$	$\overline{\text{WR}}$	$\overline{\text{RD}}$
J	$\overline{\text{BAD10}}$	$\overline{\text{BAD9}}$	$\overline{\text{BAD8}}$										V_{SS}	$\overline{\text{LOCK}}$	MINT
K	$\overline{\text{BAD12}}$	$\overline{\text{BAD11}}$	V_{SS}										$\overline{\text{COM}}$	$\overline{\text{ERR}}$	EINT
L	$\overline{\text{BAD15}}$	$\overline{\text{BAD14}}$	$\overline{\text{BAD13}}$										$\overline{\text{BSC1}}$	$\overline{\text{BSC0}}$	$\overline{\text{SEL}}$
M	$\overline{\text{BAD18}}$	$\overline{\text{BAD17}}$	$\overline{\text{BAD16}}$	V_{CC}								V_{CC}	$\overline{\text{BSC4}}$	$\overline{\text{BSC3}}$	$\overline{\text{BSC2}}$
N	$\overline{\text{BAD20}}$	$\overline{\text{BAD19}}$	V_{SS}	$\overline{\text{BAD27}}$	V_{SS}	$\overline{\text{BPAR0}}$	V_{SS}	V_{CC}	$\overline{\text{REFADR}}$	V_{SS}	$\overline{\text{ARB5}}$	V_{SS}	$\overline{\text{BSC5}}$	SCDIR0	$\overline{\text{BSC9}}$
P	$\overline{\text{BAD21}}$	$\overline{\text{BAD23}}$	$\overline{\text{BAD25}}$	$\overline{\text{BAD28}}$	$\overline{\text{BAD30}}$	$\overline{\text{BPAR1}}$	$\overline{\text{BPAR3}}$	LACHn	$\overline{\text{BUSERR}}$	$\overline{\text{BREQ}}$	$\overline{\text{ARB4}}$	$\overline{\text{ARB3}}$	$\overline{\text{ARB1}}$	$\overline{\text{BSC7}}$	$\overline{\text{BSC6}}$
Q	$\overline{\text{BAD22}}$	$\overline{\text{BAD24}}$	$\overline{\text{BAD26}}$	$\overline{\text{BAD29}}$	$\overline{\text{BAD31}}$	$\overline{\text{BPAR2}}$	RESET	BBCLK	TIMOUT	ADDIR	$\overline{\text{RSTNC}}$	$\overline{\text{ARB2}}$	$\overline{\text{ARB0}}$	SCDIR1	$\overline{\text{BSC8}}$
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

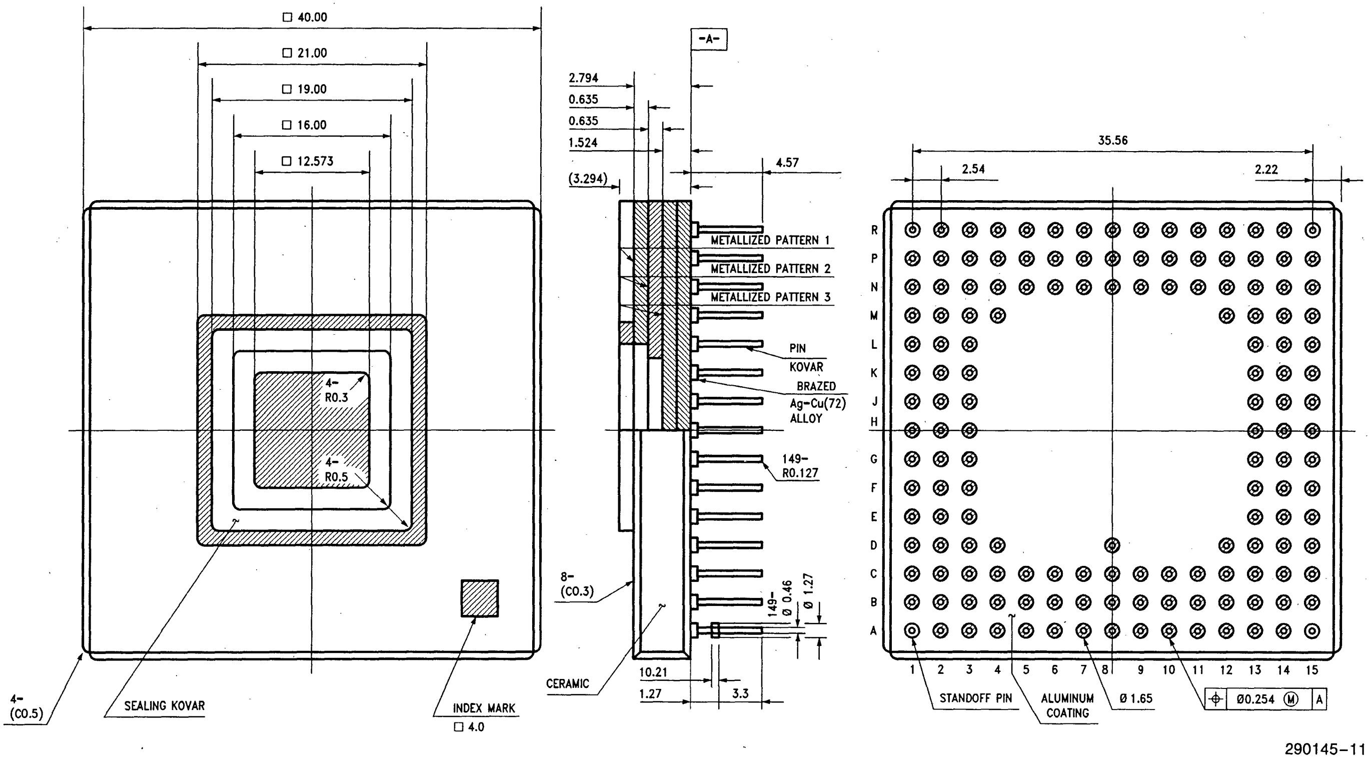
290145-10

Figure 5-2. MPC 82389 Pinout—View from Pin Side

Table 5-1

Signal and Characteristic		Pin #	Signal and Characteristic		Pin #	Signal and Characteristic		Pin #
V_{CC}		D4	REFADR	O	N7	IAST	I	E14
A5	I	B1	ADDIR	O	Q6	$\overline{\text{IRD}}$	I	D15
A4	I	C2	$\overline{\text{BPAR3}}$	I/O	P9	$\overline{\text{IWR}}$	I	C15
A3	I	D2	$\overline{\text{BAD31}}$	I/O	Q11	IAD7	I/O	A15
A2	I	D3	$\overline{\text{BAD30}}$	I/O	P11	IAD6	I/O	B15
$\overline{\text{BE3}}$	I	C1	$\overline{\text{BAD29}}$	I/O	Q12	IAD5	I/O	B14
$\overline{\text{BE2}}$	I	D1	$\overline{\text{BAD28}}$	I/O	P12	IAD4	I/O	C15
$\overline{\text{BE1}}$	I	E2	$\overline{\text{BAD27}}$	I/O	N12	IAD3	I/O	C13
$\overline{\text{BE0}}$	I	E3	$\overline{\text{BAD26}}$	I/O	Q13	IAD2	I/O	D14
$\overline{\text{IOSEL}}$	I	F3	$\overline{\text{BAD25}}$	I/O	P13	IAD1	I/O	D13
$\overline{\text{MEMSEL}}$	I	F2	$\overline{\text{BAD24}}$	I/O	Q14	IAD0	I/O	E13
$\overline{\text{REGSEL}}$	I	E1	$\overline{\text{BAD23}}$	I/O	P14	V_{CC}		D12
$\overline{\text{IDACK}}$	I	G2	$\overline{\text{BAD22}}$	I/O	Q15	V_{SS}		C12
$\overline{\text{ODACK}}$	I	G3	$\overline{\text{BAD21}}$	I/O	P15	D31	I/O	A14
IDREQ	O	F1	$\overline{\text{BAD20}}$	I/O	N15	D30	I/O	A13
ODREQ	O	G1	$\overline{\text{BAD19}}$	I/O	N14	D29	I/O	B13
$\overline{\text{WR}}$	I	H2	$\overline{\text{BAD18}}$	I/O	M15	D28	I/O	A12
$\overline{\text{RD}}$	I	H1	$\overline{\text{BAD17}}$	I/O	M14	D27	I/O	B12
$\overline{\text{WAIT}}$	O	H3	$\overline{\text{BAD16}}$	I/O	M13	D26	I/O	A11
V_{SS}		J3	$\overline{\text{BAD15}}$	I/O	Q14	D25	I/O	B11
MINT	O	J1	$\overline{\text{BAD14}}$	I/O	L14	D24	I/O	C11
EINT	O	K1	$\overline{\text{BAD13}}$	I/O	L13	D23	I/O	A10
$\overline{\text{LOCK}}$	I	J2	$\overline{\text{BAD12}}$	I/O	K15	D22	I/O	B10
$\overline{\text{ERR}}$	O	K2	$\overline{\text{BAD11}}$	I/O	K14	D21	I/O	C10
$\overline{\text{SEL}}$	O	L1	$\overline{\text{BAD10}}$	I/O	J15	D20	I/O	A9
$\overline{\text{COM}}$	I	K3	$\overline{\text{BAD9}}$	I/O	J14	D19	I/O	B9
$\overline{\text{BSC9}}$	I/O	N1	$\overline{\text{BAD8}}$	I/O	J13	D18	I/O	C9
$\overline{\text{BSC8}}$	I/O	Q1	$\overline{\text{BAD7}}$	I/O	H15	D17	I/O	A8
$\overline{\text{BSC7}}$	I/O	P2	$\overline{\text{BAD6}}$	I/O	H14	D16	I/O	B8
$\overline{\text{BSC6}}$	I/O	P1	$\overline{\text{BAD5}}$	I/O	H13	D15	I/O	C8
$\overline{\text{BSC5}}$	I/O	N3	$\overline{\text{BAD4}}$	I/O	G15	D14	I/O	A7
$\overline{\text{BSC4}}$	I/O	M3	$\overline{\text{BAD3}}$	I/O	G14	D13	I/O	B7
$\overline{\text{BSC3}}$	I/O	M2	$\overline{\text{BAD2}}$	I/O	G13	D12	I/O	A6
$\overline{\text{BSC2}}$	I/O	M1	$\overline{\text{BAD1}}$	I/O	F15	D11	I/O	B6
$\overline{\text{BSC1}}$	I/O	L3	$\overline{\text{BAD0}}$	I/O	F14	D10	I/O	C6
$\overline{\text{BSC0}}$	I/O	L2	$\overline{\text{BPAR2}}$	I/O	Q10	D9	I/O	A5
SCDIR1	O	Q2	$\overline{\text{BPAR1}}$	I/O	P10	D8	I/O	B5
SCDIR0	O	N2	$\overline{\text{BPAR0}}$	I/O	N10	D7	I/O	A5
V_{CC}		M4	V_{CC}		N8	D6	I/O	B4
V_{SS}		N4	V_{SS}		N9	D5	I/O	C4
$\overline{\text{ARB5}}$	I/O, OC	A15	V_{SS}		N11	D4	I/O	A3
$\overline{\text{ARB4}}$	I/O, OC	P5	V_{CC}		M12	D3	I/O	B3
$\overline{\text{ARB3}}$	I/O, OC	P4	V_{SS}		N13	D2	I/O	A2
$\overline{\text{ARB2}}$	I/O, OC	Q4	V_{SS}		F13	D1	I/O	B2
$\overline{\text{ARB1}}$	I/O, OC	P3	V_{SS}		K13	D0	I/O	A1
$\overline{\text{ARB0}}$	I/O, OC	Q3	$\overline{\text{BBCLK}}$	I	Q8	V_{CC}		C7
V_{SS}		N6	LACHn	I	P8	V_{SS}		D8
$\overline{\text{BREQ}}$	I/O, OC	P6	RESET	I	Q9	V_{SS}		C5
TIMOUT	O	Q7	$\overline{\text{RSTNC}}$	I/O, OC	Q5	$\overline{\text{BUSERR}}$	I/O, OC	P7
$\overline{\text{IREQ}}$	O	E15	V_{SS}		C3			

NOTES:
I = Input signal
O = Output signal
I/O = Input or output signal
OC = Open-collector signal



NOTE:
Dimensions in mm.

Figure 5-3. 149-Pin PGA Package Dimensions

6.0 MPC 82389 ELECTRICAL DATA

This section provides detailed target A.C. and D.C. specifications for the MPC 82389.

6.1 Maximum Ratings

Operating Temperature (Under Bias)	−10°C to +85°C
Storage Temperature	−65°C to +150°C
Voltage on Any Pin	−0.5V to V_{CC} + 0.5V
Power Dissipation	2.5W

NOTE:
Stresses 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 82389 contains protective circuitry to resist damage from static electrical discharges, always take precautions against high static voltages or electric fields.

6.2 D.C. Specifications V_{CC} = 5.0V ±10%, T_A = 0°C to +70°C

Table 6-1. D.C. Specifications

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_{OL1}	Output Low Voltage		0.45	V	I_{OL} Max
V_{OL2}	Output Low Voltage Open Collector		0.55	V	I_{OL} Max
V_{OH}	Output High Voltage	2.4		V	I_{OH} Max
I_{CC}	Power Supply Current		400	mA	
I_L	Input Leakage Current		±10	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{L1}	Open Collector		±100	μA	0.4V ≤ V_{IN} ≤ 2.4V
	Leakage Current		±400	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{L2}	BBCLK Input Leakage Current		±100	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{OL}	Output Low Current	4.0		mA	V_{OL} = 0.45V
I_{OL1}	Open Collector Output Low Current	60.0		mA	V_{OL} = 0.55V
I_{OL2}	ADDIR and $\overline{\text{REFADR}}$ Output Low Current	8.0		mA	V_{OL} = 0.45V
I_{OH}	Output High Current	−1.0		mA	V_{OH} = 2.4V
C_I	Input Capacitance		10	pF	f_C = 1 MHz, 25°C (Note 1)
C_{IO}	I/O Capacitance		20	pF	f_C = 1 MHz, 25°C (Note 1)
C_{CLK}	Clock Input Capacitance		15	pF	f_C = 1 MHz, 25°C (Note 1)
C_{OC}	Open Collector Capacitance		20	pF	f_C = 1 MHz, 25°C (Note 1)

NOTE:
1. Sampled only, not 100% tested.

6.3 A.C. Specifications

The A.C. specifications for the MPC 82389 are specified in Tables 6-2, 6-3 and 6-4 and Figures 6-2, 6-3, 6-4 and 6-5. Figure 6-1 specifies the test points for measuring the A.C. parameters. Table 6-2 and Figures 6-2 and 6-3 specify the A.C. parameters for the local bus. Table 6-3 and Figure 6-4 specify the A.C. parameters for the interconnect bus. Table 6-4 and Figure 6-5 specify the A.C. parameters for the PSB bus. Figure 6-6 defines the test load for the A.C. specifications.

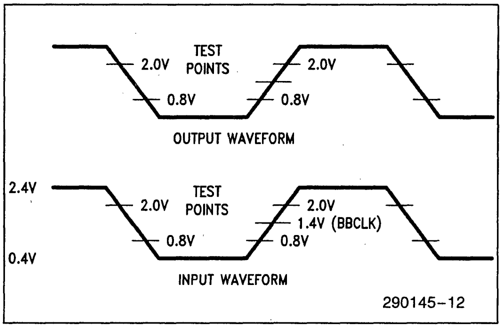


Figure 6-1. A.C. Test Waveforms

Table 6-2. Local Bus A.C. Specifications (V_{CC} = 5V ±10%, T_A = 0°C to +70°C)

Symbol	Parameter	Min	Max	Units	Test Conditions
t_1	Address and $\overline{BE}$ Setup to Command Active	30		ns	
	Select and DACK Setup to Command Active	24		ns	
t_2	Address, $\overline{BE}$, Select and DACK Hold from Command Active	10		ns	
t_3	Time between Commands	35		ns	
t_4	Command Inactive to Read Data Disable (Note 5)		24	ns	
t_5	Read Data Hold from Command Inactive	3		ns	
t_6	Read Data Enable from Command Active	0		ns	
t_7	$\overline{WAIT}$ Active from Command Active		35	ns	C_L = 50 pF
t_8	Command Inactive from $\overline{WAIT}$ Inactive	0		ns	
t_9	$\overline{WAIT}$ Inactive to Read Data Valid		50	ns	C_L = 150 pF
t_{10}	Command Active to Write Data Valid		200	ns	
t_{11}	Write Data Hold from $\overline{WAIT}$ Inactive	0		ns	
t_{12}	Command Active to $\overline{LOCK}$ Active (Note 1)		100	ns	
t_{13}	$\overline{LOCK}$ Hold from $\overline{WAIT}$ Inactive (Note 2)	0		ns	
t_{14}	Command Active Time	70		ns	
t_{15}	Read Data Valid from Command Active		60	ns	C_L = 150 pF
t_{16}	Write Data Setup to Command Inactive —Registers —DMA	 35 25		 ns ns	
t_{17}	Write Data Hold from Command Inactive	5		ns	
t_{18}	Command Active to MINT or DREQ Inactive (Notes 3, 4)		70	ns	C_L = 50 pF
t_{19}	Command Active to DREQ Inactive (Note 4)		45	ns	C_L = 50 pF

NOTES:
1. Required to guarantee locking of resource.
2. Required to guarantee resource remains locked.
3. MINT deassertion only if no other sources are pending.
4. For DREQ inactive timing, t_{19} applies to a normal last transfer deassert condition and t_{18} to an error deassert condition.
5. Disable condition occurs when the output current becomes less than the input leakage specification.

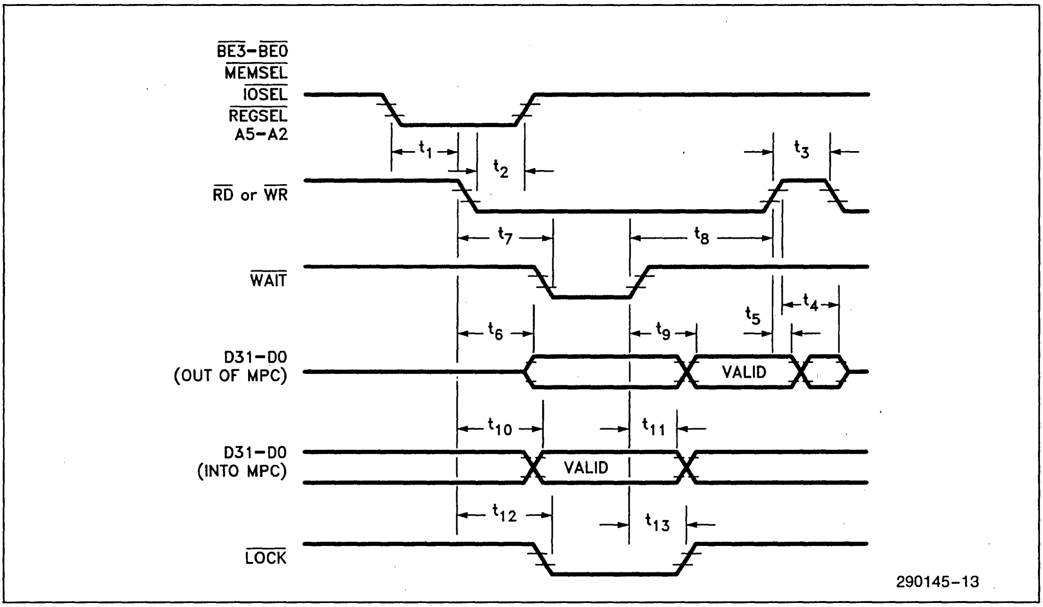


Figure 6-2. Local Bus Reference Operation Timing

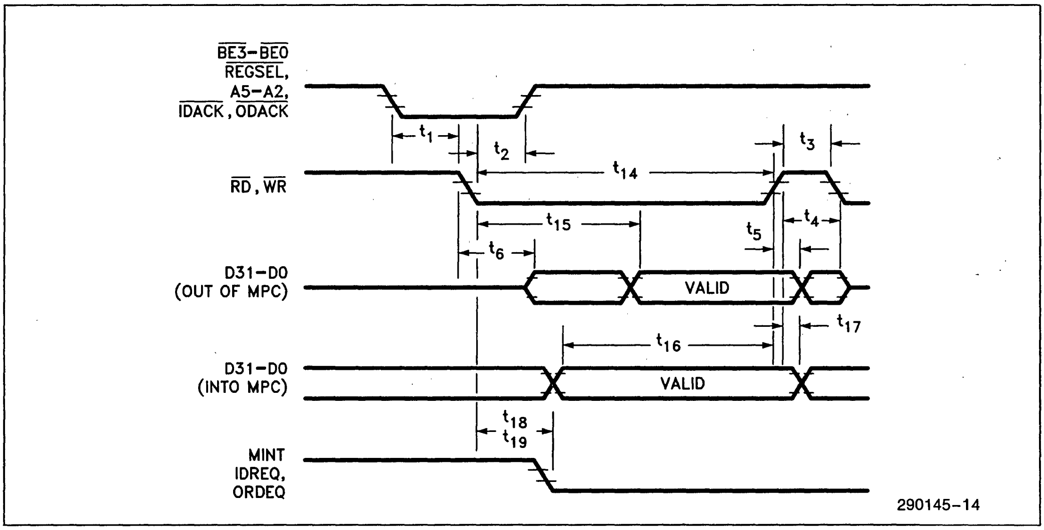


Figure 6-3. Local Bus Register and DMA Operation Timing

Table 6-3. Interconnect Bus A.C. Specifications (V_{CC} = 5V ±10%, T_A = 0°C to +70°C)

Symbol	Parameter	Min	Max	Units	Test Conditions
t_{31}	IAST Active Time	85		ns	
t_{32}	Command Active Time	250		ns	
t_{33}	Command Inactive to IAST Active	25		ns	
t_{33A}	IAST Inactive to Command Active	120		ns	
t_{34}	Address Setup to IAST Inactive	40		ns	
t_{35}	Address Hold from IAST Inactive	20		ns	
t_{36}	Write Data Hold from Command Inactive	120		ns	
t_{37}	Write Data Hold from Command Inactive	5		ns	
t_{38}	Read Data Enable from Command Active	0		ns	
t_{39}	Read Data Valid from Command Active		120	ns	C_L = 150 pF
t_{40}	Read Data Hold from Command Inactive	0		ns	
t_{41}	Read Data Disable from Command Inactive (Note 2)		30	ns	
t_{42}	EINT, $\overline{\text{IREQ}}$ Inactive from Command Active (Note 1)		100	ns	C_L = 150 pF

NOTES:
1. EINT inactive only on write to error register. $\overline{\text{IREQ}}$ inactive only on write to arbitration register.
2. Disable condition occurs when the output current becomes less than the input leakage specification.

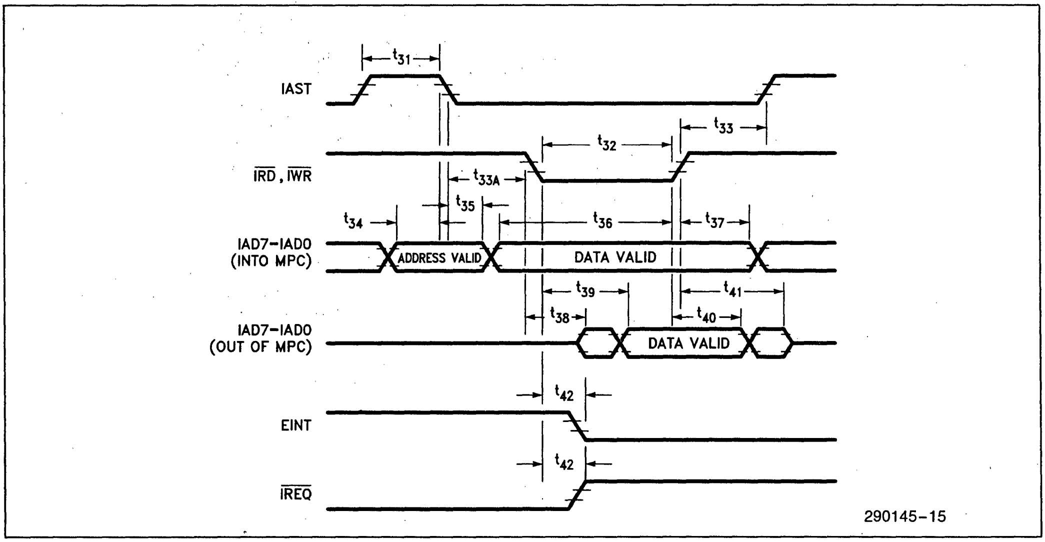


Figure 6-4. Interconnect Bus Timing

Table 6-4. PSB Bus Interface A.C. Specifications (V_{CC} = 5V ±10%, T_A = 0°C to +70°C)

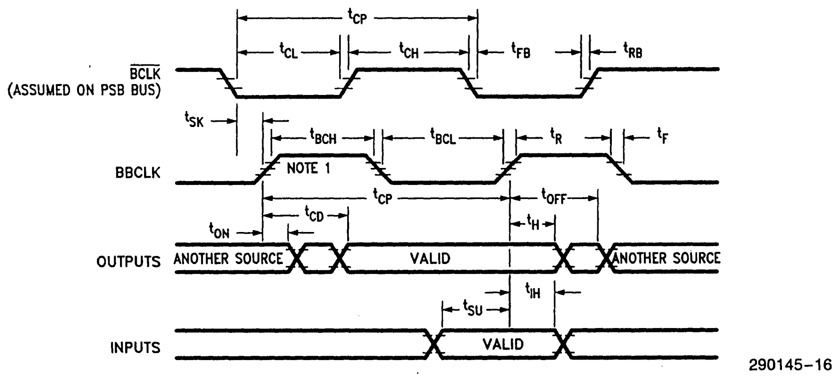
Symbol	Parameter	Min	Max	Units	Test Conditions
t_{CP}	Clock Period	99.9		ns	
t_{CL}	BCLK Low Time	40		ns	
t_{CH}	BCLK High Time	40		ns	
t_{BCL}	BBCLK Low Time	38		ns	
t_{BCH}	BBCLK High Time	38		ns	
t_{RB}	BCLK Rise Time	1	5	ns	
t_{FB}	BCLK Fall Time	1	2	ns	
t_R	BBCLK Rise Time	0.5	1	ns	
t_F	BBCLK Fall Time	0.5	1	ns	
t_{SK}	BCLK to BBCLK Skew (Note 1)	−0.5	4.0	ns	
t_{CD}	Clock to Output Delay				
	BREQ, BUSERR, RSTNC (Note 2)		36	ns	C_L = 500 pF
	ARB5–ARB0 (Notes 2, 3)		36	ns	C_L = 500 pF
	BAD31–BAD0, BSC7–BSC0		29	ns	C_L = 75 pF
	BPAR3–BPAR0, BSC9, BSC8		29	ns	C_L = 50 pF
	SCDIR0, SCDIR1 (H to L)		19	ns	C_L = 25 pF
	(L to H)		21	ns	C_L = 25 pF
	ADDIR (L to H)		21	ns	C_L = 50 pF
	(H to L)		27	ns	C_L = 50 pF
	REFADR		29	ns	C_L = 75 pF
	SEL		29	ns	C_L = 50 pF
t_H	Hold Time from Clock				
	BREQ, BUSERR, RSTNC	6.5		ns	C_L = 25 pF
	ARB5–ARB0 (Note 3)	6.5		ns	C_L = 25 pF
	BAD31–BAD0, BPAR3–BPAR0	5.0		ns	C_L = 15 pF
	BSC9–BSC0	4.0		ns	C_L = 15 pF
	SCDIR0, SCDIR1	4.0		ns	C_L = 15 pF
	ADDIR	5.0		ns	C_L = 25 pF
	REFADR	4.0		ns	C_L = 25 pF
	SEL	4.0		ns	C_L = 15 pF
t_{ON}	Turn On Delay from Clock (Note 4)				
	BREQ, BUSERR, RSTNC	6.5		ns	
	ARB5–ARB0 (Note 1)	6.5		ns	
	BAD31–BAD0, BPAR3–BPAR0	5.0		ns	
	BSC9–BSC0	4.0		ns	
t_{OFF}	Turn Off Delay from Clock (Note 5)				
	BREQ, BUSERR, RSTNC		36	ns	
	ARB5–ARB0 (Note 3)		36	ns	
	BAD31–BAD0, BPAR3–BPAR0		29	ns	
	BSC9–BSC0		29	ns	

Table 6-4. PSB Bus Interface A.C. Specifications (V_{CC} = 5V ±10%, T_A = 0°C to +70°C) (Continued)

Symbol	Parameter	Min	Max	Units	Test Conditions
t_{SU}	Input Setup Time to Clock				
	$\overline{\text{BREQ}}$, $\overline{\text{BUSERR}}$, $\overline{\text{RSTNC}}$	22		ns	
	$\overline{\text{ARB5}}$–$\overline{\text{ARB0}}$ (Note 3)	40		ns	
	$\overline{\text{BAD31}}$–$\overline{\text{BAD0}}$, $\overline{\text{BPAR3}}$–$\overline{\text{BPAR0}}$	24		ns	
	$\overline{\text{BSC9}}$–$\overline{\text{BSC0}}$	24		ns	
	TIMEOUT, LACHn, RESET	24		ns	
	$\overline{\text{COM}}$, $\overline{\text{ERR}}$	40		ns	
t_{IH}	Input Hold Time from Clock				
	$\overline{\text{BREQ}}$, $\overline{\text{BUSERR}}$, $\overline{\text{RSTNC}}$	0		ns	
	$\overline{\text{ARB5}}$–$\overline{\text{ARB0}}$ (Note 3)	0		ns	
	$\overline{\text{BAD31}}$–$\overline{\text{BAD0}}$, $\overline{\text{BPAR3}}$–$\overline{\text{BPAR0}}$	3		ns	
	$\overline{\text{BSC9}}$–$\overline{\text{BSC0}}$	2		ns	
	TIMEOUT, LACHn, RESET	2		ns	
	$\overline{\text{COM}}$, $\overline{\text{ERR}}$	3		ns	

NOTES:

1. The clock timings are provided to reference the MPC specification to the PSB bus specifications. These specifications assume a 74AS1804 or equivalent buffer.
2. The 500 pF load is a distributed load as defined in the PSB bus specification. The open drain signals are designed such that the output delay and bus loss meets the PSB specification requirement.
3. The $\overline{\text{ARB5}}$–$\overline{\text{ARB0}}$ signal timings are with respect to the first and last clock of the arbitration period. Details can be found in the PSB bus specification. Also, the arbitration logic has been designed to meet the loop delay specification accounting for the full path of input to output plus bus loss.
4. Minimum turn on times are measured the same way as hold times. Specifically, the logic level driven by another device on the previous clock cycle must not be disturbed.
5. Maximum turn off times are measured to the condition where the output leakage current becomes less than the input leakage specification.
6. All stated capacitances are based on design requirements. Production test limitations may require some parameters to be tested under a different condition.



NOTE:
1. SAMPOINT point for BBCLK is 1.4V.

Figure 6-5. PSB Bus Interface Timing

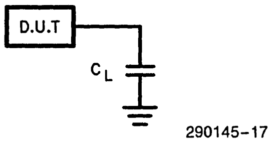


Figure 6-6. A.C. Test Load

7.0 REFERENCE DOCUMENTS

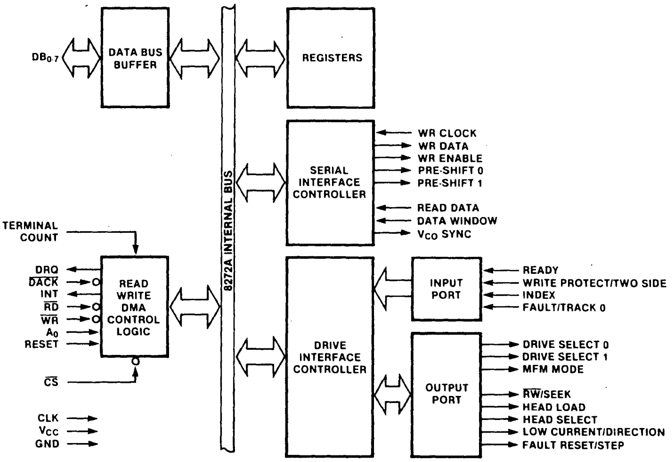
Part Number	Title Description
176526	MPC User's Manual
146077	MULTIBUS® II Architecture Specifications
149299	Interconnect Interface Specifications
149300	MULTIBUS® II MPC External Product Specifications
149247	MULTIBUS® II Transport Protocol Specifications

Floppy Disk Controllers 7

8272A
SINGLE/DOUBLE DENSITY FLOPPY DISK CONTROLLER

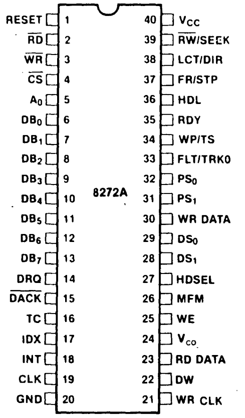
- **IBM Compatible in Both Single and Double Density Recording Formats**
- **Programmable Data Record Lengths: 128, 256, 512, or 1024 Bytes/Sector**
- **Multi-Sector and Multi-Track Transfer Capability**
- **Drives Up to 4 Floppy or Mini-Floppy Disks**
- **Controls 8", 5¼" and 3½" Floppy Disk Drives**
- **Data Transfers in DMA or Non-DMA Mode**
- **Parallel Seek Operations on Up to Four Drives**
- **Compatible with all Intel and Most Other Microprocessors**
- **Single-Phase 8 MHz Clock**
- **Single +5V Power Supply (±10%)**
- **Plastic 40 Pin DIP or 40 Pin CERDIP Packages**

The 8272A is an LSI Floppy Disk Controller (FDC) Chip, which contains the circuitry and control functions for interfacing a processor to 4 Floppy Disk Drives. It is capable of supporting either IBM 3740 single density format (FM), or IBM System 34 Double Density format (MFM) including double sided recording. The 8272A provides control signals which simplify the design of an external phase locked loop and write precompensation circuitry. The FDC simplifies and handles most of the burdens associated with implementing a Floppy Disk Drive Interface. The 8272A is a pin-compatible upgrade to the 8272.



210606–1

Figure 1. 8272A Internal Block Diagram



210606–2

Figure 2. Pin Configuration

November 1986
Order Number: 210606-002

Table 1. Pin Description

Symbol	Pin No.	Type	Connection To	Name and Function
RESET	1	I	µP	**RESET:** Places FDC in idle state. Resets output lines to FDD to "0" (low). Does not clear the last specify command.
$\overline{\text{RD}}$	2	I[1]	µP	**READ:** Control signal for transfer of data from FDC to Data Bus, when "0" (low).
$\overline{\text{WR}}$	3	I[1]	µP	**WRITE:** Control signal for transfer of data to FDC via Data Bus, when "0" (low).
$\overline{\text{CS}}$	4	I	µP	**CHIP SELECT:** IC selected when "0" (low) allowing $\overline{\text{RD}}$ and $\overline{\text{WR}}$ to be enabled.
A_0	5	I[1]	µP	**DATA/STATUS REGISTER SELECT:** Selects Data Reg ($A_0 = 1$) or Status Reg ($A_0 = 0$) contents to be sent to Data Bus.
DB_0–DB_7	6–13	I/O[1]	µP	**DATA BUS:** Bidirectional 8-Bit Data Bus.
DRQ	14	O	DMA	**DATA DMA REQUEST:** DMA Request is being made by FDC when DRQ "1".[3]
$\overline{\text{DACK}}$	15	I	DMA	**DMA ACKNOWLEDGE:** DMA cycle is active when "0" (low) and Controller is performing DMA transfer.
TC	16	I	DMA	**TERMINAL COUNT:** Indicates the termination of a DMA transfer when "1" (high)[2].
IDX	17	I	FDD	**INDEX:** Indicates the beginning of a disk track.
INT	18	O	µP	**INTERRUPT:** Interrupt Request Generated by FDC.
CLK	19	I		**CLOCK:** Single Phase 8 MHz (4 MHz for mini floppies) Squarewave Clock.
GND	20			**GROUND:** D.C. Power Return.
V_{CC}	40			**D.C. POWER:** +5V
$\overline{\text{RW}}$/SEEK	39	O	FDD	**READ WRITE/SEEK:** When "1" (high) Seek mode selected and when "0" (low) Read/Write mode selected.
LCT/DIR	38	O	FDD	**LOW CURRENT/DIRECTION:** Lowers Write current on inner tracks in Read/Write mode, determines direction head will step in Seek mode.
FR/STP	37	O	FDD	**FAULT RESET/STEP:** Resets fault FF in FDD in Read/Write mode, provides step pulses to move head to another cylinder in Seek mode.
HDL	36	O	FDD	**HEAD LOAD:** Command which causes Read/Write head in FDD to contact diskette.
RDY	35	I	FDD	**READY:** Indicates FDD is ready to send or receive data. Must be tied high (gated by the index pulse) for mini floppies which do not normally have a Ready line.
WP/TS	34	I	FDD	**WRITE PROTECT/TWO-SIDE:** Senses Write Protect status in Read/Write mode, and Two Side Media in Seek mode.
FLT/TRK0	33	I	FDD	**FAULT/TRACK 0:** Senses FDD fault condition in Read/Write mode and Track 0 condition in Seek mode.
PS_1, PS_0	31, 32	O	FDD	**PRECOMPENSATION (PRE-SHIFT):** Write precompensation status during MFM mode. Determines early, late, and normal times.
WR DATA	30	O	FDD	**WRITE DATA:** Serial clock and data bits to FDD.
DS_1, DS_0	28, 29	O	FDD	**DRIVE SELECT:** Selects FDD unit.

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Connection To	Name and Function
HDSEL	27	O	FDD	**HEAD SELECT:** Head 1 selected when "1" (high) Head 0 selected when "0" (low).
MFM	26	O	PLL	**MFM MODE:** MFM mode when "1," FM mode when "0".
WE	25	O	FDD	**WRITE ENABLE:** Enables write data into FDD.
VCO	24	O	PLL	**VCO SYNC:** Inhibits VCO in PLL when "0" (low), enables VCO when "1."
RD DATA	23	I	FDD	**READ DATA:** Read data from FDD, containing clock and data bits.
DW	22	I	PLL	**DATA WINDOW:** Generated by PLL, and used to sample data from FDD.
WR CLK	21	I		**WRITE CLOCK:** Write data rate to FDD FM = 500 kHz, MFM = 1 MHz, with a pulse width of 250 ns for both FM and MFM. Must be enabled for all operations, both Read and Write.

NOTES:
1. Disabled when $\overline{CS}$ = 1.
2. TC must be activated to terminate the Execution Phase of any command.
3. DRQ is also an input for certain test modes. It should have a 5 kΩ pull-up resistor to prevent activation.

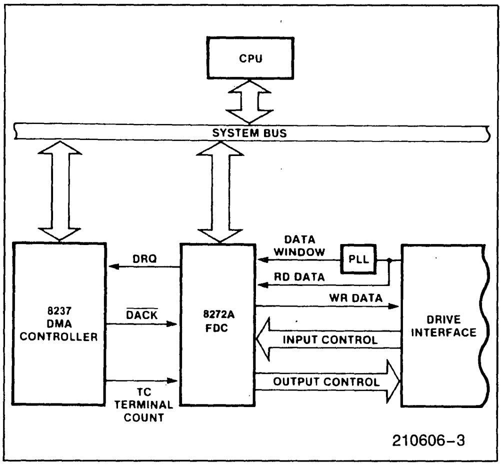


Figure 3. 8272A System Block Diagram

DESCRIPTION

Hand-shaking signals are provided in the 8272A which make DMA operation easy to incorporate with the aid of an external DMA Controller chip, such as the 8237A. The FDC will operate in either DMA or Non-DMA mode. In the Non-DMA mode, the FDC generates interrupts to the processor for every transfer of a data byte between the CPU and the 8272A. In the DMA mode, the processor need only load a command into the FDC and all data transfers occur under control of the 8272A and DMA controller.

There are 15 separate commands which the 8272A will execute. Each of these commands require multiple 8-bit bytes to fully specify the operation which the processor wishes the FDC to perform. The following commands are available.

Read Data
Read ID
Read Deleted Data
Read a Track
Scan Equal
Scan High or Equal
Scan Low or Equal
Specify

Write Data
Format a Track
Write Deleted Data
Seek
Recalibrate (Restore to Track 0)
Sense Interrupt Status
Sense Drive Status

For more information see the Intel Application Notes AP-116 and AP-121.

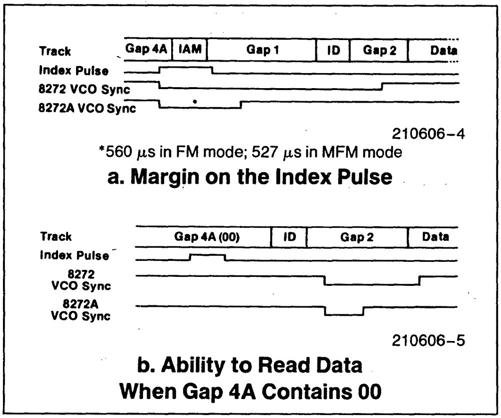
FEATURES

Address mark detection circuitry is internal to the FDC which simplifies the phase locked loop and read electronics. The track stepping rate, head load time, and head unload time may be programmed by the user. The 8272A offers many additional features such as multiple sector transfers in both read and write modes with a single command, and full IBM compatibility in both single (FM) and double density (MFM) modes.

8272A ENHANCEMENTS

On the 8272A, after detecting the Index Pulse, the VCO Sync output stays low for a shorter period of time. See Figure 4a.

On the 8272 there can be a problem reading data when Gap 4A is 00 and there is no IAM. This occurs on some older floppy formats. The 8272A cures this problem by adjusting the VCO Sync timing so that it is not low during the data field. See Figure 4b.



210606-4

*560 μs in FM mode; 527 μs in MFM mode

a. Margin on the Index Pulse

210606-5

b. Ability to Read Data When Gap 4A Contains 00

Figure 4. 8272A Enhancements over the 8272

8272A REGISTERS—CPU INTERFACE

The 8272A contains two registers which may be accessed by the main system processor; a Status Register and a Data Register. The 8-bit Main Status Register contains the status information of the FDC, and may be accessed at any time. The 8-bit Data Register (actually consists of several registers in a stack with only one register presented to the data bus at a time), stores data, commands, parameters, and FDD status information. Data bytes are read out of, or written into, the Data Register in order to program or obtain the results after execution of a command. The Status Register may only be read and is used to facilitate the transfer of data between the processor and 8272A.

The relationship between the Status/Down registers and the signals $\overline{RD}$, $\overline{WR}$, and A_0 is shown in Table 2.

Table 2. A_0, $\overline{RD}$, $\overline{WR}$ Decoding for the Selection of Status/Data Register Functions.

A_0	$\overline{RD}$	$\overline{WR}$	Function
0	0	1	Read Main Status Register
0	1	0	Illegal(1)
0	0	0	Illegal(1)
1	0	0	Illegal(1)
1	0	1	Read from Data Register
1	1	0	Write into Data Register

NOTE:
1. Design must guarantee that the 8272A is not subjected to illegal inputs.

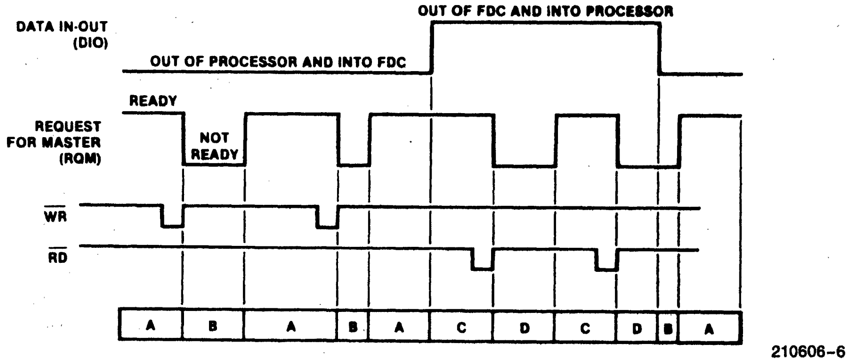
The Main Status Register bits are defined in Table 3.

Table 3. Main Status Register Bit Description

Bit Number	Name	Symbol	Description
D_0	FDD 0 Busy	D_0B	FDD number 0 is in the Seek mode.
D_1	FDD 1 Busy	D_1B	FDD number 1 is in the Seek mode.
D_2	FDD 2 Busy	D_2B	FDD number 2 is in the Seek mode.
D_3	FDD 3 Busy	D_3B	FDD number 3 is in the Seek mode.
D_4	FDC Busy	CB	A read or write command is in process.
D_5	Non-DMA Mode	NDM	The FDC is in the non-DMA mode. This bit is set only during the execution phase in non-DMA mode. Transition to "0" state indicates execution phase has ended.
D_6	Data Input/Output	DIO	Indicates direction of data transfer between FDC and Data Register. If DIO = "1" then transfer is from Data Register to the Processor. If DIO = "0", then transfer is from the Processor to Data Register.
D_7	Request for Master	RQM	Indicates Data Register is ready to send or receive data to or from the Processor. Both bits DIO and RQM should be used to perform the handshaking functions of "ready" and "direction" to the processor.

The DIO and RQM bits in the Status Register indicate when Data is ready and in which direction data will be transferred on the Data Bus.

NOTE:
There is a 12 μs or 24μs RQM flag delay when using an 8 or 4 MHz clock respectively.



NOTES:
A—Data register ready to be written into by processor
B—Data register not ready to be written into by processor
C—Data register ready for next data byte to be read by the processor
D—Data register not ready for next data byte to be read by processor

Figure 5. Status Register Timing

The 8272A is capable of executing 15 different commands. Each command is initiated by a multi-byte transfer from the processor, and the result after execution of the command may also be a multi-byte transfer back to the processor. Because of this multi-byte interchange of information between the 8272A and the processor, it is convenient to consider each command as consisting of three phases:

Command Phase: The FDC receives all information required to perform a particular operation from the processor.

Execution Phase: The FDC performs the operation it was instructed to do.

Result Phase: After completion of the operation, status and other housekeeping information are made available to the processor.

During Command or Result Phases the Main Status Register (described in Table 3) must be read by the processor before each byte of information is written into or read from the Data Register. Bits D6 and D7 in the Main Status Register must be in a 0 and 1 state, respectively, before each byte of the command word may be written into the 8272A. Many of the commands require multiple bytes, and as a result the Main Status Register must be read prior to each byte transfer to the 8272A. On the other hand, during the Result Phase, D6 and D7 in the Main Status Register must both be 1's (D6 = 1 and D7 = 1) before reading each byte from the Data Register.

NOTE:
This reading of the Main Status Register before each byte transfer to the 8272A is required in only the Command and Result Phases, and NOT during the Execution Phase.

During the Execution Phase, the Main Status Register need not be read. If the 8272A is in the non-DMA Mode, then the receipt of each data byte (if 8272A is reading data from FDD) is indicated by an interrupt signal on pin 18 (INT = 1). The generation of a Read signal (RD = 0) will reset the interrupt as well as output the Data onto the Data Bus. For example, if the processor cannot handle Interrupts fast enough (every 13 μs for MFM mode) then it may poll the Main Status Register and then bit D7 (RQM) functions just like the Interrupt signal. If a Write Command is in process, then the WR signal performs the reset to the Interrupt signal.

The 8272A always operates in a multi-sector transfer mode. It continues to transfer data until the TC input is active. In Non-DMA Mode, the system must supply the TC input.

If the 8272A is in the DMA Mode, no Interrupts are generated during the Execution Phase. The 8272A generates DRQ's (DMA Requests) when each byte of data is available. The DMA Controller responds to this request with both a $\overline{\text{DACK}}$ = 0 (DMA Acknowledge) and a $\overline{\text{RD}}$ = 0 (Read signal). When the DMA Acknowledge signal goes low ($\overline{\text{DACK}}$ = 0) then the DMA Request is reset (DRQ = 0). If a Write Command has been programmed then a $\overline{\text{WR}}$ signal will appear instead of $\overline{\text{RD}}$. After the Execution Phase has been completed (Terminal Count has occurred) then an Interrupt will occur (INT = 1). This signifies the beginning of the Result Phase. When the first byte of data is read during the Result Phase, the Interrupt is automatically reset (INT = 0).

It is important to note that during the Result Phase all bytes shown in the Command Table must be read. The Read Data Command, for example has seven bytes of data in the Result Phase. All seven bytes must be read in order to successfully complete the Read Data Command. The 8272A will not accept a new command until all seven bytes have been read. Other commands may require fewer bytes to be read during the Result Phase.

The 8272A contains five Status Registers. The Main Status Register mentioned above may be read by the processor at any time. The other four Status Registers (ST0, ST1, ST2, and ST3) are only available during the Result Phase, and may be read only after successfully completing a command. The particular command which has been executed determines how many of the Status Registers will be read.

The bytes of data which are sent to the 8272A to form the Command Phase, and are read out of the 8272A in the Result Phase, must occur in the order shown in the Table 4. That is, the Command Code must be sent first and the other bytes sent in the prescribed sequence. No foreshortening of the Command or Result Phases are allowed. After the last byte of data in the Command Phase is sent to the 8272A, the Execution Phase automatically starts. In a similar fashion, when the last byte of

Table 4. 8272A Command Set

Phase	R/W	Data Bus								Remarks
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
READ DATA										
Command	W	MT	MFM	SK	0	0	1	1	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W		———		C			———		Sector ID Information
	W		———		H			———		Prior to Command
	W		———		R			———		Execution
	W		———		N			———		
	W		———		EOT			———		
	W		———		GPL			———		
	W		———		DTL			———		
Execution										Data Transfer Between the FDD and Main-System
Result	R		———		ST 0			———		Status Information
	R		———		ST 1			———		After Command
	R		———		ST 2			———		Execution
	R		———		C			———		
	R		———		H			———		Sector ID Information
	R		———		R			———		After Command
	R		———		N			———		Execution
READ DELETED DATA										
Command	W	MT	MFM	SK	0	1	1	0	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W		———		C			———		Sector ID Information
	W		———		H			———		Prior to Command
	W		———		R			———		Execution
	W		———		N			———		
	W		———		EOT			———		
	W		———		GPL			———		
	W		———		DTL			———		
Execution										Data Transfer Between the FDD and Main-System
Result	R		———		ST 0			———		Status Information
	R		———		ST 1			———		After Command
	R		———		ST 2			———		Execution
	R		———		C			———		
	R		———		H			———		Sector ID Information
	R		———		R			———		After Command
	R		———		N			———		Execution

Table 4. 8272A Command Set (Continued)

Phase	R/W	Data Bus								Remarks
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
WRITE DATA										
Command	W	MT	MFM	0	0	0	1	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID Information Prior to Command Execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data Transfer Between the Main-System and FDD
Result	R	ST 0								Status Information After Command Execution
	R	ST 1								
	R	ST 2								
	R	C								
	R	H								Sector ID Information After Command Execution
	R	R								
	R	N								
WRITE DELETED DATA										
Command	W	MT	MFM	0	0	1	0	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID Information Prior to Command Execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data Transfer Between the FDD and Main-System
Result	R	ST 0								Status Information After Command Execution
	R	ST 1								
	R	ST 2								
	R	C								
	R	H								Sector ID Information After Command Execution
	R	R								
	R	N								

Table 4. 8272A Command Set (Continued)

Phase	R/W	Data Bus								Remarks
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
READ A TRACK										
Command	W	0	MFM	SK	0	0	0	1	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID Information
	W	H								Prior to Command
	W	R								Execution
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data Transfer Between the FDD and Main-System. FDC Reads all of Cylinders Contents from Index Hole to EOT
Result	R	ST 0								Status Information
	R	ST 1								After Command
	R	ST 2								Execution
	R	C								
	R	H								Sector ID Information
	R	R								After Command
	R	N								Execution
READ ID										
Command	W	0	MFM	0	0	1	0	1	0	Commands
	W	0	0	0	0	0	HDS	DS1	DS0	
Execution										The First Correct ID Information on the Cylinder is Stored in Data Register
Result	R	ST 0								Status Information
	R	ST 1								After Command
	R	ST 2								Execution
	R	C								
	R	H								Sector ID Information
	R	R								During Execution
	R	N								Phase

Table 4. 8272A Command Set (Continued)

Phase	R/W	Data Bus								Remarks
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
FORMAT A TRACK										
Command	W	0	MFM	0	0	1	1	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W		———		N			———		Bytes/Sector
	W		———		SC			———		Sectors/Cylinder
	W		———		GPL			———		Gap 3
	W		———		D			———		Filler Byte
Execution										FDC Formats an Entire Cylinder
Result	R		———		ST 0			———		Status Information
	R		———		ST 1			———		After Command
	R		———		ST 2			———		Execution
	R		———		C			———		
	R		———		H			———		In This Case, the ID
	R		———		R			———		Information has no
	R		———		N			———		Meaning

Table 4. 8272A Command Set (Continued)

Phase	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
		Data Bus								
SCAN EQUAL										
Command	W	MT	MFM	SK	1	0	0	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W		———		C		———			Sector ID Information Prior to Command Execution
	W		———		H		———			
	W		———		R		———			
	W		———		N		———			
	W		———		EOT		———			
	W		———		GPL		———			
	W		———		STP		———			
Execution										Data Compared Between the FDD and Main-System
Result	R		———		ST 0		———			Status Information After Command Execution
	R		———		ST 1		———			
	R		———		ST 2		———			
	R		———		C		———			
	R		———		H		———			Sector ID Information After Command Execution
	R		———		R		———			
	R		———		N		———			
SCAN LOW OR EQUAL										
Command	W	MT	MFM	SK	1	1	0	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W		———		C		———			Sector ID Information Prior to Command Execution
	W		———		H		———			
	W		———		R		———			
	W		———		N		———			
	W		———		EOT		———			
	W		———		GPL		———			
	W		———		STP		———			
Execution										Data Compared Between the FDD and Main-System
Result	R		———		ST 0		———			Status Information After Command Execution
	R		———		ST 1		———			
	R		———		ST 2		———			
	R		———		C		———			
	R		———		H		———			Sector ID Information After Command Execution
	R		———		R		———			
	R		———		N		———			

Table 4. 8272A Command Set (Continued)

Phase	R/W	Data Bus								Remarks
		D7	D6	D5	D4	D3	D2	D1	D0	
SCAN HIGH OR EQUAL										
Command	W	MT	MFM	SK	1	1	1	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID Information
	W	H								Prior to Command
	W	R								Execution
	W	N								
	W	EOT								
	W	GPL								
	W	STP								
Execution										Data Compared
										Between the FDD
										and Main-System
Result	R	ST 0								Status Information
	R	ST 1								After Command
	R	ST 2								Execution
	R	C								
	R	H								Sector ID Information
	R	R								After Command
	R	N								Execution
RECALIBRATE										
Command	W	0	0	0	0	0	1	1	1	Command Codes
	W	0	0	0	0	0	0	DS1	DS0	
Execution										Head Retracted to
										Track 0
SENSE INTERRUPT STATUS										
Command	W	0	0	0	0	1	0	0	0	Command Codes
Result	R	ST 0								Status Information at
	R	PCN								the End of Each Seek
										Operation About the
										FDC

Table 4. **8272A Command Set** (Continued)

Phase	R/W	Data Bus								Remarks
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
SPECIFY										
Command	W	0	0	0	0	0	0	1	1	Command Codes
	W	—	SRT	———	——>	<——	———	HUT	—	
	W	—	HLT	———	———	———	———	——>	ND	
SENSE DRIVE STATUS										
Command	W	0	0	0	0	0	1	0	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
Result	R		———	———	ST 3		———	———		Status Information about FDD
SEEK										
Command	W	0	0	0	0	1	1	1	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W		———	———	NCN		———	———		
Execution										Head is Positioned Over Proper Cylinder on Diskette
INVALID										
Command	W		———	———	Invalid Codes		———	———		Invalid Command Codes (NoOp—FDC Goes Into Standby State)
Result	R		———	———	ST 0		———	———		ST 0 = 80 (16)

Table 5. Command Mnemonics

Symbol	Name	Description
A_0	Address Line 0	A_0 controls selection of Main Status Register ($A_0 = 0$) or Data Register ($A_0 = 1$).
C	Cylinder Number	C stands for the current selected Cylinder track number 0 through 76 of the medium.
D	Data	D stands for the data pattern which is going to be written into a Sector.
D_7–D_0	Data Bus	8-bit Data bus where D_7 is the most significant bit, and D_0 is the least significant bit.
DS0, DS1	Drive Select	DS stands for a selected drive number 0 or 1.
DTL	Data Length	When N is defined as 00, DTL stands for the data length which users are going to read out or write into the Sector.
EOT	End of Track	EOT stands for the final Sector number of a Cylinder.
GPL	Gap Length	GPL stands for the length of Gap 3 (spacing between Sectors excluding VCO Sync Field).
H	Head Address	H stands for head number 0 or 1, as specified in ID field.
HDS	Head Select	HDS stands for a selected head number 0 or 1 (H = HDS in all command words).
HLT	Head Load Time	HLT stands for the head load time in the FDD (2 to 254 ms in 2 ms increments).
HUT	Head Unload Time	HUT stands for the head unload time after a read or write operation has occurred (16 to 240 ms in 16 ms increments).
MFM	FM or MFM Mode	If MF is low, FM mode is selected and if it is high, MFM mode is selected.
MT	Multi-Track	If MT is high, a multi-track operation is to be peformed (a cylinder under both HD0 and HD1 will be read or written).
N	Number	N stands for the number of data bytes written in a Sector.
NCN	New Cylinder Number	NCN stands for a new Cylinder number, which is going to be reached as a result of the Seek operation. Desired position of Head.
ND	Non-DMA Mode	ND stands for operation in the Non-DMA Mode.
PCN	Present Cylinder Number	PCN stands for the Cylinder number at the completion of SENSE INTERRUPT STATUS Command. Position of Head at present time.
R	Record	R stands for the Sector number, which will be read or written.
R/W	Read/Write	R/W stands for either Read (R) or Write (W) signal.
SC	Sector	SC indicates the number of Sectors per Cylinder.
SK	Skip	SK stands for Skip Deleted Data Address Mark.
SRT	Step Rate Time	SRT stands for the Stepping Rate for the FDD (1 to 16 ms in 1 ms increments). The same Stepping Rate applies to all drives (F = 1 ms, E = 2 ms, etc.).

Table 5. Command Mnemonics (Continued)

Symbol	Name	Description
ST 0 ST 1 ST 2 ST 3	Status 0 Status 1 Status 2 Status 3	ST 0–3 stand for one of four registers which store the status information after a command has been executed. This information is available during the result phase after command execution. These registers should not be confused with the main status register (selected by $A_0 = 0$). ST 0–3 may be read only after a command has been executed and contain information relevant to that particular command.
STP		During a Scan operation, if STP = 1, the data in contiguous sectors is compared byte by byte with data sent from the processor (or DMA), and if STP = 2, then alternate sectors are read and compared.

data is read out in the Result Phase, the command is automatically ended and the 8272A is ready for a new command. A command may be aborted by simply sending a Terminal Count signal to pin 16 (TC=1). This is a convenient means of ensuring that the processor may always get the 8272A's attention even if the disk system hangs up in an abnormal manner.

POLLING FEATURE OF THE 8272A

After power-up RESET, the Drive Select Lines DS0 and DS1 will automatically go into a polling mode. In between commands (and between step pulses in the SEEK command) the 8272A polls all four FDDs looking for a change in the Ready line from any of the drives. If the Ready line changes state (usually due to a door opening or closing) then the 8272A will generate an interrupt. When Status Register 0 (ST0) is read (after Sense Interrupt Status is issued), Not Ready (NR) will be indicated. The polling of the Ready line by the 8272A occurs continuously between instructions, thus notifying the processor which drives are on or off line. Approximate scan timing is shown in Table 6.

Table 6. Scan Timing

DS1	DS0	Approximate Scan Timing
0	0	220 μs
0	1	220 μs
1	0	220 μs
1	1	440 μs

COMMAND DESCRIPTIONS

During the Command Phase, the Main Status Register must be polled by the CPU before each byte is written into the Data Register. The DIO (DB6) and RQM (BD7) bits in the Main Status Register must be in the "0" and "1" states respectively, before each byte of the command may be written into the 8272A. The beginning of the execution phase for any of these commands will cause DIO and RQM to switch to "1" and "0" states respectively.

READ DATA

A set of nine (9) byte words are required to place the FDC into the Read Data Mode. After the Read Data command has been issued the FDC loads the head (if it is in the unloaded state), waits the specified head settling time (defined in the Specify Command), and begins reading ID Address Marks and ID fields. When the current sector number ("R") stored in the ID Register (IDR) compares with the sector number read off the diskette, then the FDC outputs data (from the data field) byte-by-byte to the main system via the data bus.

After completion of the read operation from the current sector, the Sector Number is incremented by one, and the data from the next sector is read and output on the data bus. This continuous function is called a "Multi-Sector Read Operation." The Read Data Command must be terminated by the receipt of a Terminal Count signal. Upon receipt of this signal, the FDC stops outputting data to the processor, but will continue to read data from the current sector, check CRC (Cyclic Redundancy Count) bytes, and then at the end of the sector terminate the Read Data Command.

The amount of data which can be handled with a single command to the FDC depends upon MT (multi-track), MFM (MFM/FM), and N (Number of Bytes/Sector). Table 7 on the next page shows the Transfer Capacity.

Table 7. Transfer Capacity

Multi-Track MT	MFM/FM MFM	Bytes/Sector N	Maximum Transfer Capacity (Bytes/Sector)(Number of Sectors)	Final Sector Read from Diskette
0 0	0 1	00 01	(128) (26) = 3,328 (256) (26) = 6,656	26 at Side 0 or 26 at Side 1
1 1	0 1	00 01	(128) (52) = 6,656 (256) (52) = 13,312	26 at Side 1
0 0	0 1	01 02	(256) (15) = 3,840 (512) (15) = 7,680	15 at Side 0 or 15 at Side 1
1 1	0 1	01 02	(256) (30) = 7,680 (512) (30) = 15,360	15 at Side 1
0 0	0 1	02 03	(512) (8) = 4,096 (1024) (8) = 8,192	8 at Side 0 or 8 at Side 1
1 1	0 1	02 03	(512) (16) = 8,192 (1024) (16) = 16,384	8 at Side 1

The "multi-track" function (MT) allows the FDC to read data from both sides of the diskette. For a particular cylinder, data will be transferred starting at Sector 1, Side 0 and completing at Sector L, Side 1 (Sector L = last sector on the side).

NOTE:

This function pertains to only one cylinder (the same track) on each side of the diskette.

When N = 0, then DTL defines the data length which the FDC must treat as a sector. If DTL is smaller than the actual data length in a Sector, the data beyond DTL in the Sector is not sent to the Data Bus. The FDC reads (internally) the complete Sector performing the CRC check, and depending upon the manner of command termination, may perform a Multi-Sector Read Operation. When N is nonzero, then DTL has no meaning and should be set to 0FFH.

At the completion of the Read Data Command, the head is not unloaded until after Head Unload Time Interval (specified in the Specify Command) has elapsed. If the processor issues another command before the head unloads then the head settling time may be saved between subsequent reads. This time out is particularly valuable when a diskette is copied from one drive to another.

If the FDC detects the Index Hole twice without finding the right sector, (indicated in "R"), then the FDC sets the ND (No Data) flag in Status Register 1 to a 1 (high), and terminates the Read Data Command. (Status Register 0 also has bits 7 and 6 set to 0 and 1 respectively.)

After reading the ID and Data Fields in each sector, the FDC checks the CRC bytes. If a read error is detected (incorrect CRC in ID field), the FDC sets the DE (Data Error) flag in Status Register 1 to a 1 (high), and if a CRC error occurs in the Data Field the FDC also sets the DD (Data Error in Data Field) flag in Status Register 2 to a 1 (high), and terminates the Read Data Command. (Status Register 0 also has bits 7 and 6 set to 0 and 1 respectively.)

If the FDC reads a Deleted Data Address Mark off the diskette, and the SK bit (bit D5 in the first Command Word) is not set (SK = 0), then the FDC sets the CM (Control Mark) flag in Status Register 2 to a 1 (high), and terminates the Read Data Command, after reading all the data in the Sector. If SK = 1, the FDC skips the sector with the Deleted Data Address Mark and reads the next sector.

During disk data transfers between the FDC and the processor, via the data bus, the FDC must be serviced by the processor every 27 μs in the FM Mode, and every 13 μs in the MFM Mode, or the FDC sets the OR (Over Run) flag in Status Register 1 to a 1 (high), and terminates the Read Data Command.

If the processor terminates a read (or write) operation in the FDC, then the ID Information in the Result Phase is dependent upon the state of the MT bit and EOT byte. Table 5 shows the values for C, H, R, and N, when the processor terminates the Command.

Table 8. ID Information When Processor Terminates Command

MT	EOT	Final Sector Transferred to Processor	ID Information at Result Phase			
			C	H	R	N
0	1A 0F 08	Sector 1 to 25 at Side 0 Sector 1 to 14 at Side 0 Sector 1 to 7 at Side 0	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 0 Sector 15 at Side 0 Sector 8 at Side 0	C + 1	NC	R = 01	NC
	1A 0F 08	Sector 1 to 25 at Side 1 Sector 1 to 14 at Side 1 Sector 1 to 7 at Side 1	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 1 Sector 15 at Side 1 Sector 8 at Side 1	C + 1	NC	R = 01	NC
1	1A 0F 08	Sector 1 to 25 at Side 0 Sector 1 to 14 at Side 0 Sector 1 to 7 at Side 0	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 0 Sector 15 at Side 0 Sector 8 at Side 0	NC	LSB	R = 01	NC
	1A 0F 08	Sector 1 to 25 at Side 1 Sector 1 to 14 at Side 1 Sector 1 to 7 at Side 1	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 1 Sector 15 at Side 1 Sector 8 at Side 1	C + 1	LSB	R = 01	NC

NOTES:
1. NC (No Change): The same value as the one at the beginning of command execution.
2. LSB (Least Significant Bit): The least significant bit of H is complemented.

WRITE DATA

A set of nine (9) bytes are required to set the FDC into the Write Data mode. After the Write Data command has been issued the FDC loads the head (if it is in the unloaded state), waits the specified head settling time (defined in the Specify Command), and begins reading ID Fields. When the current sector number ("R"), stored in the ID Register (IDR) compares with the sector number read off the diskette, then the FDC takes data from the processor byte-by-byte via the data bus, and outputs it to the FDD.

After writing data into the current sector, the Sector Number stored in "R" is incremented by one, and the next data field is written into. The FDC continues this "Multi-Sector Write Operation" until the issuance of a Terminal Count signal. If a Terminal Count signal is sent to the FDC it continues writing into the current sector to complete the data field. If the Terminal Count signal is received while a data field is being written then the remainder of the data field is filled with 00 (zeros).

The FDC reads the ID field of each sector and checks the CRC bytes. If the FDC detects a read error (incorrect CRC) in one of the ID Fields, it sets the DE (Data Error) flag of Status Register 1 to a 1 (high), and terminates the Write Data Command. (Status Register 0 also has bits 7 and 6 set to 0 and 1 respectively.)

The Write Command operates in much the same manner as the Read Command. The following items are the same; refer to the Read Data Command for details:

- Transfer Capacity
- EN (End of Cylinder) Flag
- ND (No Data) Flag
- Head Unload Time Interval
- ID Information when the processor terminates command (see Table 2)
- Definition of DTL when N = 0 and when N ≠ 0

In the Write Data mode, data transfers between the processor and FDC must occur every 31 μs in the FM mode, and every 15 μs in the MFM mode. If the time interval between data transfers is longer than this then the FDC sets the OR (Over Run) flag in Status Register 1 to a 1 (high), and terminates the Write Data Command.

For mini-floppies, multiple track writes are usually not permitted. This is because of the turn-off time of the erase head coils—the head switches tracks before the erase head turns off. Therefore the system should typically wait 1.3 ms before attempting to step or change sides.

WRITE DELETED DATA

This command is the same as the Write Data Command except a Deleted Data Address Mark is written at the beginning of the Data Field instead of the normal Data Address Mark.

READ DELETED DATA

This command is the same as the Read Data Command except that when the FDC detects a Data Address Mark at the beginning of a Data Field (and SK = 0 (low)), it will read all the data in the sector and set the CM flag in Status Register 2 to a 1 (high), and then terminate the command. If SK = 1, then the FDC skips the sector with the Data Address Mark and reads the next sector.

READ A TRACK

This command is similar to READ DATA Command except that the entire data field is read continuously from each of the sectors of a track. Immediately after encountering the INDEX HOLE, the FDC starts reading all data fields on the track as continuous blocks of data. If the FDC finds an error in the ID or DATA CRC check bytes, it continues to read data from the track. The FDC compares the ID information read from each sector with the value stored in the IDR, and sets the ND flag of Status Register 1 to a 1 (high) if there is no comparison. Multi-track or skip operations are not allowed with this command.

This command terminates when EOT number of sectors have been read. If the FDC does not find an ID Address Mark on the diskette after it encounters the INDEX HOLE for the second time, then it sets the MA (missing address mark) flag in Status Register 1 to a 1 (high), and terminates the command. (Status Register 0 has bits 7 and 6 set to 0 and 1 respectively.)

READ ID

The READ ID Command is used to give the present position of the recording head. The FDC stores the values from the first ID Field it is able to read. If no proper ID Address Mark is found on the diskette, before the INDEX HOLE is encountered for the second time then the MA (Missing Address Mark) flag in Status Register 1 is set to a 1 (high), and if no data is found then the ND (No Data) flag is also set in Status Register 1 to a 1 (high) and the command is terminated.

FORMAT A TRACK

The Format Command allows an entire track to be formatted. After the INDEX HOLE is detected, Data is written on the Diskette: Gaps, Address Marks, ID Fields and Data Fields, all per the IBM System 34 (Double Density) or System 3740 (Single Density) Format are recorded. The particular format which will be written is controlled by the values programmed into N (number of bytes/sector), SC (sectors/cylinder), GPL (Gap Length), and D (Data Pattern) which are supplied by the processor during the Command Phase. The Data Field is filled with the Byte of data stored in D. The ID Field for each sector is supplied by the processor; that is, four data requests per sector are made by the FDC for C (Cylinder Number), H(Head Number), R(Sector Number) and N(Number of Bytes/Sector). This allows the diskette to be formatted with nonsequential sector numbers, if desired.

After formatting each sector, the processor must send new values for C, H, R, and N to the 8272A for each sector on the track. The contents of the R Register is incremented by one after each sector is formatted, thus, the R register contains a value of R + 1 when it is read during the Result Phase. This incrementing and formatting continues for the whole track until the FDC encounters the INDEX HOLE for the second time, whereupon it terminates the command.

If a FAULT signal is received from the FDD at the end of a write operation, then the FDC sets the EC flag of Status Register 0 to a 1 (high), and terminates the command after setting bits 7 and 6 of Status Register 0 to 0 and 1 respectively. Also the loss of a READY signal at the beginning of a command execution phase causes command termination.

Table 9 shows the relationship between N, SC, and GPL for various sector sizes:

Table 9. Sector Size Relationships

Format	Bytes/ Sector	8" Floppy				Bytes/ Sector	5¼" Floppy				Bytes/ Sector	3½" Mini Floppy			
		N	SC	GPL(1)	GPL(2)		N	SC	GPL(1)	GPL(2)		N	SC	GPL(1)	GPL(2)
FM Mode	128	00	1A	07	1B	128	00	12	07	09	128	0	0F	07	1B
	256	01	0F	0E	2A	128	00	10	10	19	—	—	—	—	—
	512	02	08	1B	3A	256	01	08	18	30	256	1	09	0F	2A
	1024	03	04	47	8A	512	02	04	46	87	512	2	05	1B	3A
	2048	04	02	C8	FF	1024	03	02	C8	FF	—	—	—	—	—
	4096	05	01	C8	FF	2048	04	01	C8	FF	—	—	—	—	—
MPM Mode	256	01	1A	0E	36	256	01	12	0A	0C	256	1	0F	CE	36
	512	02	OF	1B	54	256	01	10	20	32	—	—	—	—	—
	1024	03	08	35	74	512	02	08	2A	50	512	2	09	1B	54
	2048	04	04	99	FF	1024	03	04	80	F0	1024	3	05	35	74
	4096	05	02	C8	FF	2048	04	02	C8	FF	—	—	—	—	—
	8192	06	01	C8	FF	4096	05	01	C8	FF	—	—	—	—	—

NOTES:

1. Suggested values of GPL in Read or Write Commands to avoid splice point between data field and ID field of contiguous sections.
2. Suggested values of GPL in format command.

SCAN COMMANDS

The SCAN Commands allow data which is being read from the diskette to be compared against data which is being supplied from the main system (Processor in NON-DMA mode, and DMA Controller in DMA mode). The FDC compares the data on a byte-by-byte basis, and looks for a sector of data which meets the conditions of $D_{FDD} = D_{Processor}$, $D_{FDD} \leq D_{Processor}$, or $D_{FDD} \geq D_{Processor}$. Ones complement arithmetic is used for comparison (FF = largest number, 00 = smallest number). After a whole sector of data is compared, if the conditions are not met, the sector number is incremented (R + STP → R), and the scan operation is continued. The scan operation continues until one of the following conditions occur; the conditions for scan are met (equal, low, or high), the last sector on the track is reached (EOT), or the terminal count signal is received.

If the conditions for scan are met then the FDC sets the SH (Scan Hit) flag of Status Register 2 to a 1 (high), and terminates the Scan Command. If the

Table 10. Scan Status Codes

Command	Status Register 2		Comments
	Bit 2 = SN	Bit 3 = SH	
Scan Equal	0	1	$D_{FDD} = D_{Processor}$
	1	0	$D_{FDD} \neq D_{Processor}$
Scan Low or Equal	0	1	$D_{FDD} = D_{Processor}$
	0	0	$D_{FDD} < D_{Processor}$
	1	0	$D_{FDD} \not> D_{Processor}$
Scan High or Equal	0	1	$D_{FDD} = D_{Processor}$
	0	0	$D_{FDD} > D_{Processor}$
	1	0	$D_{FDD} \not< D_{Processor}$

conditions for scan are not met between the starting sector (as specified by R) and the last sector on the cylinder (EOT), then the FDC sets the SN (Scan Not Satisfied) flag of Status Register 2 to a 1 (high), and terminates the Scan Command. The receipt of a TERMINAL COUNT signal from the Processor or DMA Controller during the scan operation will cause the FDC to complete the comparison of the particular byte which is in process, and then to terminate the command. Table 10 shows the status of bits SH and SN under various conditions of SCAN.

If the FDC encounters a Deleted Data Address Mark on one of the sectors (and SK = 0), then it regards the sector as the last sector on the cylinder, sets CM (Control Mark) flag of Status Register 2 to a 1 (high) and terminates the command. If SK = 1, the FDC skips the sector with the Deleted Address Mark, and reads the next sector. In the second case (SK = 1), the FDC sets the CM (Control Mark) flag of Status Register 2 to a 1 (high) in order to show that a Deleted Sector had been encountered.

When either the STP (contiguous sectors STP = 01, or alternate sectors STP = 02 sectors are read) or the MT (Multi-Track) are programmed, it is necessary to remember that the last sector on the track must be read. For example, if STP = 02, MT = 0, the sectors are numbered sequentially 1 through 26, and we start the Scan Command at sector 21; the following will happen. Sectors 21, 23, and 25 will be read, then the next sector (26) will be skipped and the Index Hole will be encountered before the EOT value of 26 can be read. This will result in an abnormal termination of the command. If the EOT had been set at 25 or the scanning started at sector 20, then the Scan Command would be completed in a normal manner.

During the Scan Command data is supplied by either the processor or DMA Controller for comparison against the data read from the diskette. In order to avoid having the OR (Over Run) flag set in Status Register 1, it is necessary to have the data available in less than 27 μs (FM Mode) or 13 μs (MFM Mode). If an Overrun occurs the FDC terminates the command.

SEEK

The read/write within the FDD is moved from cylinder to cylinder under control of the Seek Command. The FDC compares the PCN (Present Cylinder Number) which is the current head position with the NCN (New Cylinder Number), and performs the following operation if there is a difference:

PCN < NCN: Direction signal to FDD set to a 1 (high), and Step Pulses are issued. (Step In.)

PCN > NCN: Direction signal to FDD set to a 0 (low), and Step Pulses are issued. (Step Out.)

The rate at which Step Pulses are issued is controlled by SRT (Stepping Rate Time) in the SPECIFY Command. After each Step Pulse is issued NCN is compared against PCN, and when NCN = PCN, then the SE (Seek End) flag is set in Status Register 0 to a 1 (high), and the command is terminated.

During the Command Phase of the Seek operation the FDC is in the FDC BUSY state, but during the Execution Phase it is in the NON BUSY state. While the FDC is in the NON BUSY state, another Seek Command may be issued, and in this manner parallel seek operations may be done on up to 4 Drives at once.

If an FDD is in a NOT READY state at the beginning of the command execution phase or during the seek operation, then the NR (NOT READY) flag is set in Status Register 0 to a 1 (high), and the command is terminated.

Note that the 8272A Read and Write Commands do not have implied Seeks. Any R/W command should be preceded by: 1) Seek Command; 2) Sense Interrupt Status; and 3) Read ID.

RECALIBRATE

This command causes the read/write head within the FDD to retract to the Track 0 position. The FDC clears the contents of the PCN counter, and checks the status of the Track 0 signal from the FDD. As long as the Track 0 signal is low, the Direction signal remains 1 (high) and Step Pulses are issued. When the Track 0 signal goes high, the SE (SEEK END) flag in Status Register 0 is set to a 1 (high) and the command is terminated. If the Track 0 signal is still low after 77 Step Pulses have been issued, the FDC sets the SE (SEEK END) and EC (EQUIPMENT CHECK) flags of Status Register 0 to both 1s (highs), and terminates the command.

The ability to overlap RECALIBRATE Commands to multiple FDDs, and the loss of the READY signal, as described in the SEEK Command, also applies to the RECALIBRATE Command.

SENSE INTERRUPT STATUS

An Interrupt signal is generated by the FDC for one of the following reasons:

1) Upon entering the Result Phase of:
   a) Read Data Command
   b) Read a Track Command
   c) Read ID Command
   d) Read Deleted Data Command
   e) Write Data Command
   f) Format a Cylinder Command
   g) Write Deleted Data Command
   h) Scan Commands
2) Ready Line of FDD changes state
3) End of Seek or Recalibrate Command
4) During Execution Phase in the NON-DMA Mode

Interrupts caused by reasons 1 and 4 above occur during normal command operations and are easily discernible by the processor. However, interrupts caused by reasons 2 and 3 above may be uniquely identified with the aid of the Sense Interrupt Status Command. This command when issued resets the interrupt signal and via bits 5, 6, and 7 of Status Register 0 identifies the cause of the interrupt.

Neither the Seek or Recalibrate Command have a Result Phase. Therefore, it is mandatory to use the Sense Interrupt Status Command after these commands to effectively terminate them and to provide verification of the head position (PCN).

Table 11. Seek, Interrupt Codes

Seek End Bit 5	Interrupt Code		Cause
	Bit 6	Bit 7	
0	1	1	Ready Line Changed State, Either Polarity
1	0	0	Normal Termination of Seek or Recalibrate Command
1	1	0	Abnormal Termination of Seek or Recalibrate Command

SPECIFY

The Specify Command sets the intitial values for each of the three internal timers. The HUT (Head Unload Time) defines the time from the end of the Execution Phase of one of the Read/Write Commands to the head unload state. This timer is programmable from 16 to 240 ms in increments of 16 ms (01 = 16 ms, 02 = 32 ms . . . . OF = 240 ms). The SRT (Step Rate Time) defines the time interval between adjacent step pulses. This timer is programmable from 1 to 16 ms in increments of 1 ms (F = 1 ms, E = 2 ms, D = 3 ms, etc.). The HLT (Head Load Time) defines the time between when the Head Load signal goes high and when the Read/Write operation starts. This timer is programmable from 2 to 254 ms in increments of 2 ms (01 = 2 ms, 02 = 4 ms, 03 = 6 ms . . . . FE = 254 ms).

The step rate should be programmed 1 ms longer than the minimum time required by the drive.

The time intervals mentioned above are a direct function of the clock (CLK on pin 19). Times indicated above are for an 8 MHz clock, if the clock was reduced to 4 MHz (mini-floppy application) then all time intervals are increased by a factor of 2.

The choice of DMA or NON-DMA operation is made by the ND (NON-DMA) bit. When this bit is high (ND = 1) the NON-DMA mode is selected, and when ND = 0 the DMA mode is selected.

SENSE DRIVE STATUS

This command may be used by the processor whenever it wishes to obtain the status of the FDDs. Status Register 3 contains the Drive Status information.

INVALID

If an invalid command is sent to the FDC (a command not defined above), then the FDC will terminate the command. No interrupt is generated by the 8272A during this condition. Bit 6 and bit 7 (DIO and RQM) in the Main Status Register are both high ("1") indicating to the processor that the 8272A is in the Result Phase and the contents of Status Register 0 (STO) must be read. When the processor reads Status Register 0 it will find an 80H indicating an invalid command was received.

A Sense Interrupt Status Command must be sent after a Seek or Recalibrate interrupt, otherwise the FDC will consider the next command to be an Invalid Command.

In some applications the user may wish to use this command as a No-Op command, to place the FDC in a standby or no operation state.

Table 12. Status Registers

Bit			Description
No.	Name	Symbol	
STATUS REGISTER 0			
D_7	Interrupt Code	IC	$D_7 = 0$ and $D_6 = 0$ Normal Termination of Command, (NT). Command was completed and properly executed.
D_6			$D_7 = 0$ and $D_6 = 1$ Abnormal Termination of Command, (AT). Execution of Command was started, but was not successfully completed.
			$D_7 = 1$ and $D_6 = 0$ Invalid Command issue, (IC). Command which was issued was never started.
			$D_7 = 1$ and $D_6 = 1$ Abnormal Termination because during command execution the ready signal from FDD changed state.
D_5	Seek End	SE	When the FDC completes the SEEK Command, this flag is set to 1 (high).
D_4	Equipment Check	EC	If a fault Signal is received from the FDD, or if the Track 0 Signal fails to occur after 77 Step Pulses (Recalibrate Command) then this flag is set.
D_3	Not Ready	NR	When the FDD is in the not-ready state and a read or write command is issued, this flag is set. If a read or write command is issued to Side 1 of a single sided drive, then this flag is set.
D_2	Head Address	HD	This flag is used to indicate the state of the head at Interrupt.
D_1	Unit Select 1	US 1	These flags are used to indicate a Drive Unit Number at Interrupt.
D_0	Unit Select 0	US 0	
STATUS REGISTER 1			
D_7	End of Cylinder	EN	When the FDC tries to access a Sector beyond the final Sector of a Cylinder, this flag is set.
D_6			Not used. This bit is always 0 (low).
D_5	Data Error	DE	When the FDC detects a CRC error in either the ID field or the data field, this flag is set.
D_4	Over Run	OR	If the FDC is not serviced by the main-systems during data transfers, within a certain time interval, this flag is set.
D_3			Not used. This bit always 0 (low).
D_2	No Data	ND	During execution of READ DATA, WRITE DELETED DATA or SCAN Command, if the FDC cannot find the Sector specified in the IDR Register, this flag is set.
			During executing the READ ID Command, if the FDC cannot read the ID field without an error, then this flag is set.
			During the execution of the READ A Cylinder Command, if the starting sector cannot be found, then this flag is set.

Table 12. Status Register (Continued)

Bit			Description
No.	Name	Symbol	
STATUS REGISTER 1 (Continued)			
D_1	Not Writable	NW	During execution of WRITE DATA, WRITE DELETED DATA or Format A Cylinder Command, if the FDC detects a write protect signal from the FDD, then this flag is set.
D_0	Missing Address Mark	MA	If the FDC cannot detect the ID Address Mark after encountering the index hole twice, then this flag is set.
			If the FDC cannot detect the Data Address Mark or Deleted Data Address Mark, this flag is set. Also at the same time, the MD (Missing Address Mark in Data Field) of Status Register 2 is set.
STATUS REGISTER 2			
D_7			Not used. This bit is always 0 (low).
D_6	Control Mark	CM	During executing the READ DATA or SCAN Command, if the FDC encounters a Sector which contains a Deleted Data Address Mark, this flag is set.
D_5	Data Error in Data Field	DD	If the FDC detects a CRC error in the data field then this flag is set.
D_4	Wrong Cylinder	WC	This bit is related with the ND bit, and when the contents of C on the medium is different from that stored in the IDR, this flag is set.
D_3	Scan Equal Hit	SH	During execution, the SCAN Command, if the condition of "equal" is satisfied, this flag is set.
D_2	Scan Not Satisfied	SN	During executing the SCAN Command, if the FDC cannot find a Sector on the cylinder which meets the condition, then this flag is set.
D_1	Bad Cylinder	BC	This bit is related with the ND bit, and when the content of C on the medium is different from that stored in the IDR and the content of C is FF, then this flag is set.
D_0	Missing Address Mark in Data Field	MD	When data is read from the medium, if the FDC cannot find a Data Address Mark or Deleted Data Address Mark, then this flag is set.
STATUS REGISTER 3			
D_7	Fault	FT	This bit is used to indicate the status of the Fault signal from the FDD.
D_6	Write Protected	WP	This bit is used to indicate the status of the Write Protected signal from the FDD.
D_5	Ready	RDY	This bit is used to indicate the status of the Ready signal from the FDD.
D_4	Track 0	T0	This bit is used to indicate the status of the Track 0 signal from the FDD.
D_3	Two Side	TS	This bit is used to indicate the status of the Two Side signal from the FDD.
D_2	Head Address	HD	This bit is used to indicate the status of Side Select signal to the FDD.
D_1	Unit Select 1	US 1	This bit is used to indicate the status of the Unit Select 1 signal to the FDD.
D_0	Unit Select 0	US 0	This bit is used to indicate the status of the Unit Select 0 signal to the FDD.

ABSOLUTE MAXIMUM RATINGS*

Operating Temperature	0°C to +70°C
Storage Temperature	−40°C to +125°C
All Output Voltages	−0.5 to +7V
All Input Voltages	−0.5 to +7V
Supply Voltage V_{CC}	−0.5 to +7V
Power Dissipation	1 Watt

*T_A = 25°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 ±10%

Symbol	Parameter	Limits		Unit	Test Conditions
		Min	Max		
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} = 2.0 mA
V_{OH}	Output High Voltage	2.4	V_{CC}	V	I_{OH} = −400 μA
I_{CC}	V_{CC} Supply Current		120	mA	
I_{IL}	Input Load Current (All Input Pins)		10 −10	μA μA	$V_{IN} = V_{CC}$ V_{IN} = 0V
I_{LOH}	High Level Output Leakage Current		10	μA	$V_{OUT} = V_{CC}$
I_{OFL}	Output Float Leakage Current		±10	μA	0.45C ≤ V_{OUT} ≤ V_{CC}

CAPACITANCE T_A = 25°C, f_c = 1 MHz, V_{CC} = 0V

Symbol	Parameter	Limits		Unit	Test Conditions
		Min	Max		
$C_{IN(\phi)}$	Clock Input Capacitance		20	pF	All Pins Except Pin Under Test Tied to AC Ground
C_{IN}	Input Capacitance		10	pF	
$C_{I/O}$	Input/Output Capacitance		20	pF	

A.C. CHARACTERISTICS T_A = 0°C to +70°C, V_{CC} = +5.0V ± 10%

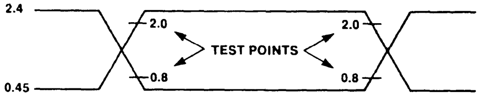
Symbol	Parameter	Typ(1)	Min	Max	Unit	Notes
CLOCK TIMING						
t_{CY}	Clock Period		120	500	ns	(Note 5)
t_{CH}	Clock High Period		40		ns	(Note 4, 5)
t_{RST}	Reset Width		14		t_{CY}	
READ CYCLE						
t_{AR}	Select Setup to $\overline{RD}$ ↓		0		ns	
t_{RA}	Select Hold from $\overline{RD}$ ↑		0		ns	
t_{RR}	$\overline{RD}$ Pulse Width		250		ns	
t_{RD}	Data Delay from $\overline{RD}$ ↓			200	ns	
t_{DF}	Output Float Delay		20	100	ns	
WRITE CYCLE						
t_{AW}	Select Setup to $\overline{WR}$ ↓		0		ns	
t_{WA}	Select Hold from $\overline{WR}$ ↑		0		ns	
t_{WW}	$\overline{WR}$ Pulse Width		250		ns	
t_{DW}	Data Setup to $\overline{WR}$ ↑		150		ns	
t_{WD}	Data Hold from $\overline{WR}$ ↑		5		ns	
INTERRUPTS						
t_{RI}	INT Delay from $\overline{RD}$ ↑			500	ns	(Note 6)
t_{WI}	INT Delay from $\overline{WR}$ ↑			500	ns	(Note 6)
DMA						
t_{RQCY}	DRQ Cycle Period		13		μs	(Note 6)
t_{AKRQ}	$\overline{DACK}$ ↓ to DRQ ↓			200	ns	
t_{RQR}	DRQ ↑ to $\overline{RD}$ ↓		800		ns	(Note 6)
t_{RQW}	DRQ ↑ to $\overline{WR}$ ↓		250		ns	(Note 6)
t_{RQRW}	DRQ ↑ to $\overline{RD}$ ↑ or $\overline{WR}$ ↑			12	μs	(Note 6)

A.C. CHARACTERISTICS T_A = 0°C to +70°C, V_{CC} = +5.0V ±10% (Continued)

Symbol	Parameter	Typ(1)	Min	Max	Unit	Notes
FDD INTERFACE						
t_{WCY}	WCK Cycle Time	2 or 4 1 or 2			μs	MFM = 0 MFM = 1 (Note 2)
t_{WCH}	WCK High Time	250	80	350	ns	
t_{CP}	Pre-Shift Delay from WCK ↑		20	100	ns	
t_{CD}	WDA Delay from WCK ↑		20	100	ns	
t_{WDD}	Write Data Width		t_{WCH} − 50		ns	
t_{WE}	WE ↑ to WCK ↑ or WE ↓ to WCK ↓ Delay		20	100	ns	
t_{WWCY}	Window Cycle Time	2 1			μs	MM = 0 MFM = 1
t_{WRD}	Window Setup to RDD ↑		15		ns	
t_{RDW}	Window Hold from RDD ↓		15		ns	
t_{RDD}	RDD Active Time (HIGH)		40		ns	
FDD SEEK/DIRECTION/STEP						
t_{US}	$US_{0,1}$ Setup to $\overline{RW}$/SEEK ↑		12		μs	(Note 6)
t_{SU}	$US_{0,1}$ Hold after $\overline{RW}$/SEEK ↓		15		μs	(Note 6)
t_{SD}	$\overline{RW}$/SEEK Setup to LCT/DIR		7		μs	(Note 6)
t_{DS}	$\overline{RW}$/SEEK Hold from LCT/DIR		30		μs	(Note 6)
t_{DST}	LCT/DIR Setup to FR/STEP ↑		1		μs	(Note 6)
t_{STD}	LCT/DIR Hold from FR/STEP ↓		24		μs	(Note 6)
t_{STU}	$DS_{2,1}$ Hold from FR/Step ↓		5		μs	(Note 6)
t_{STP}	STEP Active Time (High)	5			μs	(Note 6)
t_{SC}	STEP Cycle Time		33		μs	(Note 3, 6)
t_{FR}	FAULT RESET Active Time (High)		8	10	μs	(Note 6)
t_{IDX}	INDEX Pulse Width	10			t_{CY}	
t_{TC}	Terminal Count Width		1		t_{CY}	

NOTES:
1. Typical values for T_A = 25°C and nominal supply voltage.
2. The former values are used for standard floppy and the latter values are used for mini-floppies.
3. t_{SC} = 33 μs min. is for different drive units. In the case of same unit, t_{SC} can be ranged from 1 ms to 16 ms with 8 MHz clock period, and 2 ms to 32 ms with 4 MHz clock, under software control.
4. From 2.0V to +2.0V.
5. At 4 MHz, the clock duty cycle may range from 16% to 76%. Using an 8 MHz clock the duty cycle can range from 32% to 52%. Duty cycle is defined as: D.C. = 100 (t_{CH} ÷ t_{CY}) with typical rise and fall times of 5 ns.
6. The specified values listed are for an 8 MHz clock period. Multiply timings by 2 when using a 4 MHz clock period.

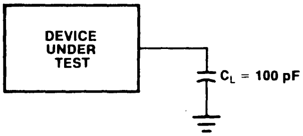
A.C. TESTING INPUT, OUTPUT WAVEFORM



210606-7

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

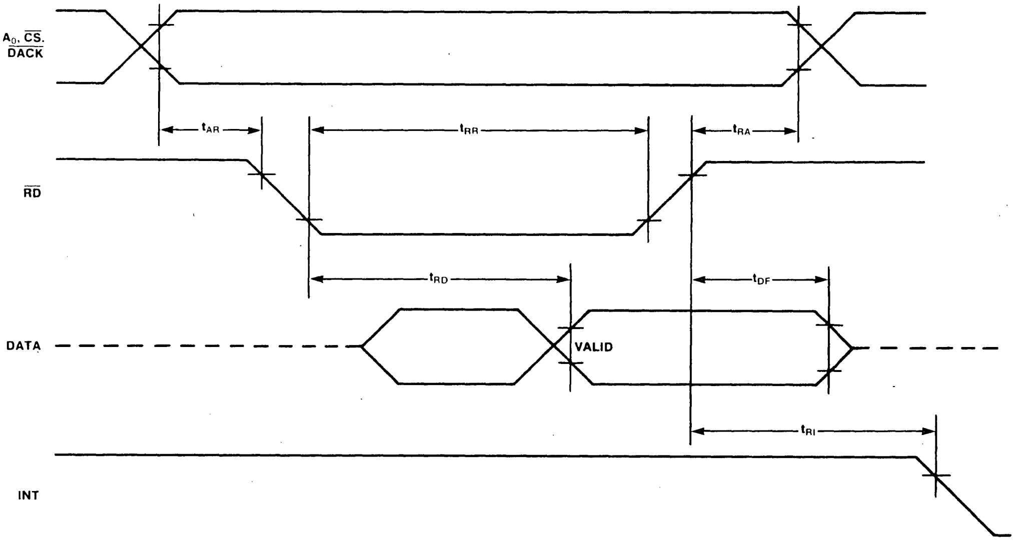


210606-8

C_L = 100 pF
C_L Includes Jig Capacitance

WAVEFORMS

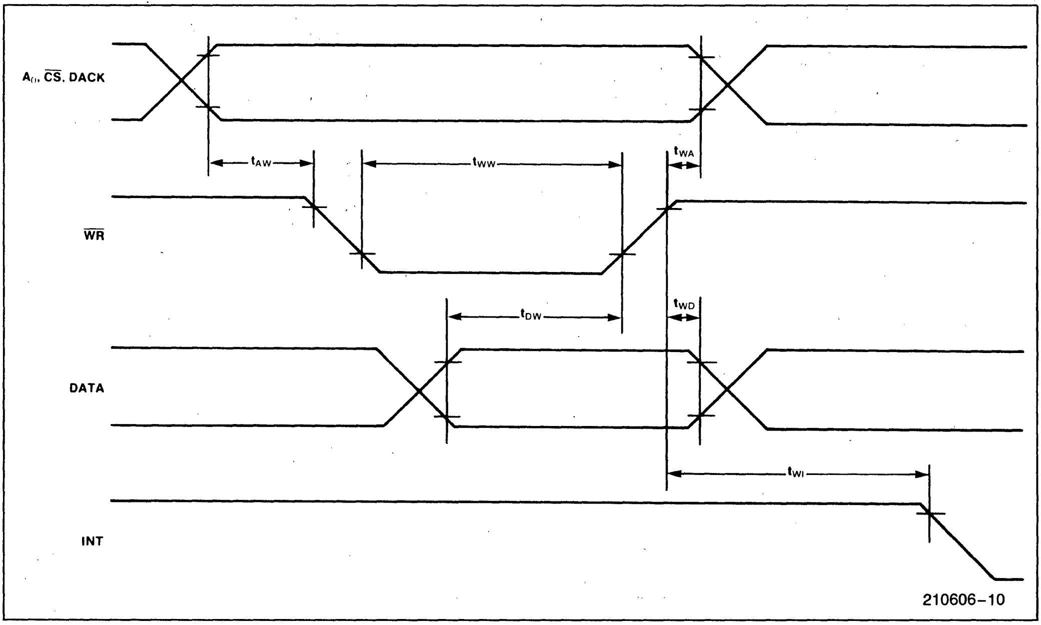
PROCESSOR READ OPERATION



210606-9

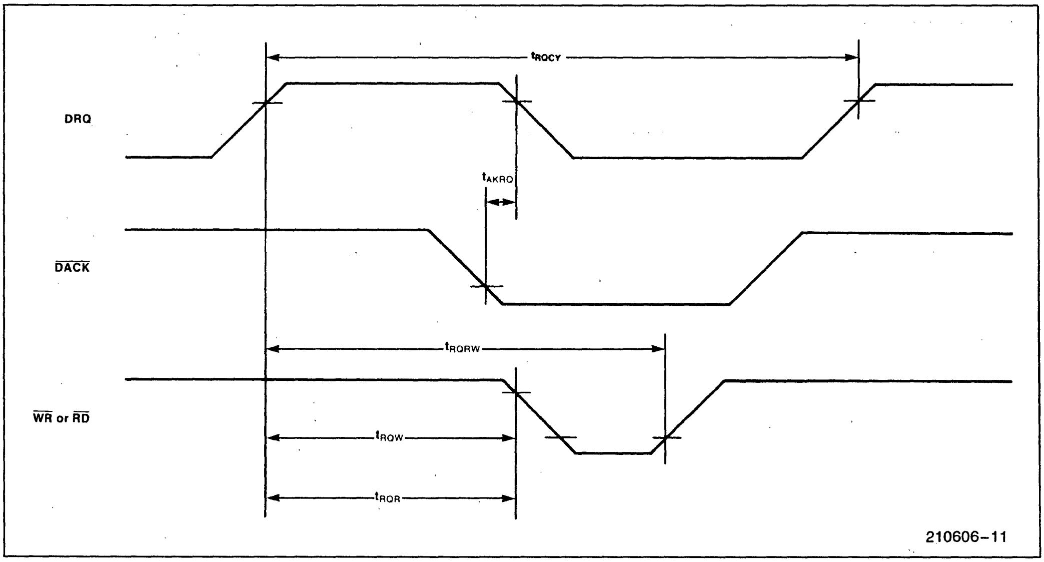
WAVEFORMS (Continued)

PROCESSOR WRITE OPERATION



210606-10

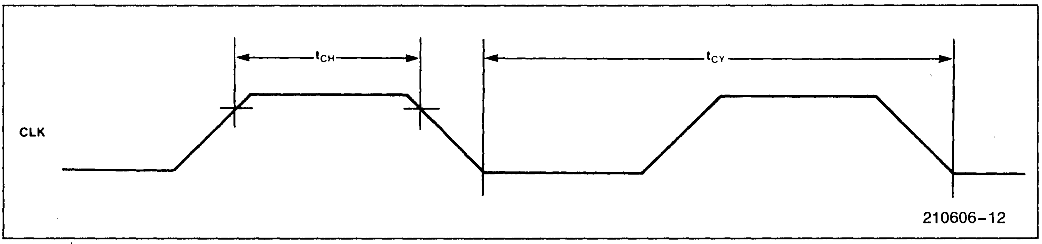
DMA OPERATION



210606-11

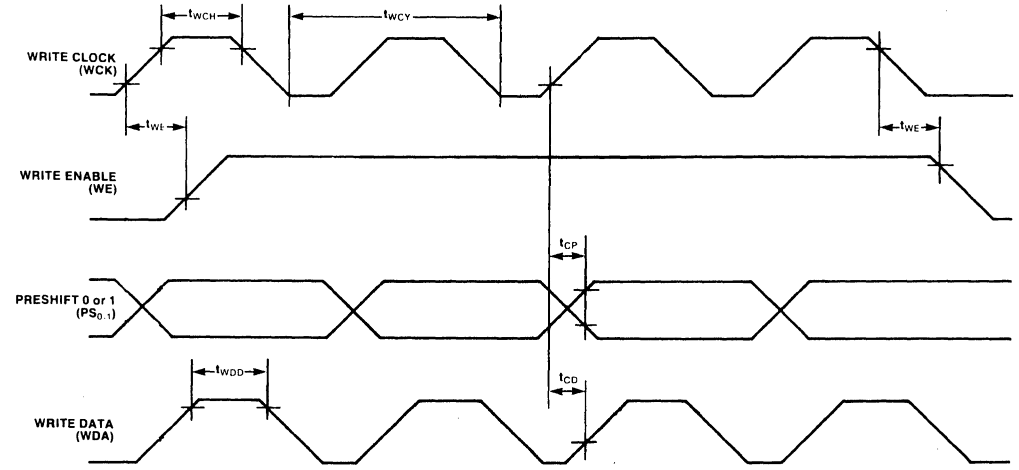
WAVEFORMS (Continued)

CLOCK TIMING



210606-12

FDD WRITE OPERATION

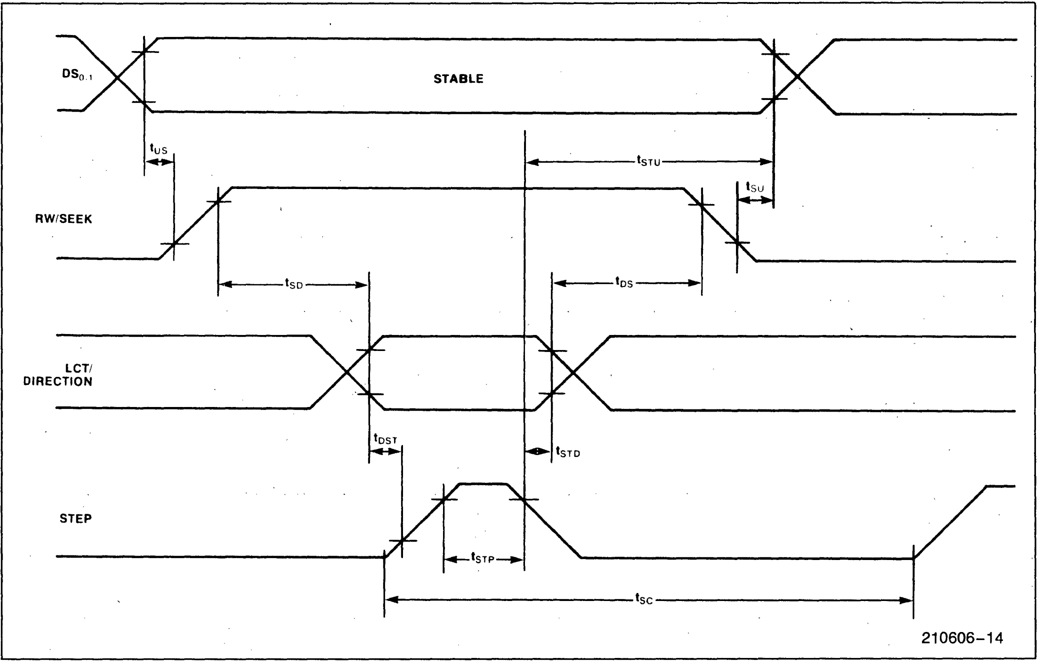


210606-13

	Preshift 0	Preshift 1
Normal	0	0
Late	0	1
Early	1	0
Invalid	1	1

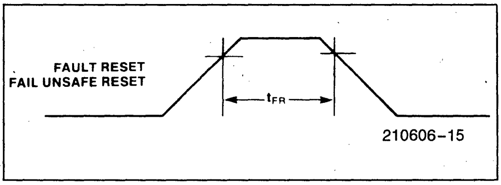
WAVEFORMS (Continued)

SEEK OPERATION



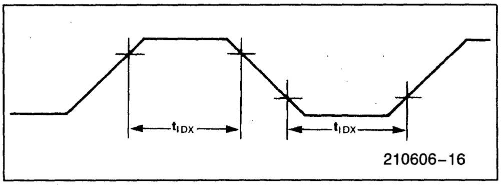
210606–14

FLT RESET



210606–15

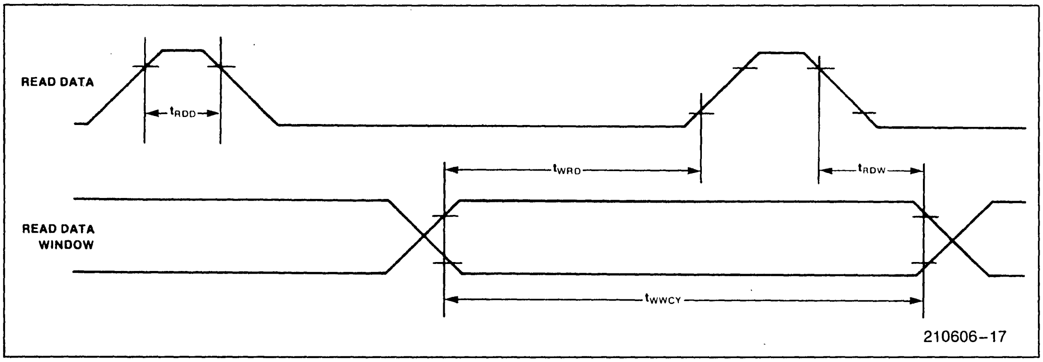
INDEX



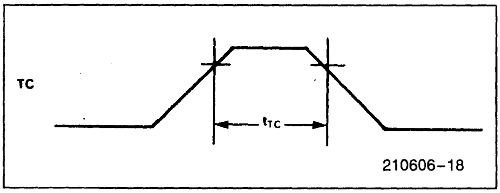
210606–16

WAVEFORMS (Continued)

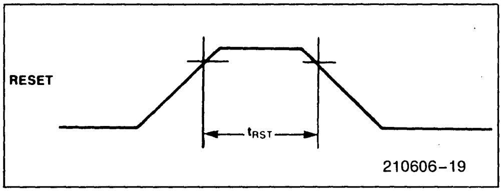
FDD READ OPERATION



TERMINAL COUNT



RESET

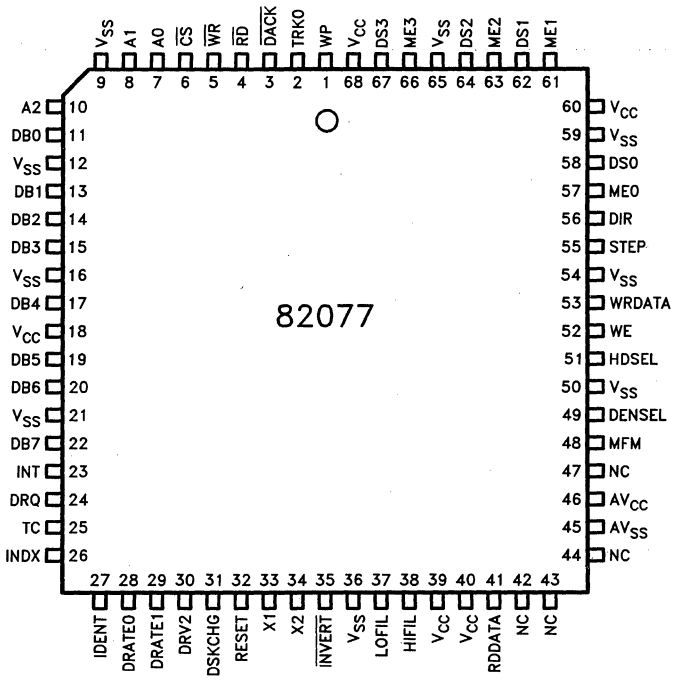


82077
CHMOS SINGLE-CHIP FLOPPY DISK CONTROLLER

- **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

PS/2™ is a trademark of IBM.

October 1988
Order Number: 290166-001

Table 1. 82077 Pin Description

<table>
<tr><th>Symbol</th><th>Pin #</th><th>I/O</th><th>Description</th></tr>
<tr><td colspan="4"><b>HOST INTERFACE</b></td></tr>
<tr><td>RESET</td><td>32</td><td>I</td><td><b>RESET:</b> A high level places the 82077 in a known idle state. All registers are cleared except those set by the Specify command.</td></tr>
<tr><td>$\overline{\text{CS}}$</td><td>6</td><td>I</td><td><b>CHIP SELECT:</b> Decodes base address range and qualifies $\overline{\text{RD}}$ and $\overline{\text{WR}}$ inputs.</td></tr>
<tr><td>A0
A1
A2</td><td>7
8
10</td><td>I</td><td><b>ADDRESS:</b> Selects one of the host interface registers:
<table>
<tr><th>A2</th><th>A1</th><th>A0</th><th></th><th>Register</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>R</td><td>Status Register A</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>R</td><td>Status Register B</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>R/W</td><td>Digital Output Register</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td></td><td>Reserved</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>R</td><td>Main Status Register</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>W</td><td>Data Rate Select Register</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>R/W</td><td>Data (FIFO)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td></td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>R</td><td>Digital Input Register</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>W</td><td>Configuration Control Register</td></tr>
</table></td></tr>
<tr><td>DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7</td><td>11
13
14
15
17
19
20
22</td><td>I/O</td><td><b>DATA BUS:</b> Data bus with 12 mA drive</td></tr>
<tr><td>$\overline{\text{RD}}$</td><td>4</td><td>I</td><td><b>READ:</b> Control signal</td></tr>
<tr><td>$\overline{\text{WR}}$</td><td>5</td><td>I</td><td><b>WRITE:</b> Control signal</td></tr>
<tr><td>DRQ</td><td>24</td><td>O</td><td><b>DMA REQUEST:</b> Requests service from a DMA controller. Normally active high, but goes to high impedance in AT mode when the appropriate bit is set in the DOR.</td></tr>
<tr><td>$\overline{\text{DACK}}$</td><td>3</td><td>I</td><td><b>DMA ACKNOWLEDGE:</b> Control input that qualifies the $\overline{\text{RD}}$, $\overline{\text{WR}}$ inputs in DMA cycles.</td></tr>
<tr><td>TC</td><td>25</td><td>I</td><td><b>TERMINAL COUNT:</b> Control line from a DMA controller that terminates the current disk transfer. TC is accepted only while $\overline{\text{DACK}}$ is active. This input is active high in the AT mode and active low in the PS/2TM mode.</td></tr>
<tr><td>INT</td><td>23</td><td>O</td><td><b>INTERRUPT:</b> Signals a data transfer in non-DMA mode and when status is valid. Normally active high, but goes to high impedance in AT mode when the appropriate bit is set in the DOR.</td></tr>
<tr><td>X1
X2</td><td>33
34</td><td></td><td><b>CRYSTAL 1,2:</b> Connection for a 24 MHz fundamental mode parallel resonant crystal. X1 may be driven with a MOS level clock and X2 would be left unconnected.</td></tr>
</table>

Table 1. 82077 Pin Description (Continued)

Symbol	Pin #	I/O	Description
HOST INTERFACE (Continued)			
IDENT	27	I	**IDENTITY:** Strapping option for either PC-AT or PS/2™ compatibility. Various signals change levels depending upon the mode. **AT MODE:** The AT mode is selected by strapping IDENT to VCC. In this mode, DMA Gate logic is enabled, TC and DENSEL become active high signals (defaults to a 5.25″ floppy disk). **PS/2™ MODE:** The PS/2™ standard mode is selected by strapping IDENT to ground. In this mode, the DMA Gate logic is disabled, TC and DENSEL become active low signals (defaults to a 3.5″ disk).
DISK CONTROL (All outputs have 40 mA drive capability)			
$\overline{\text{INVERT}}$	35	I	**INVERT:** Strapping option. Determines the polartity of **all** signals in this section. Should be strapped to ground when using the internal buffers and these signals become active LOW. When strapped to VCC, these signals become active high and external inverting drivers and receivers are required.
ME0 ME1 ME2 ME3	57 61 63 66	O	**ME0–3:** Decoded Motor enables for drives 0–3. The motor enable pins are directly controlled via the Digital Output Register.
DS0 DS1 DS2 DS3	58 62 64 67	O	**DRIVE SELECT 0–3:** Decoded drive selects for drives 0–3. These outputs are decoded from the select bits in the Digital Output Register and gated by ME0–3.
HDSEL	51	O	**HEAD SELECT:** Selects which side of a disk is to be used. An active level selects side 1.
STEP	55	O	**STEP:** Supplies step pulses to the drive.
DIR	56	O	**DIRECTION:** Controls the direction the head moves when a step signal is present. The head moves toward the center if active.
WRDATA	53	O	**WRITE DATA:** FM or MFM serial data to the drive. Precompensation value is selectable through software.
WE	52	O	**WRITE ENABLE:** Drivè control signal that enables the head to write onto the disk.
DENSEL	49	O	**DENSITY SELECT:** Indicates whether a low (250/300 Kbps) or high (500 Kbps/1 Mbps) data rate has been selected. When PS/2™ mode is selected, a high signal means that low density was enabled. In the AT mode, a high signal indicates high density.
DSKCHG	31	I	**DISK CHANGE:** This input is reflected in the Digital Input Register.
DRV2	30	I	**DRIVE2:** This indicates whether a second drive is installed and is reflected in Status Register A.

Table 1. 82077 Pin Description (Continued)

Symbol	Pin #	I/O	Description
DISK CONTROL (All outputs have 40 mA drive capability) (Continued)			
TRK0	2	I	**TRACK0:** Control line that indicates that the head is on track 0.
WP	1	I	**WRITE PROTECT:** Indicates whether the disk drive is write protected.
INDX	26	I	**INDEX:** Indicates the beginning of the track.
PLL SECTION			
RDDATA	41	I	**READ DATA:** Serial data from the disk. INVERT also affects the polarity of this signal.
HIFIL	38	I/O	**HIGH FILTER:** Analog reference signal for internal data separator compensation. This should be filtered by an external capacitor to LOFIL.
LOFIL	37	I/O	**LOW FILTER:** Low noise ground return for the reference filter capacitor.
MFM	48	O	**MFM:** Indication of the current data encoding/decoding mode. MFM is active high.
DRATE0 DRATE1	28 29	O	**DATARATE0-1:** Reflects the contents of bits 0,1 of the Data Rate Register.
MISCELLANEOUS			
VCC	18 39 40 60 68		**Voltage:** +5V
GND	9 12 16 21 36 50 54 59 65		**Ground**
AVCC	46		**Analog Supply**
AVSS	45		**Analog Ground**
NC	42 43 44 47		**No Connection:** These pins **MUST** be left unconnected; they access unsupported internal PLL modes.

1.0 INTRODUCTION

The 82077 is a true single-chip floppy disk controller for the PC-AT and PS/2™. The 82077, a 24 MHz crystal, a resistor package and a chip select implement a complete design. All drive control signals are fully decoded and have 40 mA drive buffers with selectable polarity. Signals returned from the drive are sent through on-chip input buffers with hysteresis for noise immunity. The integrated analog data separator needs no external compensation yet allows for a wide motor speed variation with exceptionally low soft error rates. The microprocessor interface has a 12 mA drive buffer on the data bus plus 100% hardware register compatibility for PC-AT's and PS/2™'s. The 16-byte FIFO with programmable thresholds is extremely useful in multi-master systems (PS/2™) or systems with a large amount of bus latency.

Upon reset, the 82077 defaults to 8272A functionality. New features are either selected via hardware straps or new commands. Figure 1-1 is a block diagram of the 82077.

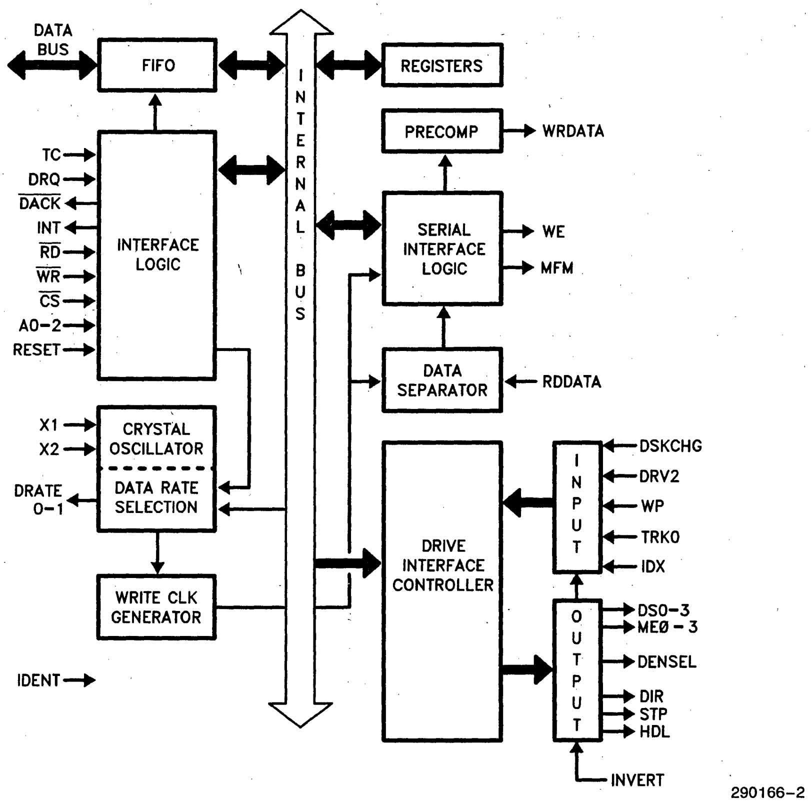


Figure 1-1. 82077 Block Diagram

1.1 Oscillator

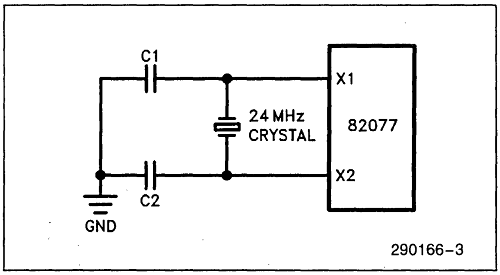


Figure 1-2. Crystal Oscillator Circuit

The 24 MHz clock can be supplied either by a crystal or a MOS level square wave. All internal timings are referenced to this clock or a scaled count which is data rate dependent.

The crystal oscillator must be allowed to run for 10 ms after VCC has reached 4.5V or exiting the POWERDOWN mode to guarantee that it is stable.

Crystal Specifications

Frequency:	24 MHz
Mode:	Parallel Resonant Fundamental Mode
Series Resistance:	Less than 40Ω
Shunt Capacitance:	Less than 5 pF

1.2 Perpendicular Recording Mode

An added capability of the 82077 is the ability to interface directly to perpendicular recording floppy drives that use the Toshiba format. Perpendicular recording differs from the traditional longitudinal method by orienting the magnetic bits vertically. This scheme then packs in more data bits for the same area.

The 82077 with Toshiba perpendicular recording drives can at a minimum, read standard 3.5″ floppies as well as read and write perpendicular media. Some manufacturers offer drives that can read and write standard and perpendicular media in a perpendicular media drive.

A single command puts the 82077 into perpendicular mode. All other commands operate as they normally do. The perpendicular mode requires the 1 Mbps data rate of the 82077. At this data rate, the FIFO eases the host interface bottleneck due to the speed of data transfer to or from the disk.

2.0 MICROPROCESSOR INTERFACE

The interface consists of the standard asynchronous signals: $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{CS}}$, A0–A3, INT, DMA control and a data bus. The address lines select between configuration registers, the FIFO and control/status registers. This interface can be switched between PC-AT or PS/2™ normal modes. The PS/2™ register set is a superset of the registers found in a PC-AT. For PC-AT applications, the PS/2™ register set is located at unused locations. Note that the registers are always present regardless of which mode is selected.

2.1 Status, Data and Control Registers

The base address range is supplied via the $\overline{\text{CS}}$ pin. For PC-AT or PS/2™ designs this would be 3F0 Hex to 3F7 Hex.

A2	A1	A0		Register	
0	0	0	R	Status Register A	SRA
0	0	1	R	Status Register B	SRB
0	1	0	R/W	Digital Output Register	DOR
0	1	1		Reserved	
1	0	0	R	Main Status Register	MSR
1	0	0	W	Data Rate Select Register	DSR
1	0	1	R/W	Data (FIFO)	FIFO
1	1	0		Reserved	
1	1	1	R	Digital Input Register	DIR
1	1	1	W	Configuration Control Register	CCR

2.1.1 STATUS REGISTER A (SRA)

This register is read-only and monitors the state of the interrupt pin and several disk interface pins. This register is part of the PS/2™ register set.

7	6*	5	4*	3	2*	1*	0
INT PENDING	$\overline{\text{DRV2}}$	STEP	$\overline{\text{TRK0}}$	HDSEL	$\overline{\text{INDX}}$	$\overline{\text{WP}}$	DIR

The INT PENDING bit is used by software to monitor the state of the 82077 INTERRUPT pin. The bits marked with a "*" reflect the state of drive signals on the cable and therefore are independent of the state of the $\overline{\text{INVERT}}$ pin.

As a read-only register, there is no default value associated with a reset other than some drive bits will change with a reset. The INT PENDING, STEP, HDSEL, and DIR bits will be low after reset.

2.1.2 STATUS REGISTER B (SRB)

This register is read-only and monitors the state of several disk interface pins. This register is part of the PS/2™ register set.

7	6	5	4	3*	2	1	0
1	1	DRIVE SEL 0	WRDATA TOGGLE	RDDATA TOGGLE	WE	MOT EN1	MOT EN0

As the only drive input, RDDATA TOGGLE's activity is independent of the $\overline{\text{INVERT}}$ pin level and reflects the level as seen on the cable.

The two TOGGLE bits do not read back the state of their respective pins directly. Instead, the pins drive a Flip/Flop which produces a wider and more reliably read pulse. Bits 6 and 7 are undefined and always return a 1.

After any reset, the activity on the TOGGLE pins is cleared. Drive select and Motor bits cleared by the RESET pin and not software resets.

2.1.3 DIGITAL OUTPUT REGISTER (DOR)

The Digital Output Register contains the drive select and motor enable bits, a reset bit and a $\overline{\text{DMA GATE}}$ bit. This register is used in both PC-AT and PS/2™ designs.

7	6	5	4	3	2	1	0
MOT EN3	MOT EN2	MOT EN1	MOT EN0	$\overline{\text{DMA GATE}}$	$\overline{\text{RESET}}$	DRIVE SEL 1	DRIVE SEL 0

The MOT ENx bits directly control their respective motor enable pins (ME0–3). A one means the pin is active, the $\overline{\text{INVERT}}$ pin determines which level that is. The DRIVE SELx bits are decoded to provide four drive select lines and only one may be active at a time. A one is active and the $\overline{\text{INVERT}}$ pin determines the level on the cable. Standard programming practice is to set both MOT ENx and DRIVE SELx bits at the same time.

Table 2-1 lists a set of DOR values to activate the drive select and motor enable for each drive.

Table 2-1. Drive Activation Values

Drive	DOR Value
0	1CH
1	2DH
2	4EH
3	8FH

The $\overline{\text{DMAGATE}}$ bit is enabled only if the IDENT pin is tied to V_{CC} (PC-AT mode). If $\overline{\text{DMAGATE}}$ is set to a low, the INT and DRQ output pins are tristated and if set high, INT and DRQ are enabled to the system. If IDENT is tied to VSS, $\overline{\text{DMAGATE}}$ has no effect upon the INT and DRQ pins and they are always active.

This $\overline{\text{RESET}}$ bit clears the basic core of the 82077 and the FIFO circuits. Once set, it remains set until the user clears this bit. This bit is set by a chip reset and the 82077 is held in a reset state until the user clears this bit. The $\overline{\text{RESET}}$ bit has no effect upon this register. The $\overline{\text{RESET}}$ pin will clear this register.

2.1.4 DATARATE SELECT REGISTER (DSR)

This register is included for compatibility with the 82072 floppy controller and is write-only. Changing the data rate changes the timings of the drive control signals. To ensure that drive timings are not violated when changing data rates, choose a drive timing such that the fastest data rate will not violate the timing.

7	6	5	4	3	2	1	0
S/W RESET	POWER DOWN	0	PRE-COMP 2	PRE-COMP 1	PRE-COMP 0	DRATE SEL 1	DRATE SEL 0

This register is the same as used in the 82072 except that the internal/external PLL select bit is removed. It is recommended that bit 5 be written with a 0 for compatibility.

S/W RESET behaves the same as DOR RESET except that this reset is self clearing.

POWER DOWN deactivates the internal clocks and shuts off the oscillator. Disk control pins are put in an inactive state. All input signals must be held in a valid state (D.C. level 1 or 0). POWER DOWN is exited by activating one of the reset functions.

PRECOMP 0–2 adjusts the WRDATA output to the disk to compensate for magnetic media phenomena known as bit shifting. The data patterns that are susceptible to bit shifting are well understood and the

82077 compensates the data pattern as it is written to the disk. The amount of precompensation is dependent upon the drive and media but in most cases the default value is acceptable.

The 82077 starts precompensating the data pattern starting on Track 0. The CONFIGURE command can change the track that precompensating starts on. Table 2-2 lists the precompensation values that can be selected and Table 2-3 lists the default precompensation values. The default value is selected if the three bits are zeros.

Table 2-2. Precompensation Delays

PRECOMP 432	Precompensation Delay
111	0.00 ns—DISABLED
001	41.67 ns
010	83.34 ns
011	125.00 ns
100	166.67 ns
101	208.33 ns
110	250.00 ns
000	DEFAULT

Table 2-3. Default Precompensation Delays

Data Rate	Precompensation Delays
1 Mbps	41.67 ns
500 Kbps	125 ns
300 Kbps	125 ns
200 Kbps	125 ns

DRATE 0–1 select one of the four data rates as listed in Table 2-4. The default value is 250 Kbps upon a chip reset. Other Resets do not affect the DRATE or PRECOMP bits.

Table 2-4. Data Rates

DRATESEL		DATA RATE	
1	0	MFM	FM
1	1	1 Mbps	Illegal
0	0	500 Kbps	250 Kbps
0	1	300 Kbps	150 Kbps
1	0	250 Kbps	125 Kbps

2.1.5 MAIN STATUS REGISTER (MSR)

The Main Status Register is a read-only register and is used for controlling command input and result output for all commands.

7	6	5	4	3	2	1	0
RQM	DIO	NON DMA	CMD BSY	DRV 3 BUSY	DRV 2 BUSY	DRV 1 BUSY	DRV 0 BUSY

RQM—Indicates that the host can transfer data if set to a 1. No access is permitted if set to a 0.

DIO—Indicates the direction of a data transfer once RQM is set. A 1 indicates a read and a 0 indicates a write is required.

NON-DMA—This mode is selected in the SPECIFY command and will be set to a 1 during the execution phase of a command. This is for polled data transfers and helps differentiate between the data transfer phase and the reading of result bytes.

COMMAND BUSY—This bit is set to a one when a command is in progress. This bit will go active after the command byte has been accepted and goes inactive at the end of the results phase. If there is no result phase (SEEK, RECALIBRATE commands), this bit is returned to a 0 after the last command byte.

DRV x BUSY—These bits are set to ones when a drive is in the seek portion of a command, including implied and overlapped seeks, and recalibrates.

2.1.6 FIFO (DATA)

All command parameter information and disk data transfers go through the FIFO. The FIFO is 16 bytes in size and has programmable threshold values. Data transfers are governed by the RQM and DIO bits in the Main Status Register.

The FIFO defaults to an 8272A compatible mode after any form of reset. This maintains PC-AT hardware compatibility. The default values can be changed through the CONFIGURE command (enable full FIFO operation with threshold control). The advantage of the FIFO is that it allows the system a larger DMA latency without causing a disk error. Table 2.5 gives several examples of the delays with a FIFO. The data is based upon the following formula:

$$\text{Threshold\#} \times \left| \frac{1}{\text{DATA RATE}} \times 8 \right| - 1.5\ \mu\text{s} = \text{DELAY}$$

Table 2-5. FIFO Service Delay

FIFO Threshold Examples	Maximum Delay to Servicing at 1 Mbps Data Rate
1 byte	$1 \times 8\ \mu s - 1.5\ \mu s = 6.5\ \mu s$
2 bytes	$2 \times 8\ \mu s - 1.5\ \mu s = 14.5\ \mu s$
8 bytes	$8 \times 8\ \mu s - 1.5\ \mu s = 62.5\ \mu s$
15 bytes	$15 \times 8\ \mu s - 1.5\ \mu s = 118.5\ \mu s$

FIFO Threshold Examples	Maximum Delay to Servicing at 500 Kbps Data Rate
1 byte	1 × 16 μs − 1.5 μs = 14.5 μs
2 bytes	2 × 16 μs − 1.5 μs = 30.5 μs
8 bytes	8 × 16 μs − 1.5 μs = 126.5 μs
15 bytes	15 × 16 μs − 1.5 μs = 238.5 μs

At the start of a command, the FIFO action is always disabled and command parameters must be sent based upon the RQM and DIO bit settings. As the 82077 enters the command execution phase, it clears the FIFO of any data to ensure that invalid data is not transferred.

An overrun or underrun will terminate the current command and the transfer of data. Disk writes will complete the current sector by generating a 00 pattern and valid CRC. Reads require the host to remove the remaining data so that the result phase may be entered.

2.1.7 DIGITAL INPUT REGISTER (DIR)

This register is used to sense the state of the DSKCHG and DENSEL inputs and is read-only.

7*	6	5	4	3	2	1	0*
DSK CHG	1	1	1	1	1	1	$\overline{\text{HIGH}}$ $\overline{\text{DENS}}$

DSKCHG monitors the pin of the same name and reflects the opposite value seen on the disk cable, regardless of the value of $\overline{\text{INVERT}}$.

$\overline{\text{HIGH DENS}}$ is low whenever the 500 Kbps or 1 Mbps data rates are selected. This bit is independent of the effects of the IDENT and $\overline{\text{INVERT}}$ pins.

Table 2-6 shows the state of the DENSEL pin when $\overline{\text{INVERT}}$ is low.

Table 2-6. DENSEL Encoding

Data Rate	IDENT	DENSEL
1 Mbps	0	0
	1	1
500 Kbps	0	0
	1	1
300 Kbps	0	1
	1	0
250 Kbps	0	1
	1	0

This pin is set high after a pin RESET and is unaffected by DOR and DSR resets.

All other bits are unused and returned as 1's.

2.1.8 CONFIGURATION CONTROL REGISTER (CCR)

This register sets the datarate and is write only. In the PC-AT it is named the DSR.

7	6	5	4	3	2	1	0
—	—	—	—	—	—	DRATE SEL 1	DRATE SEL 0

Refer to the table in the Data Rate Select Register for values. Unused bits should be set to 0.

2.2 RESET

There are three sources of reset on the 82077; the RESET pin, a reset generated via a bit in the DOR and a reset generated via a bit in the DSR. All resets take the 82077 out of the power down state.

On entering the reset state, all operations are terminated and the 82077 enters an idle state. Activating reset while a disk write activity is in progress will corrupt the data and CRC.

On exiting the reset state, various internal registers are cleared, including the CONFIGURE command information, and the 82077 waits for a new command. Drive polling will start unless disabled by a new CONFIGURE command.

2.2.1 RESET PIN

The RESET pin is a global reset and clears all registers except those programmed by the SPECIFY command. The DOR Reset bit is enabled and must be cleared by the host to exit the reset state.

2.2.2 DOR RESET vs DSR RESET

These two resets are functionally the same. The DSR Reset is included to maintain 82072 compatibility. Both will reset the 8272 core which affects drive status information and the FIFO circuits. The DSR Reset clears itself automatically while the DOR Reset requires the host to manually clear it. DOR Reset has precedence over the DSR Reset. The DOR Reset is set automatically upon a pin RESET. The user must manually clear this reset bit in the DOR to exit the reset state.

2.3 DMA Transfers

DMA transfers are enabled with the SPECIFY command and are initiated by the 82077 by activating the DRQ pin during a data transfer command. The FIFO is enabled directly by asserting $\overline{\text{DACK}}$ and addresses need not be valid.

3.0 DRIVE INTERFACE

The 82077 has integrated all of the logic needed to interface to a floppy disk drive. All drive outputs have 40 mA drive capability and all inputs use a receive buffer with hysteresis. The internal analog data separator requires no external components, yet allows for an extremely wide capture range with high levels of read-data jitter. The designer needs only to run the 82077 disk drive signals to the disk drive connector.

3.1 Cable Interface

The $\overline{\text{INVERT}}$ pin selects between using the internal buffers on the 82077 or user supplied inverting buffers. $\overline{\text{INVERT}}$ pulled to V_{CC} disables the internal buffers; pulled to ground will enable them. There is no need to use external buffers with the 82077 in PC-AT or PS/2™ applications.

The selection between the PC-AT or PS/2™ interface, through the $\overline{\text{IDENT}}$ pin, also changes the definition of the DENSEL pin. The AT interface assumes using 5.25″ drives where a high on DENSEL tells the drive that either the 500 Kbps or 1 Mbps datarate is selected. The PS/2™ interface assumes 3.5″ drives and DENSEL changes to a low for the high datarates.

Additionally, the two types of drives have different electrical interfaces. Generally, the 5.25″ drive uses open collector drivers and the 3.5″ drives (as used on PS/2™) use totem-pole drivers. The output buffers on the 82077 do not change between open collector or totem-pole, they are always totem-pole. For design information on interfacing 5.25″ and 3.5″ drives to a single 82077, refer to Section 9.

3.2 Data Separator

The function of the data separator is to lock onto the incoming serial read data. When lock is achieved the serial front end logic of the chip is provided with a clock which is synchronized to the read data. The synchronized clock, called Data Window, is used to internally sample the serial data. One state of Data Window is used to sample the data portion of the bit cell, and the alternate state samples the clock portion. Serial to parallel conversion logic separates the read data into clock and data bytes.

To support reliable disk reads the data separator must track fluctuations in the disk data frequency. Frequency errors primarily arise from two sources: drive rotation speed variation and instantaneous speed variation (ISV). A second condition, and one that opposes the ability to track frequency shifts is the response to bit jitter.

The internal data separator consists of two analog phase lock loops (PLLs) as shown in Figure 3-1. The two PLLs are referred to as the reference PLL and the data PLL. The reference PLL (the master PLL) is used to bias the data PLL (the slave PLL). The reference PLL adjusts the data PLL's operating point as a function of process, junction temperature and supply voltage. Using this architecture it was possible to eliminate the need for external trim components.

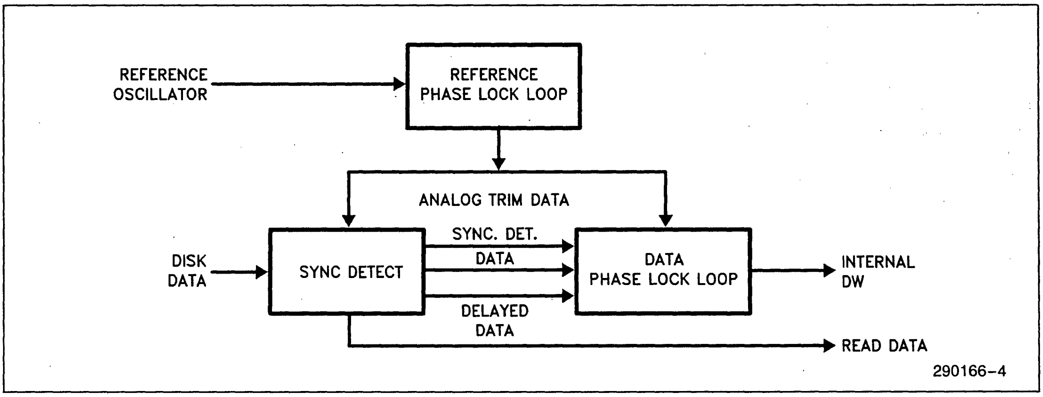


Figure 3-1. Data Separator Block Diagram

3.2.1 PHASE LOCK LOOP OVERVIEW

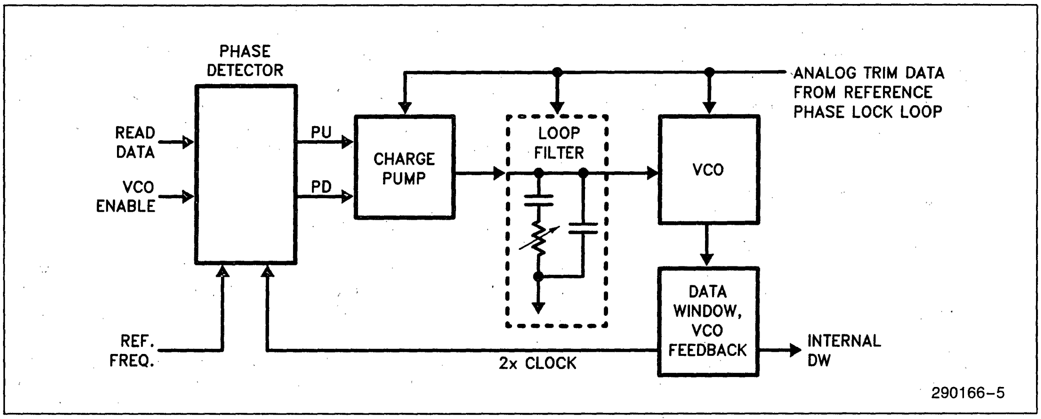


Figure 3-2. Data PLL

Figure 3-2 shows the data PLL. The reference PLL has control over the loop gain by its influence on the charge pump and the VCO. In addition the reference PLL controls the loop filter time constant. As a result the closed loop transfer function of the data PLL is controlled, and immune, to the first order, to environmental factors, and process variation.

Systems of this type are often very sensitive to noise. In the design of this data separator many steps were taken to avoid noise sensitivity problems. The analog section of the chip has a separate VSS pin (AVSS) which should be connected externally to a noise free ground. This provides a clean basis for VSS referenced signals. In addition many analog circuit features were employed to make the overall system as insensitive to noise as possible.

3.2.1 JITTER TOLERANCE

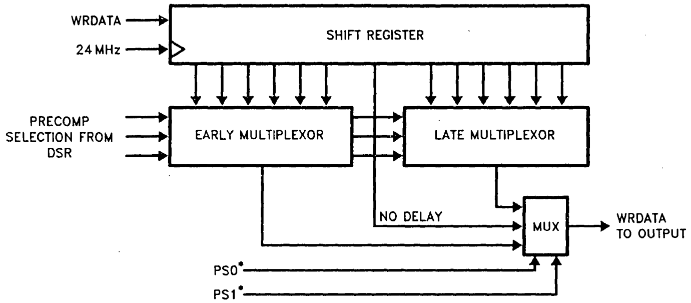
The jitter immunity of the system is dominated by the data PLL's response to phase impulses. This is measured as a percentage of the theoretical data window by dividing the maximum readable bit shift by a ¼ bitcell distance. For instance, if the maximum allowable bit shift is 300 ns for a 500 Kbps data stream, the jitter tolerance is 60%. The graph in Figure 12-1 of the Data Separator Characteristics section illustrates the jitter tolerance of the 82077 across a ±6% frequency range.

3.2.2 LOCKTIME (tLOCK)

The lock, or settling time of the data PLL is designed to be 64 bit times. This corresponds to 4 sync bytes in the FM mode and 8 sync bytes in the MFM mode. This value assumes that the sync field jitter is 5% the bit cell or less. This level of jitter should be easily achieved for a constant bit pattern, since intersymbol interference should be equal, thus nearly eliminating random bit shifting.

3.2.3 CAPTURE RANGE

Capture Range is the maximum frequency range over which the data separator will acquire phase lock with the incoming RDDATA signal. In a floppy disk environment, this frequency variation is composed of two components: drive motor speed error and ISV. Frequency is a factor which may determine the maximum level of the ISV (Instantaneous Speed Variation) component. In general, as frequency increases the allowed magnitude of the ISV component will decrease. When determining the capture range requirements, the designer should take the maximum amount of frequency error for the disk drive and double it to account for media switching between drives.



290166-6

NOTE:
PS0,1 are 8272A control signals but are not available as outputs on the 82077.

Figure 3-3. Precompensation Block Diagram

3.2.4 REFERENCE FILTER

To provide a clean bias voltage for the internal data separator, two pins have been provided to filter this signal. It is recommended to place a 0.005 uF capacitor between HIFIL and LOFIL to filter the reference signal. A smaller capacitance will reduce the effectiveness of the filter and could result in a lower jitter tolerance. Conversely, a larger capacitance has the potential to further improve jitter tolerance, but will result in an increased settling time after a change in data rate. For instance, a filter capacitor of 0.005 uF will yield a settling time of approximately 500 microseconds. Since HIFIL generates a relatively low current signal (approximately 10 uA), care also needs to be taken to avoid external leakage on this pin. The quality of the capacitor, solder flux, grease, and dirt can all impact the amount of leakage on the board.

3.3 Write Precompensation

The write precompensation logic is used to minimize bit shifts in the RDDATA stream from the disk drive. The shifting of bits is a known phenomena of magnetic media and is dependent upon the disk media AND the floppy drive.

The 82077 monitors the bit stream that is being sent to the drive. The data patterns that require precompensation are well known. Depending upon the pattern, the bit is shifted either early or late (or not at all) relative to the surrounding bits. Figure 3-3 is a block diagram of the internal circuit.

The top block is a 13-bit shift register with the no delay tap being in the center. This allows 6 levels of early and late shifting with respect to nominal. The shift register is clocked at the main clock rate (24 MHz). The output is fed into 2 multiplexors—one for early and one for late. A final stage of multiplexors combines the early, late and normal data stream back into one which is the WRDATA output.

4.0 CONTROLLER PHASES

For simplicity, command handling in the 82077 can be divided into three phases: Command, Execution and Result. Each phase is described in the following secions.

4.1 Command Phase

After a reset, the 82077 enters the command phase and is ready to accept a command from the host. For each of the commands, a defined set of command code bytes and parameter bytes has to be written to the 82077 before the command phase is complete (Please refer to Section 5.0 for the command descriptions). These bytes of data must be transferred in the order prescribed.

Before writing to the 82077, the host must examine the RQM and DIO bits of the Main Status Register. RQM, DIO must be equal to "1" and "0" respectively before command bytes may be written. RQM is set false by the 82077 after each write cycle until the received byte is processed. The 82077 asserts RQM again to request each parameter byte of the command, unless an illegal command condition is detected. After the last parameter byte is received, RQM remains "0", and the 82077 automatically enters the next phase as defined by the command definition.

The FIFO is disabled during the command phase to retain compatibility with the 8272A, and to provide for the proper handling of the "Invalid Command" condition.

4.2 Execution Phase

All data transfers to or from the 82077 occur during the execution phase, which can proceed in DMA or non-DMA mode as indicated in the SPECIFY command.

After a reset, the FIFO is disabled. Each data byte is transferred by an INT or DRQ depending on the DMA mode. The CONFIGURE command can enable the FIFO and set the FIFO threshold value.

The following paragraphs detail the operation of the FIFO flow control. In these descriptions, <threshold> is defined as the number of bytes available to the 82077 when service is requested from the host, and ranges from 1 to 16. The parameter FIFOTHR which the user programs is one less, and ranges from 0 to 15.

A low threshold value (i.e. 2) results in longer periods of time between service requests, but requires faster servicing of the request, for both read and write cases. The host reads (writes) from (to) the FIFO until empty (full), then the transfer request goes inactive. The host must be very responsive to the service request. This is the desired case for use with a "fast" system.

A high value of threshold (i.e. 12) is used with a "sluggish" system by affording a long latency period after a service request, but results in more frequent service requests.

4.2.1 NON-DMA MODE, TRANSFERS FROM THE FIFO TO THE HOST

The INT pin and RQM bits in the Main Status Register are activated when the FIFO contains (16−<threshold>) bytes, or the last bytes of a full sector transfer have been placed in the FIFO. The INT pin can be used for interrupt driven systems and RQM can be used for polled sytems. The host must respond to the request by reading data from the FIFO. This process is repeated until the last byte is transferred out of the FIFO. The 82077 will deactivate the INT pin and RQM bit when the FIFO becomes empty.

4.2.2 NON-DMA MODE, TRANSFERS FROM THE HOST TO THE FIFO

The INT pin and RQM bit in the Main Status Register are activated upon entering the execution phase of data transfer commands. The host must respond to the request by writing data into the FIFO. The INT pin and RQM bit remain true until the FIFO becomes full. They are set true again when the FIFO has <threshold> bytes remaining in the FIFO. The INT pin will also be deactivated if TC and DACK# both go inactive. The 82077 enters the result phase after the last byte is taken by the 82077 from the FIFO (i.e. FIFO empty condition).

4.2.3 DMA MODE, TRANSFERS FROM THE FIFO TO THE HOST

The 82077 activates the DRQ pin when the FIFO contains (16−<threshold>) bytes, or the last byte of a full sector transfer has been placed in the FIFO. The DMA controller must respond to the request by reading data from the FIFO. The 82077 will deactivate the DRQ pin when the FIFO becomes empty. DRQ goes inactive after DACK# goes active for the last byte of a data transfer (or on the active edge of RD#, on the last byte, if no edge is present on DACK#). A data underrun may occur if DRQ is not removed in time to prevent an unwanted cycle.

4.2.4 DMA MODE, TRANSFERS FROM THE HOST TO THE FIFO

The 82077 activated the DRQ pin when entering the execution phase of the data transfer commands. The DMA controller must respond by activating the DACK# and WR# pins and placing data in the FIFO. DRQ remains active until the FIFO becomes full. DRQ is again set true when the FIFO has <threshold> bytes remaining in the FIFO. The 82077 will also deactivate the DRQ pin when TC becomes true (qualified by DACK#), indicating that no more data is required. DRQ goes inactive after DACK# goes active for the last byte of a data transfer (or on the active edge of WR# of the last byte, if no edge is present on DACK#). A data overrun may occur if DRQ is not removed in time to prevent an unwanted cycle.

4.2.5 DATA TRANSFER TERMINATION

The 82077 supports terminal count explicitly through the TC pin and implicitly through the underrun/overrun and end-of-track (EOT) functions. For full sector transfers, the EOT parameter can define the last sector to be transferred in a single or multisector transfer. If the last sector to be transferred is a partial sector, the host can stop transferring the data in mid-sector, and the 82077 will continue to complete the sector as if a hardware TC was received. The only difference between these implicit functions and TC is that they return "abnormal termination" result status. Such status indications can be ignored if they were expected.

Note that when the host is sending data to the FIFO of the 82077, the internal sector count will be complete when 82077 reads the last byte from its side of the FIFO. There may be a delay in the removal of the transfer request signal of up to the time taken for the 82077 to read the last 16 bytes from the FIFO. The host must tolerate this delay.

4.3 Result Phase

The generation of INT determines the beginning of the result phase. For each of the commands, a defined set of result bytes has to be read from the 82077 before the result phase is complete. (Refer to Section 5.0 on command descriptions.) These bytes of data must be read out for another command to start.

RQM and DIO must both equal "1" before the result bytes may be read from the FIFO. After all the result bytes have been read, the RQM and DIO bits switch to "1" and "0" respectively, and the CB bit is cleared. This indicates that the 82077 is ready to accept the next command.

5.0 COMMAND SET/DESCRIPTIONS

Commands can be written whenever the 82077 is in the command phase. Each command has a unique set of needed parameters and status results. The 82077 checks to see that the first byte is a valid command and, if valid, proceeds with the command. If it was invalid, an interrupt is issued. The user would send a SENSE INTERRUPT STATUS command which would return an invalid command error. Table 5-1 is a summary of the Command set.

Table 5-1. 82077 Command Set

Phase	R/W	DATA BUS								Remarks
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
					READ DATA					
Command	W	MT	MFM	SK	0	0	1	1	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W				C					Sector ID information prior to Command execution
	W				H					
	W				R					
	W				N					
	W				EOT					
	W				GPL					
	W				DTL					
Execution										Data transfer between the FDD and system
Result	R				ST 0					Status information after Command execution
	R				ST 1					
	R				ST 2					
	R				C					
	R				H					Sector ID information after Command execution
	R				R					
	R				N					
					READ DELETED DATA					
Command	W	MT	MFM	SK	0	1	1	0	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W				C					Sector ID information prior to Command execution
	W				H					
	W				R					
	W				N					
	W				EOT					
	W				GPL					
	W				DTL					
Execution										Data transfer between the FDD and system
Result	R				ST 0					Status information after Command execution
	R				ST 1					
	R				ST 2					
	R				C					
	R				H					Sector ID information after Command execution
	R				R					
	R				N					
					WRITE DATA					
Command	W	MT	MFM	0	0	0	1	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W				C					Sector ID information prior to Command execution
	W				H					
	W				R					
	W				N					
	W				EOT					
	W				GPL					
	W				DTL					
Execution										Data transfer between the system and FDD
Result	R				ST 0					Status information after Command execution
	R				ST 1					
	R				ST 2					
	R				C					
	R				H					Sector ID information after Command execution
	R				R					
	R				N					

Table 5-1. 82077 Command Set (Continued)

Phase	R/W	DATA BUS								Remarks
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
WRITE DELETED DATA										
Command	W	MT	MFM	0	0	1	0	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior to Command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and system
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								
	R	H								Sector ID information after Command execution
	R	R								
	R	N								
READ TRACK										
Command	W	0	MFM	0	0	0	0	1	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior to Command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and system. FDC reads all of cylinders contents from index hole to EOT
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								
	R	H								Sector ID information after Command execution
	R	R								
	R	N								
VERIFY										
Command	W	MT	MFM	SK	1	0	1	1	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior to Command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										No data transfer takes place
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								
	R	H								Sector ID information after Command execution
	R	R								
	R	N								
VERSION										
Command	W	0	0	0	1	0	0	0	0	Command Code
Result	R	1	0	0	1	0	0	0	0	Enhanced Controller

Table 5-1. 82077 Command Set (Continued)

Phase	R/W	DATA BUS								Remarks
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
FORMAT TRACK										
Command	W	0	MFM	0	0	1	1	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	N								Bytes/Sector
	W	SC								Sectors/Cylinder
	W	GPL								Gap 3
	W	D								Filler Byte
Execution										
For Each	W	C								
Sector	W	H								Input Sector Parameters
Repeat:	W	R								
	W	N								
										82077 formats an entire cylinder
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	Undefined								
	R	Undefined								
	R	Undefined								
	R	Undefined								
RECALIBRATE										
Command	W	0	0	0	0	0	1	1	1	Command Codes
	W	0	0	0	0	0	0	DS1	DS0	
Execution										Head retracted to Track 0 Interrupt
SENSE INTERRUPT STATUS										
Command	W	0	0	0	0	1	0	0	0	Command Codes
Result	R	ST 0								Status information at the end of each seek operation
	R	PCN								
SPECIFY										
Command	W	0	0	0	0	0	0	1	1	Command Codes
	W	SRT				HUT				
	W	HLT							ND	
SENSE DRIVE STATUS										
Command	W	0	0	0	0	0	1	0	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
Result	R	ST 3								Status information about FDD
SEEK										
Command	W	0	0	0	0	1	1	1	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	NCN								
Execution										Head is positioned over proper Cylinder on Diskette
CONFIGURE										
Command	W	0	0	0	1	0	0	1	1	Configure Information
	W	0	0	0	0	0	0	0	0	
	W	0	EIS	EFIFO	POLL	FIFOTHR				
	W	PRETRK								
RELATIVE SEEK										
Command	W	1	DIR	0	0	1	1	1	1	
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	RCN								

Table 5-1. 82077 Command Set (Continued)

Phase	R/W	DATA BUS								Remarks
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
DUMPREG										
Command	W	0	0	0	0	1	1	1	0	*Note Registers placed in FIFO
Execution										
Result	R	PCN-Drive 0								
	R	PCN-Drive 1								
	R	PCN-Drive 2								
	R	PCN-Drive 3								
	R	SRT				HUT				
	R	HLT							ND	
	R	SC/EOT								
	R	Undefined								
	R	0	EIS	EFIFO	POLL	FIFOTHR				
	R	PRETRK								
READ ID										
Command	W	0	MFM	0	0	1	0	1	0	Commands
	W	0	0	0	0	0	HDS	DS1	DS0	
Execution										The first correct ID information on the Cylinder is stored in Data Register
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								
	R	H								Disk status after the Command has completed.
	R	R								
	R	N								
PERPENDICULAR MODE										
Command	W	0	0	0	1	0	0	1	0	Command Codes
		0	0	0	0	0	0	WGATE	GAP	
INVALID										
Command	W	Invalid Codes								Invalid Command Codes (NoOp — 82077 goes into Standby State)
Result	R	ST 0								ST 0 = 80H

SC is returned if the last command that was issued was the FORMAT command. EOT is returned if the last command was a READ or WRITE.

NOTE:
These bits are used internally only. They are not reflected in the Drive Select pins. It is the users responsibility to maintain correspondence between these bits and the Drive Select pins (DOR).

PARAMETER ABBREVIATIONS

Symbol	Description
C	Cylinder address. The currently selected cylinder address, 0 to 255.
D	Data pattern. The pattern to be written in each sector data field during formatting.
DIR	Direction control. If this bit is 0, then the head will step out from the spindle during a relative seek. If set to a 1, the head will step in toward the spindle.
DS0, DS1	Disk Drive Select.

DS1	DS0	
0	0	drive 0
0	1	drive 1
1	0	drive 2
1	1	drive 3

Symbol	Description
DTL	Special sector size. By setting N to zero (00), DTL may be used to control the number of bytes transferred in disk read/write commands. The sector size (N = 0) is set to 128. If the actual sector (on the diskette) is larger than DTL, the remainder of the actual sector is read but is not passed to the host during read commands; during write commands, the remainder of the actual sector is written with all zero bytes. The CRC check code is calculated with the actual sector. When N is not zero, DTL has no meaning and should be set to FF HEX.
EFIFO	Enable FIFO. When this bit is 0, the FIFO is enabled. A "1" puts the 82077 in the 8272A compatible mode where the FIFO is disabled.
EIS	Enable implied seek. When set, a seek operation will be performed before executing any read or write command that requires the C parameter in the command phase. A "0" disables the implied seek.
EOT	End of track. The final sector number of the current track.
GPL	Gap length. The gap 3 size. (Gap 3 is the space between sectors excluding the VCO synchronization field).
H/HDS	Head address. Selected head: 0 or 1 (disk side 0 or 1) as encoded in the sector ID field.
HLT	Head load time. The time interval that 82077 waits after loading the head and before initiating a read or write operation. Refer to the SPECIFY command for actual delays.
HUT	Head unload time. The time interval from the end of the execution phase (of a read or write command) until the head is unloaded. Refer to the SPECIFY command for actual delays.
MFM	MFM/FM mode selector. A one selects the double density (MFM) mode. A zero selects single density (FM) mode.
MT	Multi-track selector. When set, this flag selects the multi-track operating mode. In this mode, the 82077 treats a complete cylinder, under head 0 and 1, as a single track. The 82077 operates as if this expanded track started at the first sector under head 0 and ended at the last sector under head 1. With this flag set, a multitrack read or write operation will automatically continue to the first sector under head 1 when the 82077 finishes operating on the last sector under head 0.

Symbol	Description
N	Sector size code. This specifies the number of bytes in a sector. If this parameter is "00", then the sector size is 128 bytes. The number of bytes transferred is determined by the DTL parameter. Otherwise the sector size is (2 raised to the "N'th" power) times 128. All values up to "07" hex are allowable. "07"h would equal a sector size of 16k. It is the users responsibility to not select combinations that are not possible with the drive.

N	Sector Size
00	128 bytes
01	256 bytes
02	512 bytes
03	1024 bytes
. .	. . .
07	16 Kbytes

Symbol	Description
NCN	New cylinder number. The desired cylinder number.
ND	Non-DMA mode flag. When set to 1, indicates that the 82077 is to operate in the non-DMA mode. In this mode, the host is interrupted for each data transfer. When set to 0, the 82077 operates in DMA mode, interfacing to a DMA controller by means of the DRQ and DACK# signals.
PCN	Present cylinder number. The current position of the head at the completion of SENSE INTERRUPT STATUS command.
POLL	Polling disable. When set, the internal polling routine is disabled. When clear, polling is enabled.
PRETRK	Precompensation start track number. Programmable from track 00 to FFH.
R	Sector address. The sector number to be read or written. In multi-sector transfers, this parameter specifies the sector number of the first sector to be read or written.
RCN	Relative cylinder number. Relative cylinder offset from present cylinder as used by the RELATIVE SEEK command.
SC	Number of sectors per track. The number of sectors per track to be initialized by the FORMAT command.
SK	Skip flag. When set to 1, sectors containing a deleted data address mark will automatically be skipped during the execution of READ DATA. If READ DELETED is executed, only sectors with a deleted address mark will be accessed. When set to "0", the sector is read or written the same as the read and write commands.

Symbol	Description
SRT	Step rate interval. The time interval between step pulses issued by the 82077. Programmable from 0.5 to 8 milliseconds, in increments of 0.5 ms at the 1 Mbit data rate. Refer to the SPECIFY command for actual delays.
ST0 ST1 ST2 ST3	Status register 0–3. Registers within the 82077 that store status information after a command has been executed. This status information is available to the host during the result phase after command execution.

5.1 Data Transfer Commands

All of the READ DATA, WRITE DATA and VERIFY type commands use the same parameter bytes and return the same results information. The only difference being the coding of bits 0–4 in the first byte.

An implied seek will be executed if the feature was enabled by the CONFIGURE command. This seek is completely transparent to the user. The Drive Busy bit for the drive will go active in the Main Status Register during the seek portion of the command. If the seek portion fails, it will be reflected in the results status normally returned for a READ/WRITE DATA command. Status Register 0 (ST0) would contain the error code and C would contain the cylinder on which the seek failed.

5.1.1 READ DATA

A set of nine (9) bytes is required to place the 82077 into the Read Data Mode. After the READ DATA command has been issued, the 82077 loads the head (if it is in the unloaded state), waits the specified head settling time (defined in the SPECIFY command), and begins reading ID Address Marks and ID fields. When the sector address read off the diskette matches with the sector address specified in the command, the 82077 reads the sector's data field and transfers the data to the FIFO.

After completion of the read operation from the current sector, the sector address is incremented by one, and the data from the next logical sector is read and output via the FIFO. This continuous read function is called "Multi-Sector Read Operation". Upon receipt of TC, or an implied TC (FIFO overrun/underrun), the 82077 stops sending data, but will continue to read data from the current sector, check the CRC bytes, and at the end of the sector terminate the READ DATA Command.

N determines the number of bytes per sector (see Table 5-2 below). If N is set to zero, the sector size is set to 128. The DTL value determines the number of bytes to be transferred. If DTL is less than 128, the 82077 transfers the specified number of bytes to the host. For reads, it continues to read the entire 128 byte sector and checks for CRC errors. For writes it completes the 128 byte sector by filling in zeroes. If N is not set to 00 Hex, DTL should be set to FF Hex, and has no impact on the number of bytes transferred.

Table 5-2. Sector Sizes

N	Sector Size
00	128 bytes
01	256 bytes
02	512 bytes
03	1024 bytes
..	...
07	16 Kbytes

The amount of data which can be handled with a single command to the 82077 depends upon MT (multi-track) and N (Number of bytes/sector).

Table 5-3. Effects of MT and N Bits

MT	N	Max. Transfer Capacity	Final Sector Read from Disk
0	1	256 × 26 = 6,656	26 at side 0 or 1
1	1	256 × 52 = 13,312	26 at side 1
0	2	512 × 15 = 7,680	15 at side 0 or 1
1	2	512 × 30 = 15,360	15 at side 1
0	3	1024 × 8 = 8,192	8 at side 0 or 1
1	3	1024 × 16 = 16,384	16 at side 1

The Multi-Track function (MT) allows the 82077 to read data from both sides of the diskette. For a particular cylinder, data will be transferred starting at Sector 1, Side 0 and completing at the last sector of the same track at Side 1.

If the host terminates a read or write operation in the 82077, then the ID information in the result phase is dependent upon the state of the MT bit and EOT byte. Refer to Table 5-6.

At the completion of the READ DATA Command, the head is not unloaded until after the Head Unload Time Interval (specified in the SPECIFY command) has elapsed. If the host issues another command before the head unloads then the head settling time may be saved between subsequent reads.

If the 82077 detects a pulse on the IDX pin twice without finding the specified sector (meaning that the diskette's index hole passes through index detect logic in the drive twice), the 82077 sets the IC code in Status Register 0 to "01" (Abnormal termination), and sets the ND bit in Status Register 1 to "1" indicating a sector not found, and terminates the READ DATA Command.

After reading the ID and Data Fields in each sector, the 82077 checks the CRC bytes. If a CRC error occurs in the ID or data field, the 82077 sets the IC code in Status Register 0 to "01" (Abnormal termination), sets the DE bit flag in Status Register 1 to "1", sets the DD bit in Status Register 2 to "1" if CRC is incorrect in the ID field, and terminates the READ DATA Command.

Table 5-4 below describes the affect of the SK bit on the READ DATA command execution and results.

Table 5-4. Skip Bit vs READ DATA Command

SK Bit Value	Data Address Mark Type Encountered	Results		
		Sector Read?	CM Bit of ST2 Set?	Description of Results
0	Normal Data	Yes	No	Normal Termination.
0	Deleted Data	Yes	Yes	Address Not Incremented. Next Sector Not Searched For.
1	Normal Data	Yes	No	Normal Termination.
1	Deleted Data	No	Yes	Normal Termination Sector Not Read ("Skipped").

Except where noted in Table 5-4, the C or R value of the sector address is automatically incremented (see Table 5-6).

5.1.2 READ DELETED DATA

This command is the same as the READ DATA command, only it operates on sectors that contain a Deleted Data Address Mark at the beginning of a Data Field.

Table 5-5 describes the affect of the SK bit on the READ DELETED DATA command execution and results.

Table 5-5. Skip Bit vs READ DELETED DATA Command

SK Bit Value	Data Address Mark Type Encountered	Results		
		Sector Read?	CM Bit of ST2 Set?	Description of Results
0	Normal Data	Yes	Yes	Address Not Incremented. Next Sector Not Searched For.
0	Deleted Data	Yes	No	Normal Termination.
1	Normal Data	No	Yes	Normal Termination Sector Not Read ("Skipped").
1	Deleted Data	Yes	No	Normal Termination.

Except where noted in Table 5-5 above, the C or R value of the sector address is automatically incremented (See Table 5-6).

5.1.3 READ TRACK

This command is similar to the READ DATA command except that the entire data field is read continuously from each of the sectors of a track. Immediately after encountering a pulse on the IDX pin, the 82077 starts to read all data fields on the track as continuous blocks of data without regard to logical sector numbers. If the 82077 finds an error in the ID or DATA CRC check bytes, it continues to read data from the track and sets the appropriate error bits at the end of the command. The 82077 compares the

Table 5-6. Result Phase Table

MT	Head	Final Sector Transferred to Host	ID Information at Result Phase			
			C	H	R	N
0	0	Less than EOT	NC	NC	R+1	NC
		Equal to EOT	C+1	NC	01	NC
	1	Less than EOT	NC	NC	R+1	NC
		Equal to EOT	C+1	NC	01	NC
1	0	Less than EOT	NC	NC	R+1	NC
		Equal to EOT	NC	LSB	01	NC
	1	Less than EOT	NC	NC	R+1	NC
		Equal to EOT	C+1	LSB	01	NC

NC: no change, the same value as the one at the beginning of command execution.
LSB: least significant bit, the LSB of H is complemented.

ID information read from each sector with the specified value in the command, and sets the ND flag of Status Register 1 to a "1" if there is no comparison. Multi-track or skip operations are not allowed with this command. The SK bit should always be set to "0".

This command terminates when the EOT specified number of sectors have been read. If the 82077 does not find an ID Address Mark on the diskette after the second occurrence of a pulse on the IDX pin, then it sets the IC code in Status Register 0 to "01" (Abnormal termination), sets the MA bit in Status Register 1 to "1", and terminates the command.

5.1.4 WRITE DATA

After the WRITE DATA command has been issued, the 82077 loads the head (if it is in the unloaded state), waits the specified head load time if unloaded (defined in the SPECIFY command), and begins reading ID Fields. When the sector address read from the diskette matches the sector address specified in the command, the 82077 reads the data from the host via the FIFO, and writes it to the sector's data field.

After writing data into the current sector, the 82077 computes the CRC value and writes it into the CRC field at the end of the sector transfer. The Sector Number stored in "R" is incremented by one, and the 82077 continues writing to the next data field. The 82077 continues this "Multi-Sector Write Operation". Upon receipt of a terminal count signal or if a FIFO over/under run occurs while a data field is being written, then the remainder of the data field is filled with zeros.

The 82077 reads the ID field of each sector and checks the CRC bytes. If it detects a CRC error in one of the ID Fields, it sets the IC code in Status Register 0 to "01" (Abnormal termination), sets the DE bit of Status Register 1 to "1", and terminates the WRITE DATA command.

The WRITE DATA command operates in much the same manner as the READ DATA command. The following items are the same. Please refer to the READ DATA Command for details:

- Transfer Capacity
- EN (End of Cylinder) bit
- ND (No Data) bit
- Head Load, Unload Time Interval
- ID information when the host terminates the command.
- Definition of DTL when N = 0 and when N does not = 0.

5.1.5 WRITE DELETED DATA

This command is almost the same as the WRITE DATA command except that a Deleted Data Address Mark is written at the beginning of the Data Field instead of the normal Data Address Mark. This command is typically used to mark a bad sector containing an error on the floppy disk.

5.1.6 VERIFY

The VERIFY command is used to verify the data stored on a disk. This command acts exactly like a READ DATA command except that no data is transferred to the host. Data is read from the disk, CRC computed and checked against the previously stored value.

The EOT value should be set to the final sector to be checked. If EOT is greater than the number of sectors on a disk, the command will stop due to an error and no useful CRC information will be obtained.

5.1.7 FORMAT TRACK

The FORMAT command allows an entire track to be formatted. After a pulse from the IDX pin is detected, the 82077 starts writing data on the disk including Gaps, Address Marks, ID Fields and Data Fields, per the IBM System 34 or 3740 format (MFM or FM respectively). The particular values that will be written to the gap and data field are controlled by the values programmed into N, SC, GPL, and D which are specified by the host during the command phase. The data field of the sector is filled with the data byte specified by D. The ID Field for each sector is supplied by the host; that is, four data bytes per sector are needed by the 82077 for C, H, R, and N (cylinder, head, sector number and sector size respectively).

After formatting each sector, the host must send new values for C, H, R and N to the 82077 for the next sector on the track. The R value (sector number) is the only value that must be changed by the host after each sector is formatted. This allows the disk to be formatted with nonsequential sector addresses (interleaving). This incrementing and formatting continues for the whole track until the 82077 encounters a pulse on the IDX pin again and it terminates the command.

Table 5-7 contains typical values for gap fields which are dependent upon the size of the sector and the number of sectors on each track. Actual values can vary due to drive electronics.

Table 5-7. Typical Values for Formatting

		Sector Size	N	SC	GPL1	GPL2
5.25" Drives	FM	128	00	12	07	09
		128	00	10	10	19
		512	02	08	18	30
		1024	03	04	46	87
		2048	04	02	C8	FF
		4096	05	01	C8	FF
		...	...			
	MFM	256	01	12	0A	0C
		256	01	10	20	32
		512*	02	09	2A	50
		1024	03	04	80	F0
		2048	04	02	C8	FF
		4096	05	01	C8	FF
		...	...			
3.5" Drives	FM	128	0	0F	07	1B
		256	1	09	0F	2A
		512	2	05	1B	3A
	MFM	256	1	0F	0E	36
		512**	2	09	1B	54
		1024	3	05	35	74

GPL1 = suggested GPL values in read and write commands to avoid splice point between data field and ID field of contiguous sections.
GPL2 = suggested GPL value in FORMAT TRACK command.
*PC-AT values (typical)
**PS/2™ values (typical)

NOTE:
All values except Sector Size are in Hex.

5.1.7.1 Format Fields

GAP 4a	SYNC	IAM		GAP 1	SYNC	IDAM		CYL	HD	SEC	NO	CRC	GAP 2	SYNC	DATA AM		DATA	CRC	GAP 3	GAP 4b
80x 4E	12x 00	3x C2	FC	50x 4E	12x 00	3x A1	FE						22x 4E	12x 00	3x A1	FB F8				

Figure 5-1. System 34 Format Double Density

GAP 4a	SYNC	IAM	GAP 1	SYNC	IDAM	CYL	HD	SEC	NO	CRC	GAP 2	SYNC	DATA AM	DATA	CRC	GAP 3	GAP 4b
40x FF	6x 00	FC	26x FF	6x 00	FE						11x FF	6x 00	FB or F8				

Figure 5-2. System 3740 Format Single Density

GAP 4a	SYNC	IAM		GAP 1	SYNC	IDAM		CYL	HD	SEC	NO	CRC	GAP 2	SYNC	DATA AM		DATA	CRC	GAP 3	GAP 4b
80x 4E	12x 00	3x C2	FC	50x 4E	12x 00	3x A1	FE						41x 4E	12x 00	3x A1	FB F8				

Figure 5-3. Toshiba Perpendicular Format

5.2 Control Commands

Control commands differ from the other commands in that no data transfer takes place. Three commands generate an interrupt when complete; READ ID, RECALIBRATE and SEEK. The other control commands do not generate an interrupt.

5.2.1 READ ID

The READ ID command is used to find the present position of the recording heads. The 82077 stores the values from the first ID Field it is able to read into its registers. If the 82077 does not find an ID Address Mark on the diskette after the second occurrence of a pulse on the IDX pin, it then sets the IC code in Status Register 0 to "01" (Abnormal termination), sets the MA bit in Status Register 1 to "1", and terminates the command.

The following commands will generate an interrupt upon completion. They do not return any result bytes. It is highly recommended that control commands be followed by the SENSE INTERRUPT STATUS command. Otherwise, valuable interrupt status information will be lost.

5.2.2 RECALIBRATE

This command causes the read/write head within the 82077 to retract to the track 0 position. The 82077 clears the contents of the PCN counter, and checks the status of the TRK0 pin from the FDD. As long as the TRK0 pin is low, the DIR pin remains 0 and step pulses are issued. When the TRK0 pin goes high, the SE bit in Status Register 0 is set to "1", and the command is terminated. If the TRK0 pin is still low after 255 step pulses have been issued, the 82077 sets the SE and the EC bits of Status Register 0 to "1", and terminates the command. Disks capable of handling more than 256 tracks per side may require more than one RECALIBRATE command to return the head back to physical Track 0.

The RECALIBRATE command does not have a result phase. SENSE INTERRUPT STATUS command must be issued after the RECALIBRATE command to effectively terminate it and to provide verification of the head position (PCN). During the command phase of the recalibrate operation, the 82077 is in the BUSY state, but during the execution phase it is in a NON BUSY state. At this time another RECALIBRATE command may be issued, and in this manner, parallel RECALIBRATE operations may be done on up to 4 drives at once.

Upon power up, the software must issue a RECALIBRATE command to properly initialize all drives and the controller.

5.2.3 SEEK

The read/write head within the drive is moved from track to track under the control of the SEEK Command. The 82077 compares the PCN which is the current head position with the NCN and performs the following operation if there is a difference:

—PCN < NCN: Direction signal to drive set to "1" (step in), and issues step pulses.

—PCN > NCN: Direction signal to drive set to "0" (step out), and issues step pulses.

The rate at which step pulses are issued is controlled by SRT (Stepping Rate Time) in the SPECIFY command. After each step pulse is issued, NCN is compared against PCN, and when NCN = PCN, then the SE bit in Status Register 0 is set to "1", and the command is terminated.

During the command phase of the seek or recalibrate operation, the 82077 is in the BUSY state, but during the execution phase it is in the NON BUSY state. At this time another SEEK or RECALIBRATE command may be issued, and in this manner, parallel seek operations may be done on up to 4 drives at once.

Note that if implied seek is not enabled, the read and write commands should be preceded by:

1) SEEK command;	Step to the proper track
2) SENSE INTERRUPT STATUS command;	Terminate the Seek command
3) READ ID.	Verify head is on proper track
4) Issue READ/WRITE command.	

The SEEK command does not have a result phase. Therefore, it is highly recommended that the SENSE INTERRUPT STATUS Command be issued after the SEEK command to terminate it and to provide verification of the head position (PCN). The H bit (Head Address) in ST0 will always return a "0". When exiting POWERDOWN mode, the 82077 clears the PCN value and the status information to zero. Prior to issuing the POWERDOWN command, it is highly recommended that the user service all pending interrupts through the SENSE INTERRUPT STATUS command.

5.2.4 SENSE INTERRUPT STATUS

An interrupt signal on INT pin is generated by the 82077 for one of the following reasons:

1. Upon entering the Result Phase of:
   a. READ DATA Command
   b. READ TRACK Command
   c. READ ID Command
   d. READ DELETED DATA Command
   e. WRITE DATA Command
   f. FORMAT TRACK Command
   g. WRITE DELETED DATA Command
2. End of SEEK, RELATIVE SEEK or RECALIBRATE Command
3. 82077 requires a data transfer during the execution phase in the non-DMA Mode

The SENSE INTERRUPT STATUS command resets the interrupt signal and via the IC code and SE bit of Status Register 0, identifies the cause of the interrupt.

Table 5-8. Interrupt Identification

SE	IC	Interrupt Due To
0	11	Polling
1	00	Normal Termination of SEEK or RECALIBRATE command
1	01	Abnormal Termination of SEEK or RECALIBRATE command

The SEEK, RELATIVE SEEK and the RECALIBRATE commands have no result phase. SENSE INTERRUPT STATUS command must be issued immediately after these commands to terminate them and to provide verification of the head position (PCN). The H (Head Address) bit in ST0 will always return a "0". If a SENSE INTERRUPT STATUS is not issued, the drive, will continue to be BUSY and may effect the operation of the next command.

5.2.5 SENSE DRIVE STATUS

SENSE DRIVE STATUS obtains drive status information. It has no execution phase and goes directly to the result phase from the command phase. STATUS REGISTER 3 contains the drive status information.

5.2.6 SPECIFY

The SPECIFY command sets the initial values for each of the three internal timers. The HUT (Head Unload Time) defines the time from the end of the execution phase of one of the read/write commands to the head unload state. The SRT (Step Rate Time) defines the time interval between adjacent step pulses. Note that the spacing between the first and second step pulses may be shorter than the remaining step pulses. The HLT (Head Load Time) defines the time between the Head Load signal goes high and the read, write operation starts. The values change with the data rate speed selection and are documented in Table 5-9. The values are the same for MFM and FM.

Table 5-9. Drive Control Delays (ms)

	HUT				SRT			
	1M	**500K**	**300K**	**250K**	**1M**	**500K**	**300K**	**250K**
0	128	256	426	512	8.0	16	26.7	32
1	8	16	26.7	32	7.5	15	25	30
..	..	..	..	..	..	..	..	..
E	112	224	373	448	1.0	2	3.33	4
F	120	240	400	480	0.5	1	1.67	2

	HLT			
	1M	**500K**	**300K**	**250K**
00	128	256	426	512
01	1	2	3.3	4
02	2	4	6.7	8
..	..	..	..	.
7F	126	252	420	504
7F	127	254	423	508

The choice of DMA or NON-DMA operations is made by the ND bit. When this bit is "1", the NON-DMA mode is selected, and when ND is "0", the DMA mode is selected. In DMA mode, data transfers are signalled by the DRQ pin. Non-DMA mode uses the RQM bit and the INT pin to signal data transfers.

5.2.7 CONFIGURE

Issued to select the special features of the 82077. A CONFIGURE command need not be issued if the default values of the 82077 meet the system requirements.

CONFIGURE DEFAULT VALUES:

EIS	—No Implied Seeks
EFIFO	—FIFO Disabled
POLL	—Polling Enabled
FIFOTHR	—FIFO Threshold Set to 1 Byte
PRETRK	—Pre-Compensation Set to Track 0

EIS—Enable implied seek. When set to "1", the 82077 will perform a SEEK operation before executing a read or write command. Defaults to no implied seek.

EFIFO—A "1" puts the FIFO into the 8272A compatible mode where the FIFO is disabled. This means data transfers are asked for on a byte by byte basis. Defaults to "1", FIFO disabled. The threshold defaults to one.

POLL—Disable polling of the drives. Defaults to "0", polling enabled. When enabled, a single interrupt is generated after a RESET. No polling is performed while the drive head is loaded and the head unload delay has not expired.

FIFOTHR—The FIFO threshold in the execution phase of read or write commands. This is programmable from 1 to 16 bytes. Defaults to one byte. A "00" selects one byte "0F" selects 16 bytes.

PRETRK—Pre-compensation start track number. Programmable from track 0 to 255. Defaults to track 0. A "00" selects track 0, "FF" selects 255.

5.2.8 VERSION

The VERSION command checks to see if the controller is an enhanced type or the older type (8272A/765A). A value of 90 H is returned immediately after the command byte and no interrupts are generated.

5.2.9 RELATIVE SEEK

The command is coded the same as for SEEK, except for the MSB of the first byte and the DIR bit.

DIR Head Step Direction Control.

DIR	Action
0	Step Head Out
1	Step Head In

RCN Relative Cylinder Number that determines how many tracks to step the head in or out from the current track number.

The RELATIVE SEEK command differs from the SEEK command in that it steps the head the absolute number of tracks specified in the command instead of making a comparison against an internal register. The SEEK command is good for drives that support a maximum of 256 tracks. RELATIVE SEEKs cannot be overlapped with other RELATIVE SEEKs. Only one RELATIVE SEEK can be active at a time. RELATIVE SEEKs may be overlapped with SEEKs and RECALIBRATEs. Bit 4 of Status Register 0 (EC) will be set if RELATIVE SEEK attempts to step outward beyond Track 0.

As an example, assume that a floppy drive has 300 useable tracks and that the host needs to read track 300 and the head is on any track (0–255). If a SEEK command was issued, the head would stop at track 255. If a RELATIVE SEEK command was issued, the 82077 would move the head the specified number of tracks, regardless of the internal cylinder position register (but would increment the register). If the head had been on track 40 (D), the maximum track that the 82077 could position the head on using RELATIVE SEEK, would be 296 (D), the initial track, +256 (D). The maximum count that the head can be moved with a single RELATIVE SEEK command is 256 (D).

The internal register, PCN, would overflow as the cylinder number crossed track 255 and would contain 40 (D). The resulting PCN value is thus (NCN + PCN) mod 256. Functionally, the 82077 starts counting from 0 again as the track number goes above 255(D). It is the users responsibility to compensate 82077 functions (precompensation track number) when accessing tracks greater than 255. The 82077 does not keep track that it is working in an "extended track area" (greater than 255). Any command issued would use the current PCN value except for the RECALIBRATE command which only looks for the TRACK0 signal. RECALIBRATE would return an error if the head was farther than 255 due to its limitation of issuing a maximum 256 step pulses. The user simply needs to issue a second RECALIBRATE command. The SEEK command and implied seeks will function correctly within the 44 (D) track (299–255) area of the "extended track area". It is the users responsibility not to issue a new track position that would exceed the maximum track that is present in the extended area.

To return to the standard floppy range (0–255) of tracks, a RELATIVE SEEK would be issued to cross the track 255 boundary.

A RELATIVE SEEK can be used instead of the normal SEEK but the host is required to calculate the difference between the current head location and the new (target) head location. This may require the host to issue a READ ID command to ensure that the head is physically on the track that software assumes it to be. Different 82077 commands will return different cylinder results which may be difficult to keep track of with software without the READ ID command.

5.2.10 DUMPREG

The DUMPREG command is designed to support system run-time diagnostics and application software development and debug.

5.2.11 PERPENDICULAR MODE COMMAND

The PERPENDICULAR MODE command should be issued prior to executing READ/WRITE/FORMAT commands that access a disk drive with perpendicular recording capability. With this command, the length of the Gap2 field and VCO enable timing can be altered to accommodate the unique requirements of these drives. Table 5-10 describes the effects of the WGATE and GAP bits for the PERPENDICULAR MODE command. Upon a reset, the 82077 will default to the conventional mode (WGATE = 0, GAP = 0).

Table 5-10 Effects of WGATE and GAP Bits

WGATE	GAP	MODE	VCO Low Time after Index Pulse	Length of Gap2 Format Field	Portion of Gap2 Written by Write Data Operation	Gap2 VCO Low Time for Read Operations
0	0	Conventional Mode	33 Bytes	22 Bytes	0 Bytes	24 Bytes
0	1	Reserved (Conventional)	33 Bytes	22 Bytes	0 Bytes	24 Bytes
1	0	Perpendicular Mode (500 Kbps Data Rate)	33 Bytes	22 Bytes	19 Bytes	24 Bytes
1	1	Perpendicular Mode (1 Mbps Data Rate)	18 Bytes	41 Bytes	38 Bytes	43 Bytes

Selection of the 500 Kbps and 1 Mbps perpendicular modes is independent of the actual data rate selected in the Data rate Select Register. The user must ensure that the two data rates remain consistent.

The Gap2 and VCO timing requirements for perpendicular recording type drives are dictated by the design of the read/write head. In the design of this head, a pre-erase head precedes the normal read/write head by a distance of 200 micrometers. This works out to about 38 bytes at a 1 Mbps recording density. Whenever the write head is enabled by the Write Gate signal the pre-erase head is also activated at the same time. Thus, when the write head is initially turned on, flux transitions recorded on the media for the first 38 bytes will not be preconditioned with the pre-erase head since it has not yet been activated. To accommodate this head activation and deactivation time, the Gap2 field is expanded to a length of 41 bytes. The format field shown in Figure 5-3 illustrates the change in the Gap2 field size for the perpendicular format.

On the read back by the 82077, the controller must begin synchronization at the beginning of the Sync field. For the conventional mode, the internal PLL VCO is enabled (VCOEN) approximately 24 bytes from the start of the Gap2 field. But when the controller operates in the 1 Mbps perpendicular mode (WGATE = 1, GAP = 1), VCOEN goes active after 43 bytes to accommodate the increased Gap2 field size. For both cases, an approximate 2 byte cushion is maintained from the beginning of the sync field for the purposes of avoiding write splices in the presence of motor speed variation.

For the WRITE DATA case, the 82077 activates Write Gate at the beginning of the sync field under the conventional mode. The controller then writes a new sync field, data address mark, data field, and CRC as shown in Figure 5-1. With the pre-erase head of the perpendicular drive, the write head must be activated in the Gap2 field to insure a proper write of the new sync field. For the 1 Mbps perpendicular mode (WGATE = 1, GAP = 1), 38 bytes will be written in the Gap2 space. Since the bit density is proportional to the data rate, 19 bytes will be written in the Gap2 field for the 500 Kbps perpendicular mode (WGATE = 1, GAP = 0).

It should be noted that none of the alterations in Gap2 size, VCO timing, or Write Gate timing affect normal program flow. The information provided here is just for background purposes and is not needed for normal operation. Once the PERPENDICULAR MODE command is invoked, 82077 software behavior from the user standpoint is unchanged.

6.0 STATUS REGISTER ENCODING

The contents of these registers are available only through a command sequence.

6.1 Status Register 0

Bit No.	Symbol	Name	Description
7, 6	IC	Interrupt Code	00-Normal termination of command. The specified command was properly executed and completed without error. 01-Abnormal termination of command. Command execution was started, but was not successfully completed. 10-Invalid command. The requested command could not be executed. 11-Abnormal termination caused by Polling.
5	SE	Seek End	The 82077 completed a SEEK or RECALIBRATE command, or a READ or WRITE with implied seek command.
4	EC	Equipment Check	The TRK0 pin failed to become a "1" after: 1. 256 step pulses in the RECALIBRATE command. 2. The RELATIVE SEEK command causes the 82077 to step outward beyond Track 0.
3	—	—	Unused. This bit is always "0".
2	H	Head Address	The current head address.
1, 0	DS1, 0	Drive Select	The current selected drive.

6.2 Status Register 1

Bit No.	Symbol	Name	Description
7	EN	End of Cylinder	The 82077 tried to access a sector beyond the final sector of the track (255D). Will be set if TC is not issued after Read or Write Data Command.
6	—	—	Unused. This bit is always "0".
5	DE	Data Error	The 82077 detected a CRC error in either the ID field or the data field of a sector.
4	OR	Overrun/ Underrun	Becomes set if the 82077 does not receive CPU or DMA service within the required time interval, resulting in data overrun or underrun.
3	—	—	Unused. This bit is always "0".
2	ND	No Data	Any one of the following: 1. READ DATA, READ DELETED DATA command, the 82077 did not find the specified sector. 2. READ ID command, the 82077 cannot read the ID field without an error. 3. READ TRACK command, the 82077 cannot find the proper sector sequence.
1	NW	Not Writable	WP pin became a "1" while the 82077 is executing a WRITE DATA, WRITE DELETED DATA, or FORMAT TRACK command.
0	MA	Missing Address Mark	Any one of the following: 1. The 82077 did not detect an ID address mark at the specified track after encountering the index pulse from the IDX pin twice. 2. The 82077 cannot detect a data address mark or a deleted data address mark on the specified track.

6.3 Status Register 2

Bit No.	Symbol	Name	Description
7	—	—	Unused. This bit is always "0".
6	CM	Control Mark	Any one of the following: 1. READ DATA command, the 82077 encounters a deleted data address mark. 2. READ DELETED DATA command, the 82077 encounters a data address mark.
5	DD	Data Error in Data Field.	The 82077 detected a CRC error in the data field.
4	WC	Wrong Cylinder	The track address from the sector ID field is different from the track address maintained inside the 82077.
3	—	—	Unused. This bit is always "0".
2	—	—	Unused. This bit is always "0".
1	BC	Bad Cylinder	The track address from the sector ID field is different from the track address maintained inside the 82077 and is equal to FF hex which indicates a bad track with a hard error according to the IBM soft-sectored format.
0	MD	Missing Data Address Mark	The 82077 cannot detect a data address mark or a deleted data address mark.

6.4 Status Register 3

Bit No.	Symbol	Name	Description
7	—	—	Unused. This bit is always "0".
6	WP	Write Protected	Indicates the status of the WP pin.
5	—	—	Unused. This bit is always "1".
4	T0	TRACK 0	Indicates the status of the TRK0 pin.
3	—	—	Unused. This bit is always "1".
2	HD	Head Address	Indicates the status of the HDSEL pin.
1, 0	DS1, 0	Drive Select	Indicates the status of the DS1, DS0 pins.

7.0 COMPATIBILITY

The 82077 was designed with software compatibility in mind. It is a fully backwards compatible solution with the older generation 8272A and NEC765A/B disk controllers. The 82077 also implements on-board registers for compatibility with the Personal System/2 as well as PC/AT and PC/XT floppy disk controller subsystems. Upon a hardware reset of the 82077, all registers, functions and enhancements default to a PS/2 or PC/AT compatible operating mode depending on how IDENT is strapped.

7.1 Register Set Compatibility

The register set contained within the 82077 is a culmination of hardware registers based on the architectural growth of the IBM personal computer line. Table 7-1 indicates the registers required for compatibility based on the type of computer.

Table 7-1. 82077 Register Support

82077 Register	8272A	82072	PC/XT	PC/AT	PS/2
SRA					X
SRB					X
DOR			X	X	X
MSR	X	X	X	X	X
DSR		X			
Data (FIFO)	X	X	X	X	X
DIR				X	X
CCR		X*		X	X

*CCR is emulated by DSR in an 82072 PC/AT design.

7.2 PS/2 Mode vs. AT Mode

To maintain compatibility with both the PS/2 environment and PC/AT environment the IDENT pin is provided. If this pin is strapped to VSS, the 82077 will operate in the PS/2 compatible mode. Strapping IDENT to V_{CC} places the 82077 in the PC/AT compatible mode. The differences between the two modes are described in the following sections.

7.2.1 PS/2 MODE

The polarity of the DENSEL output signal is designed to be compatible with the 3½″ disk drive standard. Programming the CCR or DSR to the upper data rates of 500 Kbps or 1 Mbps will cause DENSEL to go active low (assuming INVERT# is low). A comprehensive description of the DENSEL behavior is given in Table 2-6.

The DMAGATE# bit in the Digital Output Register (DOR) will not cause the DRQ or INT output signals to tristate. This maintains consistency with the operation of the floppy disk controller subsystem in the PS/2 architecture.

TC is an active low input signal that is internally qualified by DACK# being active low.

7.2.2 PC/AT MODE

The polarity of DENSEL is designed to be compatible with the 5¼″ high capacity disk drive standard. Programming the CCR or DSR to the lower data rate of 250 Kbps or 300 Kbps will cause DENSEL to go active low (assuming INVERT# is low).

If the DMAGATE# bit is written to a "0" in the Digital Output Register (DOR), DRQ and INT will tristate. If DMAGATE# is written to a "1", then DRQ and INT will be driven appropriately by the 82077.

TC is an active high input signal that is internally qualified by DACK# being active low.

7.3 Compatibility with the FIFO

The FIFO of the 82077 is designed to be transparent to non-FIFO disk controller software developed on the older generation 8272A standard. Operation of the 82077 FIFO can be broken down into two tiers of compatibility. For first tier compatibility, the FIFO is left in the default disabled condition upon a reset. In this mode the FIFO operates in a byte mode and provides complete compability with non-FIFO based software. For second tier compatibility, the FIFO is enabled via the CONFIGURE command. When the FIFO is enabled, it will temporarily enter a byte mode during the command and result phase of disk controller operation. This allows for compatible operation when interrogating the Main Status Register (MSR) for the purpose of transferring a byte at a time to or from the disk controller. For normal disk controller applications, the system designer can still take advantage of the FIFO for time critical data transfers during the execution phase and not create any conflicts with non-FIFO software during the command or result phase.

In some instances, use of the FIFO in any form has conflicted with certain specialized software. An example of a compatibility conflict using the FIFO is with software that monitors the progress of a data transfer during the execution phase. If the software assumed the disk controller was operating in a single byte mode and counted the number of bytes transferred to or from the disk controller to trigger some time dependent event on the disk media (i.e. head position over a specific data field), the same software will not have an identical time relationship if the FIFO is enabled. This is because the FIFO allows data to be queued up, and then burst transferred across the host bus. To accommodate software of this type, it is recommended that the FIFO be disabled.

7.4 Drive Polling

The 82077 supports the polling mode of the older generation 8272A. This mode is enabled upon a reset and can be disabled via the CONFIGURE command. This mode is supported for the sole purpose of providing backwards compatibility with software that expects it's presence.

The intended purpose of drive polling dates back to 8″ drives as a means to monitor any change in status for each disk drive present in the system. Each of the drives is selected for a period of time and its READY signal sampled. After a delay, the next drive is selected. Since the 82077 does not support READY in this capacity (internally tied true), the polling sequence is only simulated and does not affect the drive select lines (DS0–DS3) when it is active. If enabled, it occurs whenever the 82077 is waiting for a command or during SEEKs and RECALIBRATEs (but not IMPLIED SEEKs). Each drive is assumed to be not ready after a reset and a "ready" value for each drive is saved in an internal register as the simulated drive is polled. An interrupt will be generated on the first polling loop because of the initial "not ready" status. This interrupt must be followed with a SENSE INTERRUPT STATUS command from the host to clear the interrupt condition for each of the four logical drives.

8.0 PROGRAMMING GUIDELINES

Programming the 82077 is identical to any other 8272A compatible disk controller with the exception of some additional commands. For the new designer it is useful to provide some guidelines on how to program the 82077. A typical disk operation involves more than issuing a command and waiting for the results. The control of the floppy disk drive is a low level operation that requires software intervention at different stages. New commands and features have been added to the 82077 to reduce the complexity of this software interface.

8.1 Command and Result Phase Handshaking

Before a command or parameter byte can be issued to the 82077, the Main Status Register (MSR) must be interrogated for a ready status and proper FIFO direction. A typical floppy controller device driver should contain a subroutine for sending command or parameter bytes. For this discussion, the routine will be called "Send_byte" with the flowchart shown in Figure 8-1.

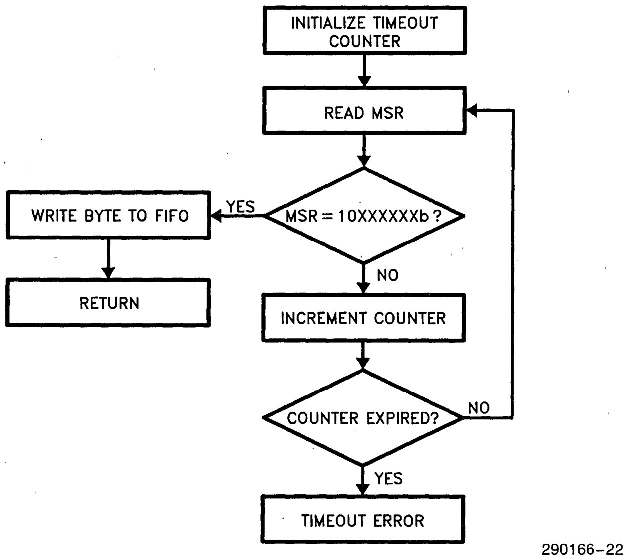


Figure 8-1. Send_Byte Routine

The routine loops until RQM is 1 and DIO is 0 indicating a ready status and FIFO direction is inward. If this condition is true, the 82077 is ready to accept a command or parameter byte. A timeout counter is used to insure software response within a reasonable amount of time in case of no response by the 82077. As a note, the programmer must be careful how the maximum delay is chosen to avoid unnecessary timeouts. For example, if a new command is issued when the 82077 is in the middle of a polling routine, the MSR will not indicate a ready status for the next parameter byte until the polling sequence completes the loop. This could cause a delay between the first and second bytes of up to 250 μs (@ 250 Kbps). If polling is disabled, this maximum delay is 175 μs. There should also be enough timeout margin to accommodate a shift of the software to a higher speed system. A timeout value that results in satisfactory operation on a 16 MHz CPU might fail when the software is moved to a system with a 25 MHz CPU. A recommended solution is to derive the timeout counter from a system hardware counter that is fixed in frequency from CPU clock to CPU clock.

For reading result bytes from the 82077, a similar routine is used. Figure 8-2 illustrates the flowchart for the routine "Get_byte". The MSR is polled until RQM is 1 and DIO is 1, which indicates a ready status and outward FIFO direction. At this point, the host can read a byte from the FIFO. As in the Send_byte routine, a timout counter should be incorporated in case of a disk controller lock-up condition. For example, if a disk was not inserted into the disk drive at the time of a read operation, the controller would fail to receive the index pulse and lock-up since the index pulses are required for termination of the execution phase.

8.2 Initialization

Initializing the 82077 involves setting up the appropriate configuration after a reset. Parameters set by the SPECIFY command are undefined after a reset and will need to be reinitialized. CONFIGURE command parameters default to a known state after a reset but will need to be reinitialized if the system requirements are different from the default settings. The flowchart for the recommended initialization sequence of the 82077 is shown in Figure 8-3.

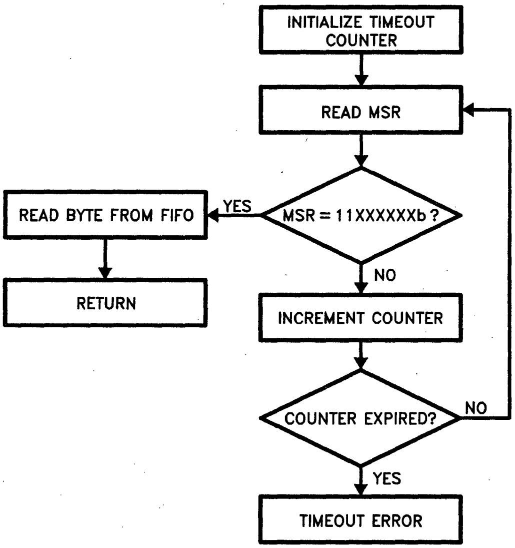


Figure 8-2. Get_Byte Routine

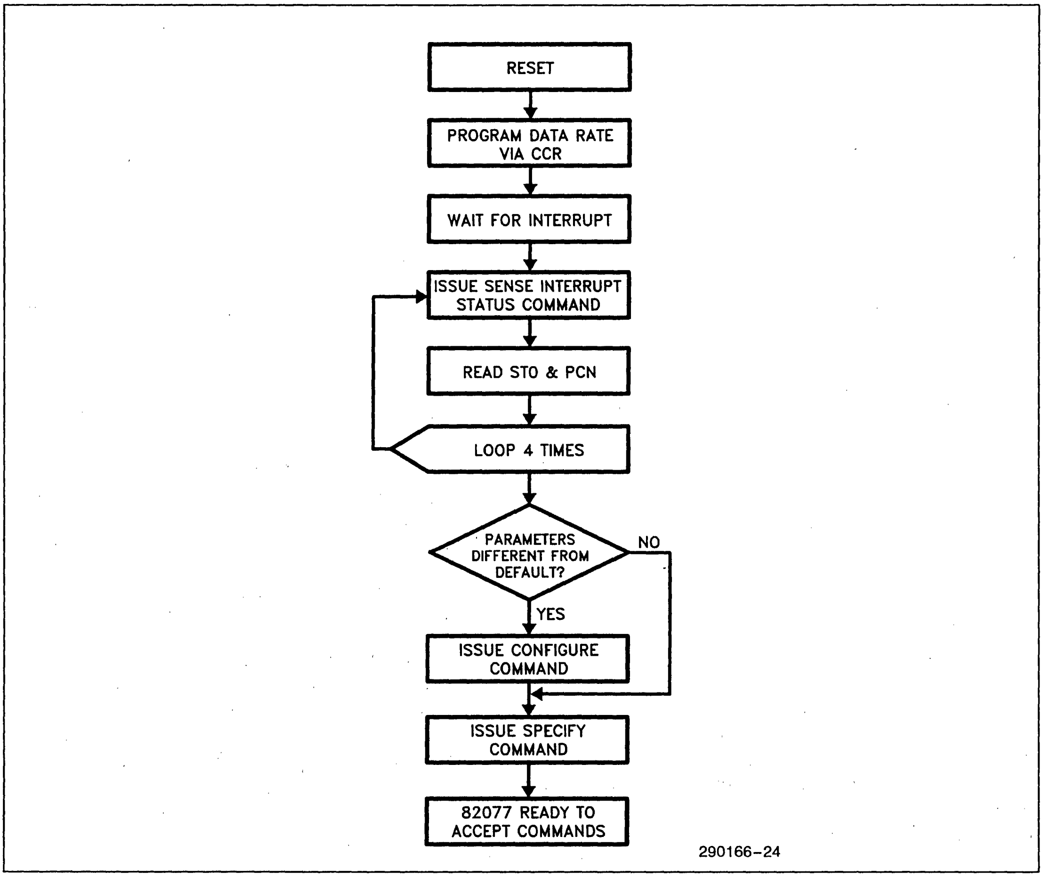

RESET
PROGRAM DATA RATE VIA CCR
WAIT FOR INTERRUPT
ISSUE SENSE INTERRUPT STATUS COMMAND
READ ST0 & PCN
LOOP 4 TIMES
PARAMETERS DIFFERENT FROM DEFAULT?
NO
YES
ISSUE CONFIGURE COMMAND
ISSUE SPECIFY COMMAND
82077 READY TO ACCEPT COMMANDS
290166-24

Figure 8-3. Initialization Flowchart

Following a reset of the 82077, the Configuration Control Register (CCR) should be reinitialized for the appropriate data rate. An external reset via the RESET pin will cause the data rate and write precompensation values to default to 250 Kbps (10b) and 125 ns (000b) respectively. Since the 125 ns write precompensation value is optimal for the 5¼″ and 3½″ disk drive environment, most applications will not require the value to be changed in the initialization sequence. As a note, a software reset issued via the DOR or DSR will not affect the data rate or write precompensation values. But it is recommended as a safe programming practice to always program the data rate after a reset, regardless of the type.

Since polling is enabled after a reset of the 82077, four SENSE INTERRUPT STATUS commands need to be issued afterwards to clear the status flags for each drive. When issuing a SENSE INTERRUPT STATUS command when no active interrupt condition is present, the status register ST0 will return a value of 80H (invalid command). The flowchart in Figure 8-3 illustrates how the software clears each of the four interrupt status flags internally queued by the 82077. It should be noted that although four SENSE INTERRUPT STATUS commands are issued, the INT pin is only active until the first SENSE INTERRUPT STATUS command is executed.

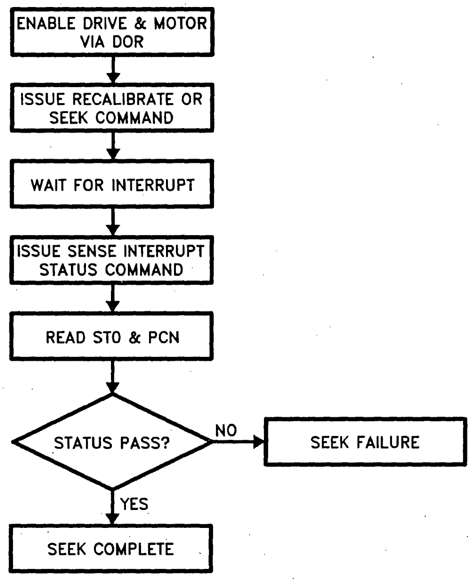
As a note, if the CONFIGURE command is issued within 250 μs of the trailing edge of reset (@ 1 Mbps), the polling mode of the 82077 can be disabled before the polling initiated interrupt occurs. Since polling stops when the 82077 enters the command phase, it is only time critical up to the first byte of the CONFIGURE command. If disabled in time, the system software no longer needs to issue the four SENSE INTERRUPT STATUS commands to clear the internal interrupt flags normally caused by polling.

The CONFIGURE command should also be issued if the system requirements are different from the default settings (as described in Section 5.2.7). For example, the CONFIGURE command can be used to enable the FIFO, set the threshold, and enable Implied Seeks.

The non-DMA mode flag, step rate (SRT), head load (HLT), and head unload times (HUT) programmed by the SPECIFY command do not default to a known state after a reset. This behavior is consistent with the 8272A and has been preserved here for compatibility. Thus, it is necessary to always issue a SPECIFY command in the initialization routine.

8.3 Recalibrates and Seeks

Commands that position the disk head are different from the typical READ/WRITE/FORMAT command in the sense that there is no result phase. Once a RECALIBRATE, SEEK, or RELATIVE SEEK command has been issued, the 82077 will return a ready status in the Main Status Register (MSR) and perform the head positioning operation as a background task. When the seek is complete, the 82077 will assert the INT signal to request service. A SENSE INTERRUPT STATUS command should then be asserted to clear the interrupt and read the status of the operation. Since the drive and motor enable signals are directly controlled through the Digital Output Register (DOR) on the 82077, a write to the DOR will need to precede the RECALIBRATE or SEEK command if the drive and motor is not already enabled. Figure 8-4 shows the flow chart for this operation.


ENABLE DRIVE & MOTOR VIA DOR
ISSUE RECALIBRATE OR SEEK COMMAND
WAIT FOR INTERRUPT
ISSUE SENSE INTERRUPT STATUS COMMAND
READ ST0 & PCN
STATUS PASS?
NO
SEEK FAILURE
YES
SEEK COMPLETE

290166–25

Figure 8-4. Recalibrate and Seek Operations

8.4 Read/Write Data Operations

A read or write data operation requires several steps to complete successfully. The motor needs to be turned on, the head positioned to the correct cylinder, the DMA controller initialized, the read or write command initiated, and an error recovery scheme implemented. The flowchart in Figure 8-5 highlights a recommended algorithm for performing a read or write data operation.

Before data can be transferred to or from the diskette, the disk drive motor must be brought up to speed. For most 3½" disk drives, the spin-up time is 300 ms, while the 5¼" drive usually requires about 500 ms due to the increased moment of inertia associated with the larger diameter diskette.

One technique for minimizing the motor spin-up delay in the read data case is to begin the read operation immediately after the motor is turned on. When the motor is not initially up to speed, the internal data separator will fail to lock onto the incoming data stream and report a failure in the status registers. The read operation is then repeated until successful status is obtained. There is no risk of a data integrity problem since the data field is CRC validated. But, it is not recommended to use this technique for the write data operation even though it requires successful reading of the ID field before the write takes place. The data separator performance of the 82077 is such that locking to the data stream could take place while the motor speed variation is still significant. This could result in errors when an attempt is made to read the disk media by other disk controllers that have a narrower incoming data stream frequency bandwidth.

After the motor has been turned on, the matching data rate for the media inserted into the disk drive should then be programmed to the 82077 via the Configuration Control Register (CCR). The 82077 is designed to allow a different data rate to be programmed arbitrarily without disrupting the integrity of the device. In some applications, it is required to automatically determine the recorded data rate of the inserted media. One technique for doing this is to perform a READ ID operation at each available data rate until a successful status is returned in the result phase.

If implied seeks are not enabled, the disk drive head must be positioned over the correct cylinder by executing a SEEK command. After the seek is complete, a head settling time needs to be asserted before the read or write operation begins. For most drives, this delay should be a minimum of 15 ms. When using implied seeks, the minimum head settling time can be enforced by the head load time (HLT) parameter designated in the SPECIFY command. For example, a HLT value of 8 will yield an effective head settling time of 16 ms for a programmed data rate of 500 Kbps. Of course if the head is already positioned over the correct cylinder, the head settling time does not need to be enforced.

The DMA controller is then initialized for the data transfer and the read or write command is executed. Typically the DMA controller will assert Terminal Count (TC) when the data transfer is complete. The 82077 will then complete the current data transfer and assert the INT signal signifying it has entered the result phase. The result phase can also be entered by the 82077 if an error is encountered or the last sector number equals the End of Track (EOT) parameter.

Based on the algorithm in Figure 8-5, if an error is encountered after reading the result bytes, two more retries are performed by reinitializing the DMA controller and re-issuing the read or write data command. A persisting failure could indicate the seek operation did not achieve proper alignment between the head and the track. The disk head should then be recalibrated and the seek repeated for a maximum of two more tries. Unsuccessful operation after this point should be reported as a disk failure to the operating system.

8.5 Formatting

The disk formatting procedure involves positioning the head on each track and creating a fixed format field used for organizing the data fields. The flowchart in Figure 8-6 highlights the typical format procedure.

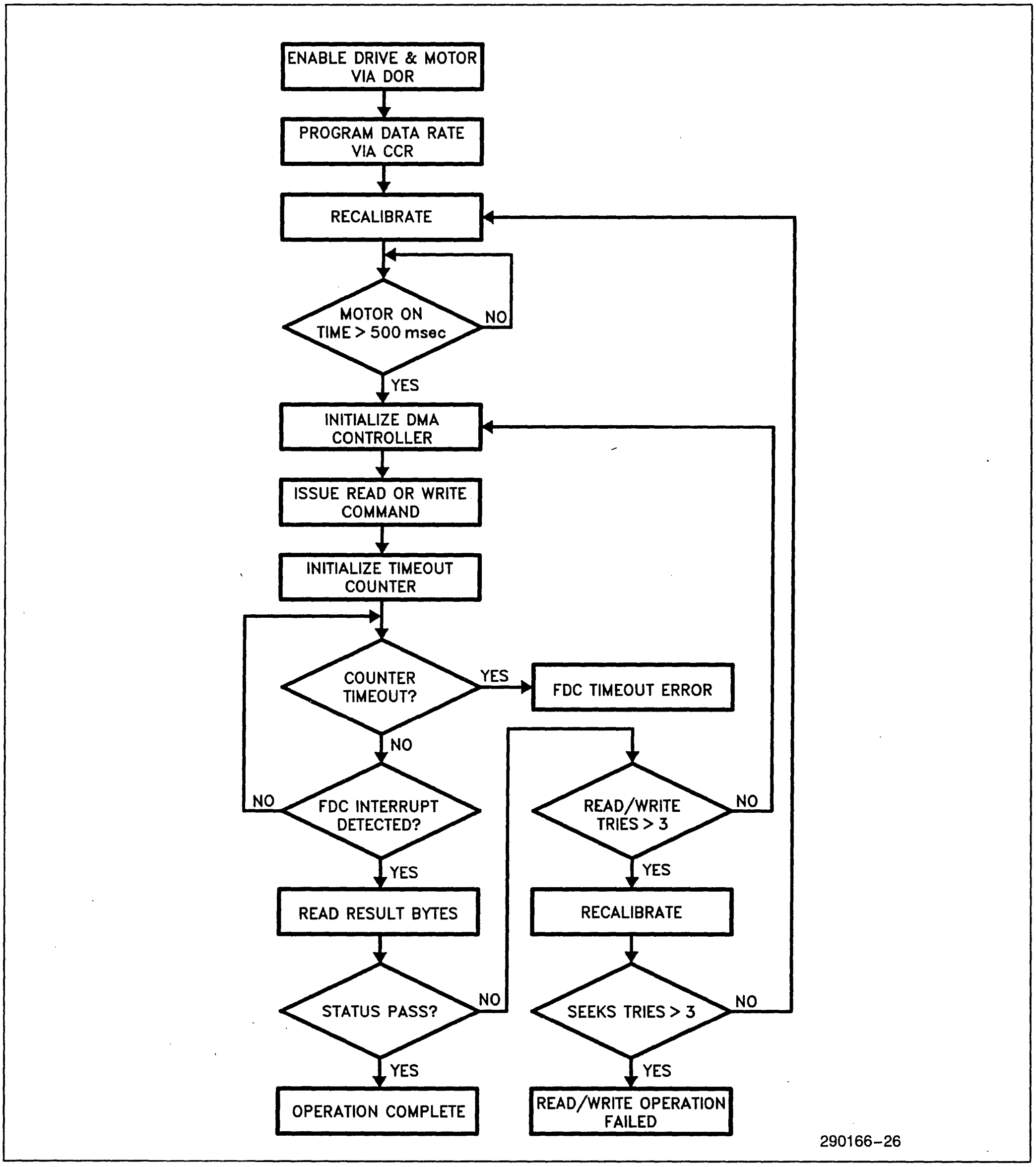


Figure 8-5. Read/Write Operation

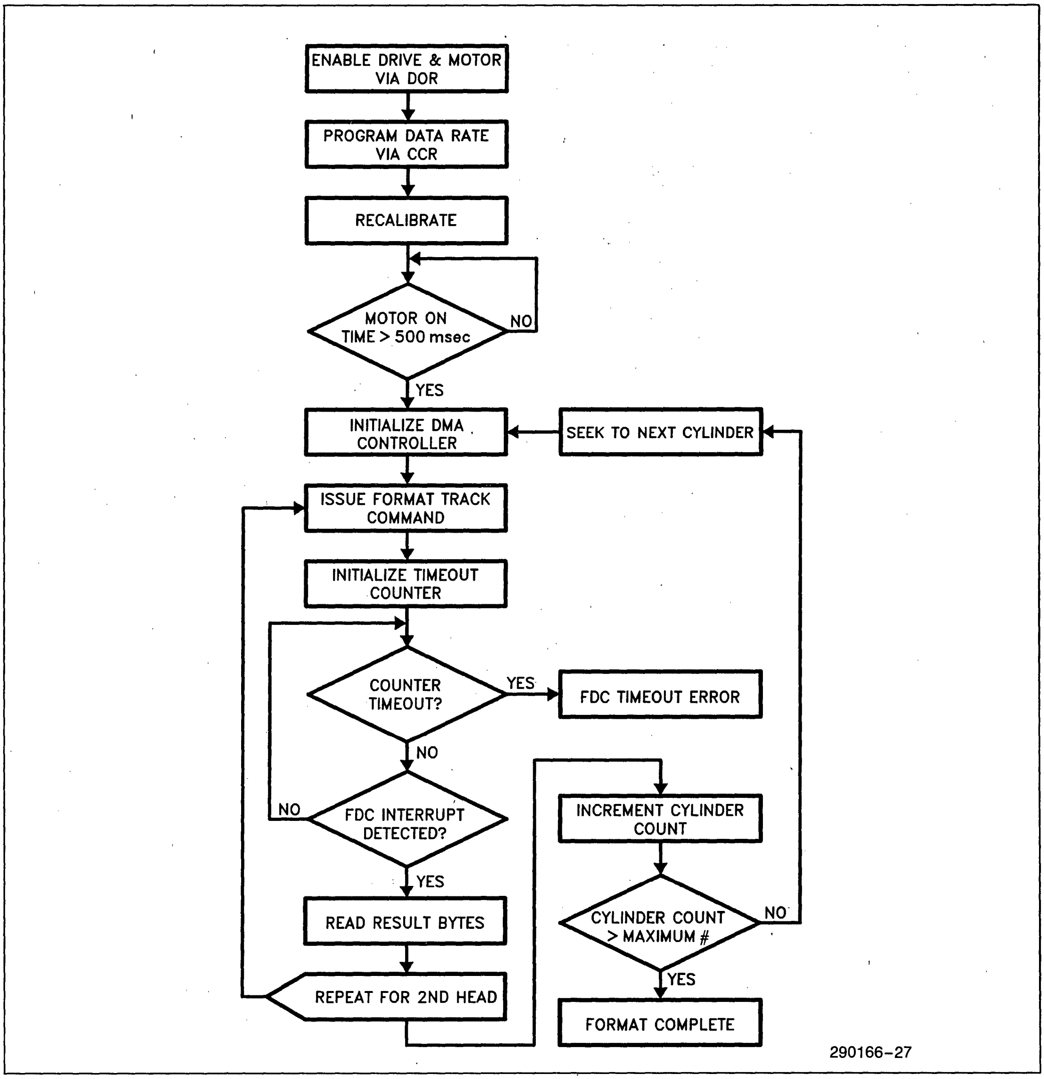


Figure 8-6 Formatting

After the motor has been turned on and the correct data rate programmed, the disk head is recalibrated to track 0. The disk is then allowed to come up to speed via a 500 ms delay. It is important the disk speed has stabilized before the actual formatting to avoid any data rate frequency variations. Since the format fields contain critical information used by the data separator of the disk controller for synchronization purposes, frequency stability of the data stream is imperative for media interchangeability among different systems.

The ID field data created on the disk during the format process is provided by the DMA controller during the execution phase. The DMA controller is initialized to send the C, H, R and N values for each sector ID field. For example, to format cylinder 7, on head 1, with 9 sectors, and a sector size of 2 (512 bytes), the DMA controller should be programmed to transfer 36 bytes (9 sectors x 4 bytes per sector) with the following data field: 7,1,1,2, 7,1,2,2, 7,1,3,2, . . . 7,1,9,2. Since the values provided to the 82077 during the execution phase of the format command are directly recorded as the ID fields on the disk, the data contents can be arbitrary. Some forms of copy protection have been implemented by taking advantage of this capability.

After each head for a cylinder has been formatted, a seek operation to the next cylinder is performed and the format process is repeated. Since the FORMAT TRACK command does not have implied seek capability, the SEEK command must be used. Also, as discussed in Section 8-2, the head settling time needs to be adhered to after each seek operation.

8.6 Verifies

In some applications, the sector data needs to be verified immediately after each write operation. The verify technique historically used with the 8272A or 82072 disk controller involved reinitializing the DMA controller to perform a read transfer or verify transfer (DACK# is asserted but not RD#) immediately after each write operation. A read command is then to be issued to the disk controller and the resulting status indicates if the CRC validated the previously written data. This technique has the drawback of requiring additional software intervention by having to reprogram the DMA controller between each sector write operation. The 82077 supports this older verify technique, but also provides a new VERIFY command that does not require the use of the DMA controller.

To verify a write data transfer or format track operation using the VERIFY command, the software simply issues the command with the same format as a READ DATA command but without the support of the DMA controller. The 82077 will then perform a disk read operation without a host data transfer. The CRC will be calculated for each sector read and compared against the value stored on the disk. When the VERIFY command is complete, the status register will report any detected CRC errors. If EOT is programmed for the final sector number of the track, and the multi-track (MT) bit is set, then both heads can be checked with one VERIFY command.

9.0 DESIGN APPLICATIONS

9.1 PC/AT Floppy Disk Controller

This section presents a design application of a PC/AT compatible floppy disk controller. With an 82077, a 24 MHz crystal, a resistor package, and a device chip select, a complete floppy disk controller can be built. The 82077 integrates all the necessary building blocks for a reliable and low cost solution. But before we discuss the design application using the 82077, it is helpful to describe the architecture of the original IBM PC/AT floppy disk controller design that uses the 8272A.

9.1.1 PC/AT FLOPPY DISK CONTROLLER ARCHITECTURE

The standard IBM PC/AT floppy disk controller using the 8272A requires 34 devices for a complete solution. The block diagram in Figure 9-1 illustrates the complexity of the disk controller. A major portion of this logic involves the design of the data separator. The reliability of the disk controller is primarily dictated by the performance and stability of the data separator. Discrete board level analog phase lock loops generally offer good bit jitter margins but suffer from instability and tuning problems in the manufacturing stage if not carefully designed. While digital data separator designs offer stability and generally a lower chip count, they suffer from poor performance in the recovery of data.

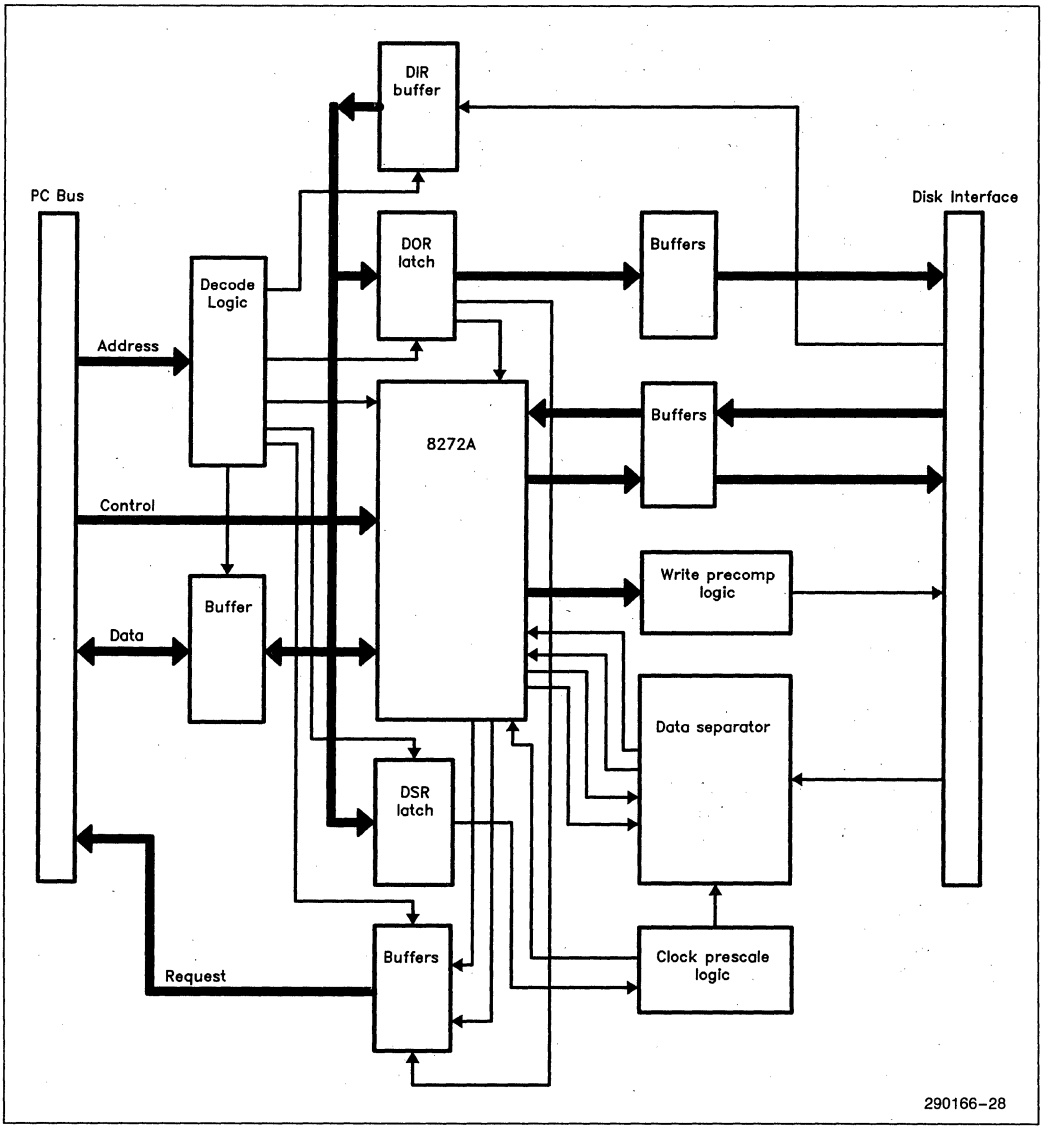


Figure 9-1. Standard IBM PC/AT Floppy Disk Controller

Table 9-1 indicates the drive and media types the IBM PC/AT disk controller can support. This requires the data separator to operate at three different data rates: 250 Kbps, 300 Kbps and 500 Kbps. Clocks to the data separator and disk controller need to be prescaled correspondingly to accommodate each of these data rates. The clock prescaling is controlled by the Data rate Select Register (DSR). Supporting all three data rates can compromise the performance of the phase lock loop (PLL) if steps are not taken in the design to adjust the performance parameters of the PLL with the data rate.

Table 9-1. Standard PC/AT Drives and Media Formats

Capacity	Drive Speed	Data Rate	Sectors	Cylinders
360 Kbyte	300 RPM	250 Kbps	9	40
*360 Kbyte	360 RPM	300 Kbps	9	40
1.2 Mbyte	360 RPM	500 Kbps	15	80

*360 Kbyte diskette in a 1.2 Mbyte drive.

The PC/AT disk controller provides direct control of the drive selects and motors via the Digital Output Register (DOR). As a result, drive selects on the 8272A are not utilized. This places drive selection and motor speed-up control responsibility with the software. The DOR is also used to perform a software reset of the disk controller and tristate the DRQ2 and IRQ6 output signals on the PC bus.

The design of the disk controller also requires address decode logic for the disk controller and register set, buffering for both the disk interface and PC bus, support for write precompensation and monitoring of the disk change signal via a separate read only register (DIR). An I/O address map of the complete register set for the PC/AT floppy disk controller is shown in Table 9-2.

Table 9-2. I/O Address Map for the PC/AT

I/O Address	Access Type	Description
3F0H	—	Unused
3F1H	—	Unused
3F2H	Write	Digital Output Register
3F3H	—	Unused
3F4H	Read	Main Status Register
3F5H	Read/Write	Data Register
3F6H	—	Unused
3F7H	Write	Data Rate Select Register
3F7H	Read	Digital Input Register

9.1.2 82077 PC/AT SOLUTION

The 82077 integrates the entire PC/AT controller design with the exception of the address decode on a single chip. The schematic for this solution is shown in Figure 9-2. The chip select for the 82077 is generated by a 16L8 PAL that is programmed to decode addresses 03F0H thru 03F7H when AEN (Address Enable) is low. The programming equation for the PAL is shown in a ABEL file format in Figure 9-3. An alternative address decode solution could be provided by using a 74LS133 13 input NAND gate and 74LS04 inverter to decode A3–A14 and AEN. Although the PC/AT allows for a 64K I/O address space, decoding down to a 32K I/O address space is sufficient with the existing base of add-in cards.

A direct connection between the disk interface and the 82077 is provided by on-chip output buffers with a 40 mA sink capability. Open collector outputs from the disk drive are terminated at the disk controller with a 150Ω resistor pack. The 82077 disk interface inputs contain a schmitt trigger input structure for higher noise immunity. The host interface is a similar direct connection with 12 mA sink capabilities on DB0–DB7, INT and DRQ.

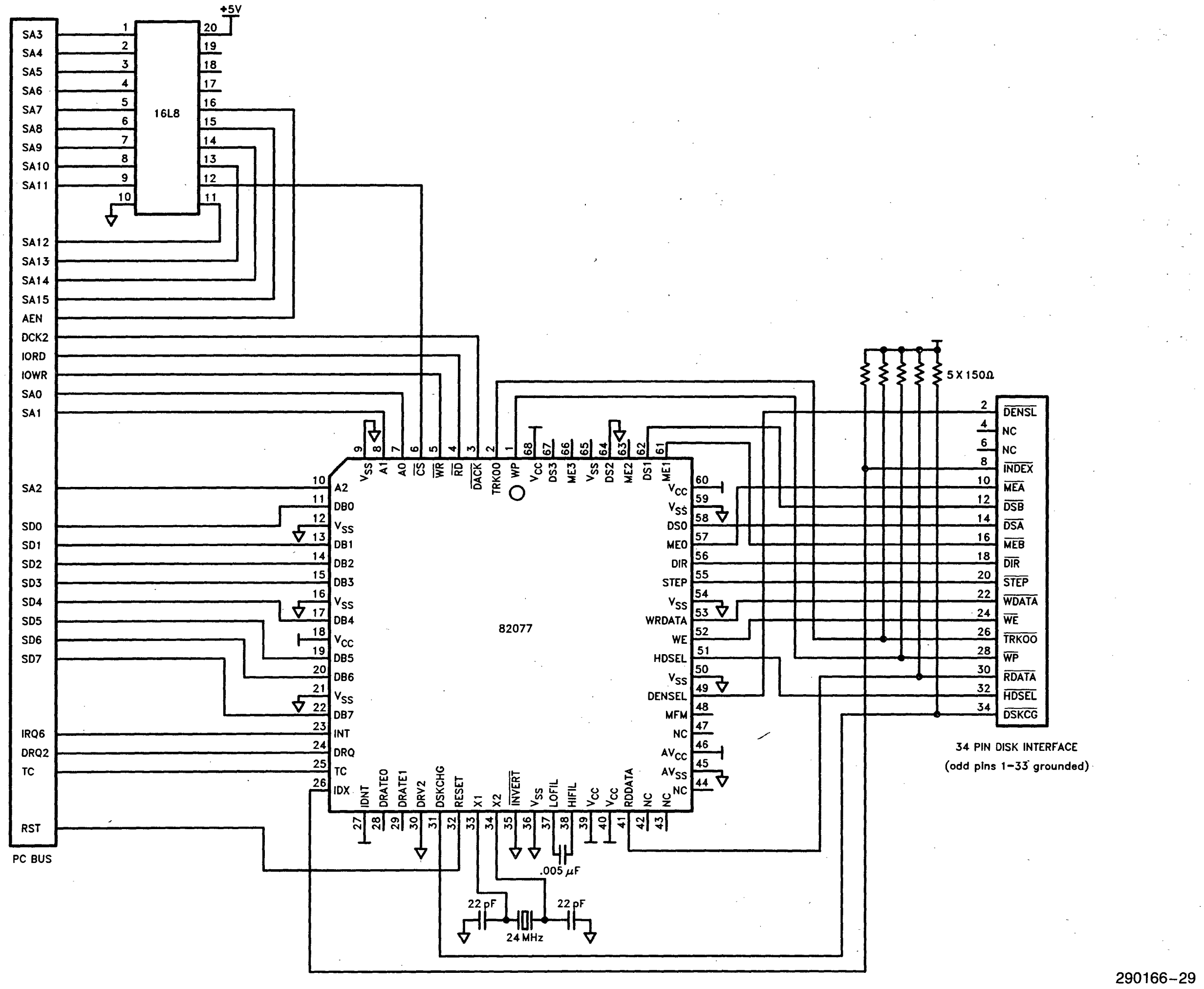


Figure 9-2. 82077 PC/AT Floppy Disk Controller

```
MODULE PCAT077_LOGIC;

TITLE "82077 PC/AT FLOPPY DISK CONTROLLER';
PCAT077 DEVICE "P16L8';

GND,VCC                            PIN 10,20;
SA3,SA4,SA5,SA6,SA7,SA8,SA9,SA10   PIN 1,2,3,4,5,6,7,8;
SA11,SA12,SA13,SA14,SA15,AEN       PIN 9,11,13,14,15,16;
CS077_                             PIN 12;

EQUATIONS

"" CHIP SELECT FOR THE 82077 (3F0H -- 3F7H)

CS077_ = !( !SA15 & !SA14 & !SA13 & !SA12 & !SA11 & !SA10 &
          SA9 & SA8 & SA7 & SA6 & SA5 & SA4 & !SA3 & !AEN) ;

END PCAT077_LOGIC
```

Figure 9-3. PAL Equation File for a PC/AT Compatible FDC Board

9.2 3.5″ Drive Interfacing

The 82077 is designed to interface to both 3.5″ and 5.25″ disk drives. This is facilitated by the 82077 by orienting the "PS/2 mode" for 3.5″ drives and the "AT mode" for 5.25″ drives. But at the same time, this does not inhibit the use of the two types of drives in the opposite mode.

9.2.1 3.5″ DRIVES UNDER THE AT MODE

If it is intended to interface the floppy disk controller with a 3.5″ disk drive in a PC/AT application, then it is likely that two design changes will need to be implemented for the design discussed in Section 9.1. Most 3.5″ disk drives incorporate a totem pole interface structure as opposed to open collector. Outputs of the disk drive will drive both high or low voltage levels when the drive is selected, and float only when the drive has been deselected. These totem pole outputs generally can only sink or source 4 mA of current. As a result, it is recommended to replace the 150Ω termination resistor pack with a 4.7 KΩ package to pull floating signals inactive. Some other 3.5″ drives do have an open collector interface, but have limited sink capability. In these cases, the drive manufacturer manuals usually suggest a 1 KΩ termination.

The second change required under "AT mode" operation involves high capacity 3.5″ disk drives that utilize a density select signal to switch between media recorded at a 250 Kbps and 500 Kbps data rate. The polarity of this signal is inverted for 3.5″ drives versus 5.25″ drives. Thus, an inverter will need to be added between the DENSEL output of the 82077 and the disk drive interface connector when using 3.5″ drives. But drives that do not support both data rates or drives with an automatic density detection feature via an optical sensor do not require the use of the DENSEL signal.

9.2.2 3.5″ DRIVES UNDER THE PS/2 MODE

Interfacing 3.5″ disk drive under the "PS/2 mode" of the 82077 has been simplified. The DENSEL output signal polarity will reflect a 3.5″ drive mode of operation. That is, DENSEL will be high for 250 Kbps or 300 Kbps and low for 500 Kbps or 1 Mbps (assuming INVERT# is low). Thus the only change from the disk interface shown in Figure 9-2 is to replace the 150Ω termination resistor pack with a value of about 10 KΩ. This will prevent excessive current consumption on the CMOS inputs of the 82077 by pulling them inactive when the drive(s) are deselected.

9.2.3 COMBINING 5.25″ AND 3.5″ DRIVES

If 5.25″ and 3.5″ drives are to be combined in a design, then steps need to be taken to avoid contention problems on the disk interface. Since 3.5″ drives do not have a large sink capability, the 150Ω termination resistor pack required by 5.25″ drives cannot be used with the 3.5″ drive. To accommodate both drives with the same disk controller, the outputs of the 3.5″ drive should be buffered before connecting to the 82077 disk interface inputs. The 82077 inputs are then connected to the necessary resistive termination load for the 5.25″ interface.

The block diagram in Figure 9-4 highlights how a combined interface could be designed. In this example, the 5.25″ drive is connected to drive select 0 (DS0) and the 3.5″ drive is connected to drive select 1 (DS1). DS1 is also used to enable a 74LS244 buffer on the output signals of the 3.5″ drive. The drive select logic of the 82077 is mutually exclusive and prevents the activation of the buffer and 5.25″ drive at the same time. Since the 74LS244 has an I_{OL} of 24 mA, the termination resistor should be increased to 220Ω. This could impact the reliability of the 5.25″ drive interface if the cable lengths are greater than 5 feet.

To accommodate the polarity reversal of the DENSEL signal for 3.5″ drives, it is routed through an inverter for the 3.5″ drive interface. A 1 KΩ pull-up should be placed on the output of the inverter to satisfy the I_{OH} requirements for the 3.5″ drive when using a 74LS04.

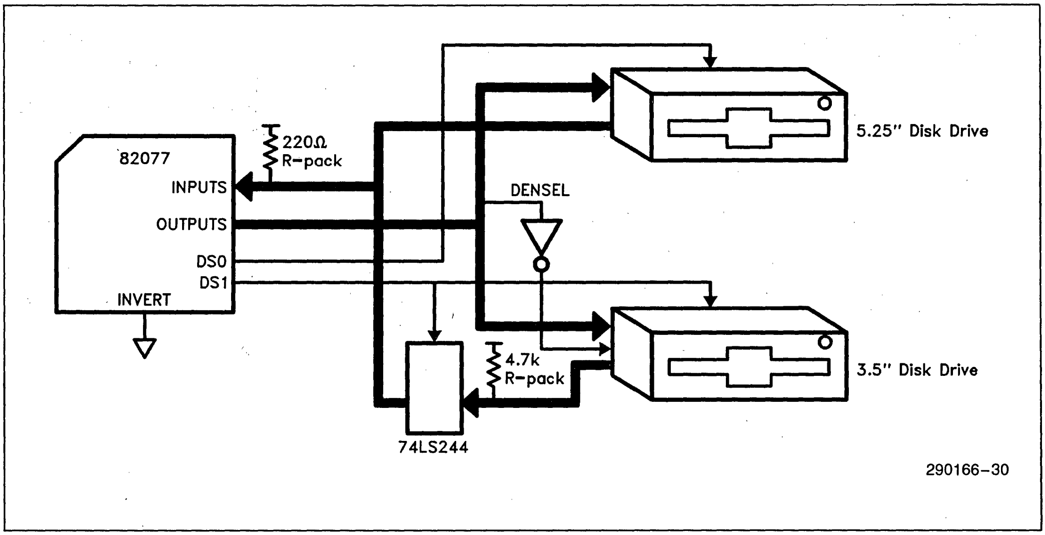


Figure 9-4. Combined 3.5″ and 5.25″ Drive Interface

10.0 D.C. SPECIFICATIONS

10.1 Absolute Maximum Ratings

Storage Temperature .... −65°C to +150°C
Supply Voltage .... −0.5 to +8.0V
Voltage on Any Input .... GND − 2V to 6.5V
Voltage on Any Output . . GND − 0.5V to VCC+0.5V
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.*

NOTICE: Specifications contained within the following tables are subject to change.

10.2 D.C. Characteristics

T_A = 0°C to = 70°C, V_{CC} = +5V ±10%, V_{SS} = AV_{SS} = 0V

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{ILC}	Input Low Voltage, X1	−0.5	0.8	V	
V_{IHC}	Input High Voltage, X1	3.9	V_{CC} +0.5	V	
V_{IL}	Input Low Voltage (all pins except X1)	−0.5	0.8	V	
V_{IH}	Input High Voltage (all pins except X1)	2.0	V_{CC} +0.5	V	
V_{OL}	Output Low Voltage DRATE0-1 and MFM		0.4	V	I_{OL} = 2.5 mA
	DB0-7, INT and DRQ		0.4	V	I_{OL} = 12 mA
	ME0-3, DS0-3, DIR, STP WRDATA, WE, HDSEL and DENSEL		0.4	V	I_{OL} = 40 mA
V_{OH}	Output High Voltage DRATE0-1 and MFM	3.0		V	I_{OH} = −2.5 mA
	All Other Outputs	3.0		V	I_{OH} = −4.0 mA
	All Outputs	V_{CC} − 0.4		V	I_{OH} = −100 μA
	V_{CC} Supply Current (Total)				
I_{CC1}	1 Mbps Data Rate, $V_{IL} = V_{SS}$, $V_{IH} = V_{CC}$		45	mA	(Notes 1, 2)
I_{CC2}	1 Mbps Data Rate, V_{IL} = 0.45, V_{IH} = 2.4		50	mA	(Notes 1, 2)
I_{CC3}	500 Kbps Data Rate, $V_{IL} = V_{CC}$, $V_{IH} = V_{CC}$		35	mA	(Notes 1, 2)
I_{CC4}	500 Kbps Data Rate, V_{IL} = 0.45, V_{IH} = 2.4		40	mA	(Notes 1, 2)
I_{CCSB}	I_{CC} in Powerdown		1.5	mA	(Note 3) Typical
I_{IL}	Input Load Current (all input pins)		10 −10	μA μA	$V_{IN} = V_{CC}$ V_{IN} = 0V
I_{OFL}	Data Bus Output Float Leakage		±10	μA	0.45 < V_{OUT} < V_{CC}

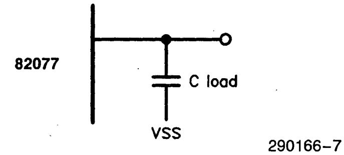
NOTES:
1. The data bus are the only inputs that may be floated.
2. Tested while reading a sync field of "00".
3. $V_{IL} = V_{SS}$, $V_{IH} = V_{CC}$; Outputs not connected to D.C. loads.

Capacitance

C_{IN} C_{IN1} $C_{I/O}$	Input Capacitance Clock Input Capacitance Input/Output Capacitance	10 20 20	pF pF pF	F = 1 MHz, T_A = 25°C Sampled, not 100% Tested

NOTE:
All pins except pins under test are tied to AC ground.

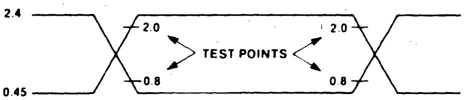
LOAD CIRCUIT



290166-7

C_{load} = 50 pF for all logic outputs, 100 pF for the data bus.

A. C. TESTING INPUT, OUTPUT WAVEFORM



290166-8

11.0 A.C. SPECIFICATIONS

T_A = 0°C to 70°C, V_{CC} = +5V ±10%, V_{SS} = AV_{SS} = 0V

Symbol	Parameter	Min	Max	Unit
CLOCK TIMINGS				
t1	Clock Rise Time		10	ns
	Clock Fall Time		10	ns
t2	Clock High Time(7)	16	26	ns
t3	Clock Low Time(7)	16	26	ns
t4	Clock Period	40	43	ns
t5	Internal Clock Period(3)			
HOST READ CYCLES				
t7	Address Setup to $\overline{RD}$	5		ns
t8	$\overline{RD}$ Pulse Width	100		ns
t9	Address Hold from RD	0		ns
t10	Data Valid from $\overline{RD}$		80	ns
t11	Command Inactive	60		ns
t12	Output Float Delay		35	ns
t13	INT Delay from RD		t5 + 125	ns
t14	Data Hold from $\overline{RD}$	5		ns

A.C. SPECIFICATIONS (Continued)

T_A = 0°C to 70°C, V_{CC} = +5V ±10%, V_{SS} = AV_{SS} = 0V

Symbol	Parameter	Min	Max	Unit
HOST WRITE CYCLES				
t15	Address Setup to $\overline{WR}$	5		ns
t16	$\overline{WR}$ Pulse Width	100		ns
t17	Address Hold from WR	0		ns
t18	Command Inactive	60		ns
t19	Data Setup to WR	70		ns
t20	Data Hold from WR	0		ns
t21	INT Delay from WR		t5 + 125	ns
DMA CYCLES				
t22	DRQ Cycle Period(1)	6.5		µs
t23	DACK to DRQ Inactive(1)		75	ns
t24	RD to DRQ Inactive(4)		100	ns
t25	DACK Setup to $\overline{RD}$, $\overline{WR}$	5		ns
t26	DACK Hold from RD, WR	0		ns
t27	DRQ to $\overline{RD}$, $\overline{WR}$ Active(1)	0	6	µs
t28	Terminal Count Width(10)	50		ns
t29	TC to DRQ Inactive		150	ns
RESET				
t30	Reset Width(5)	170		t4
t31	Reset to Control Inactive		2	µs
WRITE DATA TIMING				
t32	Write Data Width(6)			ns
DRIVE CONTROL				
t35	DIR Setup to STEP	0.5	2	µs
t36	DIR Hold from STEP	10		µs
t37	STEP Active Time (High)	2.5		µs
t38	STEP Cycle Time(2)			µs
t39	INDEX Pulse Width	5		t5

A.C. SPECIFICATIONS (Continued)

T_A = 0°C to 70°C, V_{CC} = +5V ±10%, V_{SS} = AV_{SS} = 0V

Symbol	Parameter	Min	Max	Unit
READ DATA TIMING				
t40	Read Data Pulse Width	50		ns
f44	PLL Data Rate 82077-1		1M	bit/s
	82077		500K	bit/s
t44	Data Rate Period 1/f44			
tLOCK	Lockup Time		64	t44

NOTES:

1. This timing is for FIFO threshold = 1. When FIFO threshold is N bytes, the value should be multiplied by N and subtract 1.5 μs. The value shown is for 1 Mbps, scales linearly with data rate.
2. This value can range from 0.5 ms to 8.0 ms and is dependent upon data rate and the Specify command value.
3. Many timings are a function of the selected data rate. The nominal values for the internal clock period (t5) for the various data rates are:

1 Mbps	3 x oscillator period	= 125 ns
500 Kbps	6 x oscillator period	= 250 ns
300 Kbps	10 x oscillator period	= 420 ns
250 Kbps	12 x oscillator period	= 500 ns

4. If $\overline{DACK}$ transitions before $\overline{RD}$, then this specification is ignored. If there is no transition on $\overline{DACK}$, then this becomes the DRQ inactive delay.
5. Reset requires a stable oscillator to meet the minimum active period.
6. Based on the internal clock period (t_5). For various data rates, the Write Data Width minimum values are:

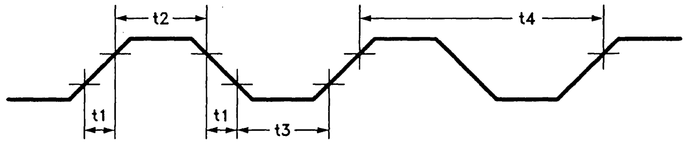
1 Mbps	3 x oscillator period −50 ns	= 150 ns
500 Kbps	6 x oscillator period −50 ns	= 360 ns
300 Kbps	10 x oscillator period −50 ns	= 615 ns
250 Kbps	12 x oscillator period −50 ns	= 740 ns

7. Test points for clock high time are 3.5V. Due to transitional times, clock high time max and clock low time max cannot be met simultaneously. Clock high time min and clock low time max cannot be met simultaneously.
8. Based on internal clock period (t_5).
9. Jitter tolerance is defined as: $\frac{\text{Maximum bit shift from nominal position}}{\text{1/4 period of nominal data rate}} \times 100\%$

It is a measure of the allowable bit jitter that may be present and still be correctly detected. The data separator jitter tolerance is measured under dynamic conditions that jitters the bit stream according to a reverse precompensation algorithm.

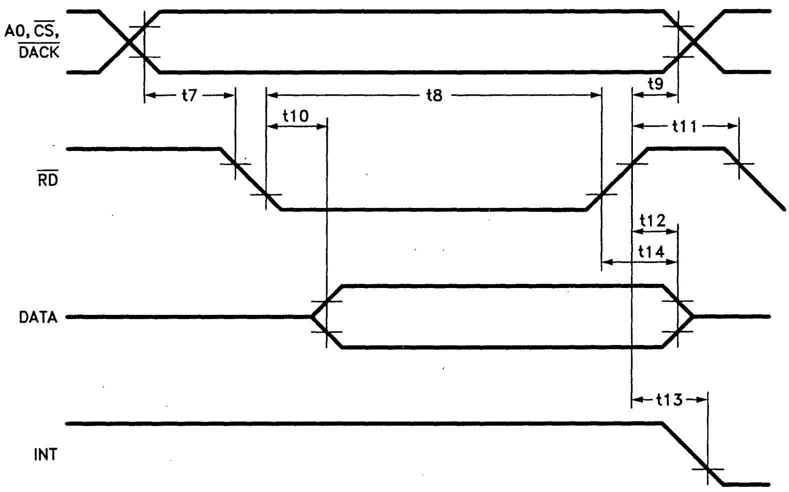
10. TC width is defined as the time that both TC and DACK are active.

CLOCK TIMINGS



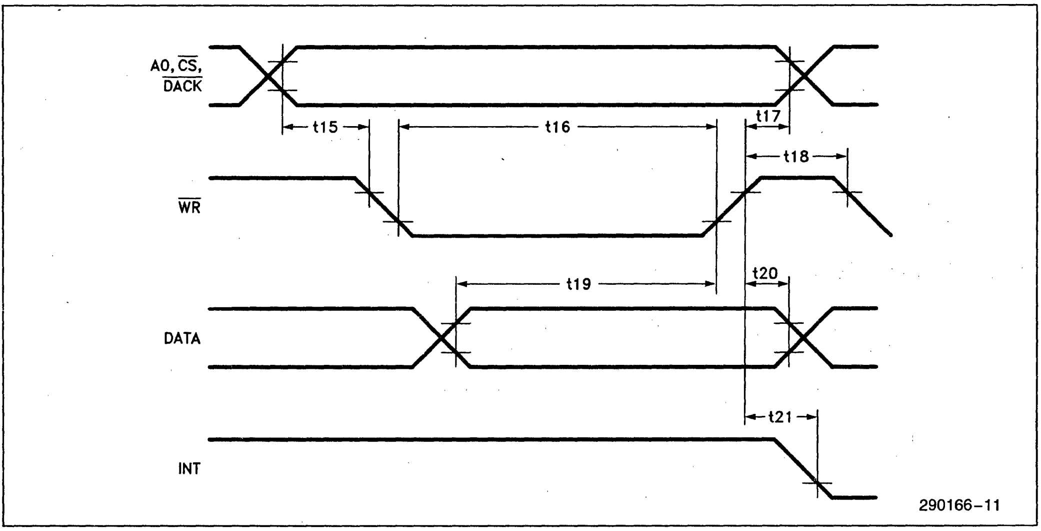
290166-9

HOST READ CYCLES

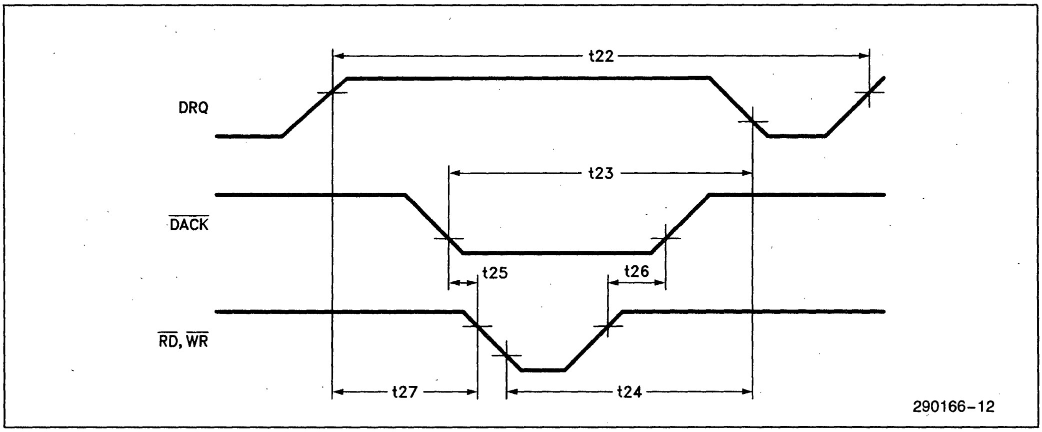


290166-10

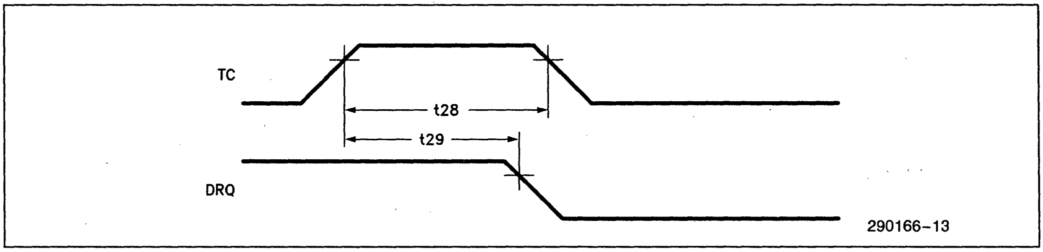
HOST WRITE CYCLES



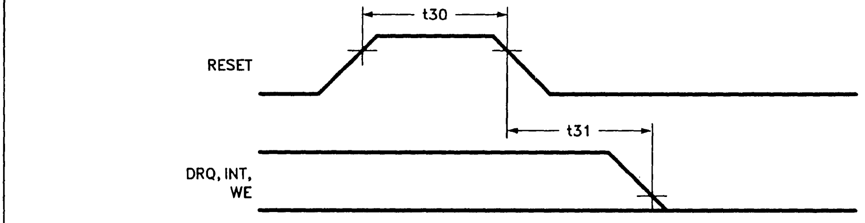
DMA CYCLES



TERMINAL COUNT

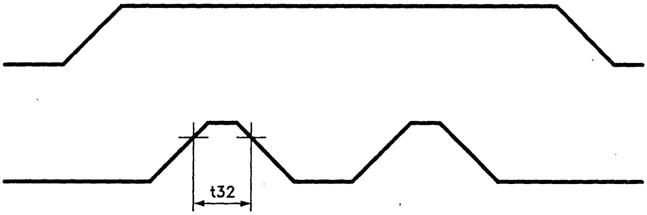


RESET



290166-14

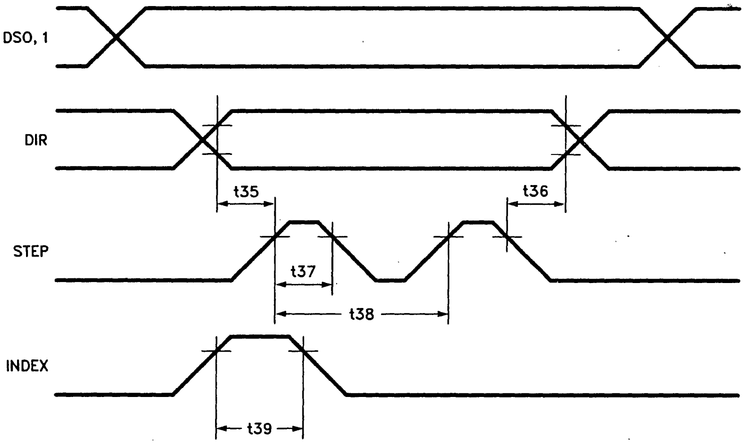
WRITE DATA TIMING



290166-15

NOTE:
Invert high.

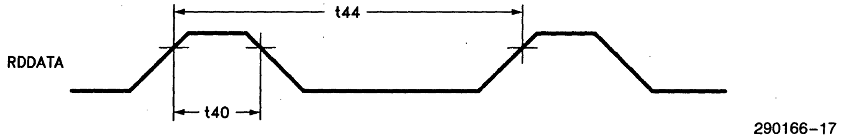
DRIVE CONTROL



290166-16

NOTE:
For overlapped seeks, only one step pulse per drive selection is issued. Non-overlapped seeks will issue all programmed step pulses. Invert high.

INTERNAL PLL



NOTE:
Invert high.

12.0 DATA SEPARATOR CHARACTERISTICS

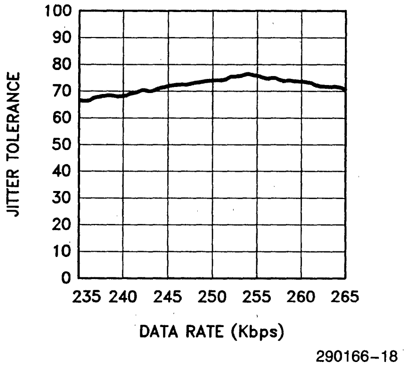


Figure 12-1. Typical Jitter Tolerance vs Data Rate (F_C = 250 Kbps)

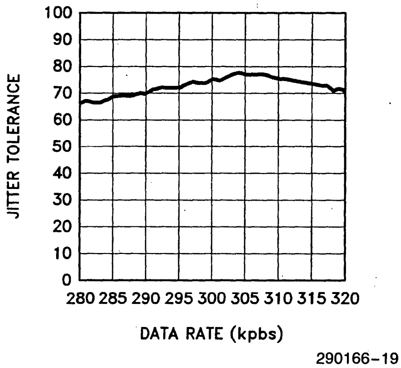


Figure 12-2. Typical Jitter Tolerance vs Data Rate (F_C = 300 Kbps)

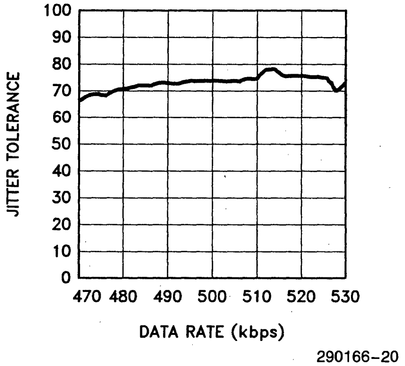


Figure 12-3. Typical Jitter Tolerance vs Data Rate (F_C = 500 Kbps)

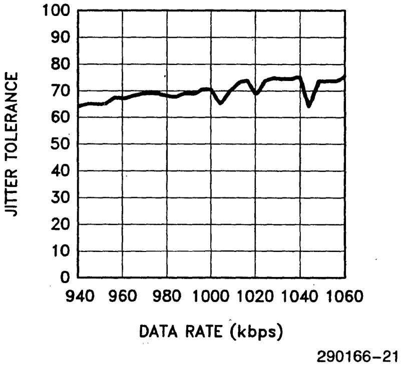


Figure 12-4. Typical Jitter Tolerance vs Data Rate (F_C = 1 Mbps)

REVISION HISTORY

DOCUMENT: 82077 Advanced Information Data Sheet
NEW REVISION NUMBER: 290166-001
OLD REVISION NUMBER: 290166-002

1. On the front page, Vertical Recording support has been added to the feature list.
2. On the front page, Figure 1 has been changed for pins 37 and 38 from NC to LOFIL and HIFIL respectively.
3. In Table 1 under the PLL section, a pin description has been added for HIFIL and LOFIL.
4. After Section 1.1, Section 1.2 has been added to describe the Perpendicular Recording Mode.
5. Under Section 2.1.3, Table 2-1 has been added that lists recommended DOR drive control values.
6. Table 5-1 has been updated to include the Perpendicular Mode command.
7. Under Section 5.1.7.1, Figure 5-3 has been added to describe the Toshiba perpendicular format.
8. Section 5.2.11 has been added to describe the Perpendicular Mode command.
9. Under Section 10.2 of the D.C. Characteristics, the I_{CC} specifications have been reduced in value and the testing descriptions clarified.
10. Under Section 11.0 of the A.C. Specifications, t10 has been reduced from 95 ns to 80 ns and t24 has been reduced from 150 ns to 100 ns.
11. Section 12.0 has been added to describe the data separator characteristics of the internal PLL.

APPLICATION NOTE

AP-116

November 1986

An Intelligent Data Base System Using the 8272

TOM ROSSI
PERIPHERALS APPLICATIONS MANAGER

Order Number: 207875-001

1.0 INTRODUCTION

Most microcomputer systems in use today require low-cost, high-density removable magnetic media for information storage. In the area of removable media, a designer's choice is limited to magnetic types and floppy disks (flexible diskettes), both of which offer non-volatile data storage. The choice between these two technologies is relatively straight-forward for a given application. Since disk drives are designed to permit random access to stored information, they are significantly faster than tape units. For example, locating information on a disk requires less than a second, while tape movement (even at the fastest rewind or fast-forward speed) often requires several minutes. This random access ability permits the use of floppy disks in on-line storage applications (where information must be located, read, and modified/updated in real-time under program or operator control). Tapes, on the other hand, are ideally suited to archival or back-up storage due to their large storage capacities (more than 10 million bytes of data can be archived on a cartridge tape).

A sophisticated controller is required to capitalize on the abilities of the disk storage unit. In the past, disk controller designs have required upwards of 150 ICs. Today, the single-chip 8272 Floppy Disk Controller (FDC) plus approximately 30 support devices can handle up to four million bytes of on-line data storage on four floppy disk drives.

The Floppy Disk

A floppy disk is a circular piece of thin plastic material covered with a magnetic coating and enclosed in a protective jacket (Figure 1). The circular piece of plastic revolves at a fixed speed (approximately 360 rpm) within its jacket in much the same manner that a record revolves at a fixed speed on a stereo turntable. Disks are manufactured in a variety of configurations for various storage capacities. Two standard physical disk sizes are commonly used. The 8-inch disk (8 inches square) is the larger of the two sizes; the smaller size (5¼ inches square) is often referred to as a mini-floppy. Single-sided disks can record information on only one side of the disk, while double-sided disks increase the storage capacity by recording on both sides. In addition, disks are classified as single-density or double-density. Double-density disks use a modified recording method to store twice as much information in the same disk area as can be stored on a single-density disk. Table 1 lists storage capacities for standard floppy disk media.

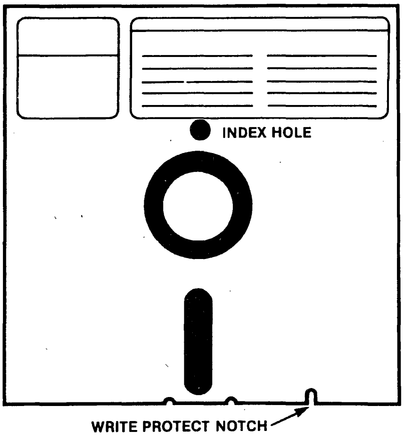


Figure 1. A Floppy Diskette

A magnetic head assembly (in contact with the disk) writes information onto the disk surface and subsequently reads the data back. This head assembly can move from the outside edge of the disk toward the center in fixed increments. Once the head assembly is positioned at one of these fixed positions, the head can read or write information in a circular path as the disk revolves beneath the head assembly. This method divides the surface into a fixed number of cylinders (as shown in Figure 2). There are normally 77 cylinders on a standard disk. Once the head assembly is positioned at a given cylinder, data may be read or written on either side of the disk. The appropriate side of the disk is selected by the read/write head address (zero or one). Of course, a single-sided disk can only use head zero. The combination of cylinder address and head address uniquely specifies a single circular track on the disk. The physical beginning of a track is located by means of a small hole (physical index mark) punched through the plastic near the center of the disk. This hole is optically sensed by the drive on every revolution of the disk.

Table 1. Formatted Disk Capacities

Single-Density Format				
Byte/Sector	128	256	512	1024
Sectors/Track	26	15	8	4
Track/Disk	77	77	77	77
Bytes/Disk	256,256	295,680	315,392	315,392
Double-Density Format				
Byte/Sector	128	256	512	1024
Sectors/Track	52	30	16	8
Track/Disk	77	77	77	77
Bytes/Disk	512,512	591,360	630,784	630,784

Each track is subdivided into a number of sectors (see detailed discussion in section 3). Sectors are generally

128, 256, 512, or 1024 data bytes in length. This track sectoring may be accomplished by one of two techniques: hard sectoring or soft sectoring. Hard sectored disks divide each track into a maximum of 32 sectors. The beginning of each sector is indicated by a sector hole punched in the disk plastic. Soft sectoring, the IBM standard method, allows software selection of sector sizes. With this technique, each data sector is preceded by a unique sector identifier that is read/written by the disk controller.

A floppy disk may also contain a write protect notch punched at the edge of the outer jacket of the disk. This notch is detected by the drive and passed to the controller as a write protect signal.

The Floppy Disk Drive

The floppy disk drive is an electromechanical device that records data on, or reads data from, the surface of a floppy disk. The disk drive contains head control electronics that move the head assembly one increment (step) forward (toward the center of the disk) or backward (toward the edge of the disk). Since the recording head must be in contact with the disk material in order to read or write information, the disk drive also contains head-load electronics. Normally the read/write head is unloaded until it is necessary to read or write information on the floppy disk. Once the head assembly has been positioned over the correct track on the disk, the head is loaded (brought into contact with the disk). This sequence prevents excessive disk wear. A small time penalty is paid when the head is loaded. Approximately thirty to fifty milliseconds are needed before data may be reliably read from, or written to, the disk. This time is known as the head load time. If desired, the head may be moved from cylinder to cylinder while loaded. In this manner, only a small time interval (head settling time) is required before data may be read from the new cylinder. The head settling time is often shorter than the head load time. Typically, disk drives also contain drive select logic that allows more than one physical drive to be connected to the same interface cable (from the controller). By means of a jumper on the drive, the drive number may be selected by the OEM or end user. The drive is enabled only when selected; when not selected, all control signals on the cable are ignored.

Finally, the drive provides additional signals to the system controller regarding the status of the drive and disk. These signals include:

Drive Ready—Signals the system that the drive door is closed and that a floppy disk is inserted into the drive.

Track Zero—Indicates that the head assembly is located over the outermost track of the disk. This signal may be used for calibration of the disk drive at system initialization and after an error condition.

Write Protect—Indicates that the floppy disk loaded into the drive is write protected.

Dual Sided—Indicates that the floppy disk in the drive is dual-sided.

Write Fault—Indicates that an error occurred during a recording operation.

Index—Informs the system that the physical index mark of the floppy disk (signifying the start of a data track) has been sensed.

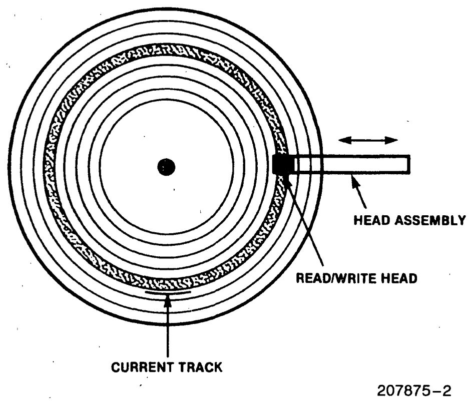


Figure 2. Concentric Cylinders on a Floppy Diskette

2.0 SUBSYSTEM OVERVIEW

A disk subsystem consists of the following functional electronic units:

1) Disk Controller Electronics
2) Disk Drive Electronics
3) Controller/Disk Interface (cables, drivers, terminators)
4) Controller/Microprocessor System Interface

The operation of these functional units is discussed in the following paragraphs.

Controller Electronics

The disk controller is responsible for converting high-level disk commands (normally issued by software executing on the system processor) into disk drive commands. This function includes:

1) Disk Drive Selection—Disk controllers typically manage the operations of multiple floppy disk drives. This controller function permits the system processor to specify which drive is to be used in a particular operation.
2) Track Selection—The controller issues a timed sequence of step pulses to move the head from its current location to the proper disk cylinder from which data is to be read or to which data is to be written. The controller stores the current cylinder number and computes the stepping distance from the current cylinder to the specified cylinder. The controller also manages the head select signal to select the correct side of the floppy disk.
3) Sector Selection—The controller monitors the data on a track until the requested sector is sensed.
4) Head Loading—The disk controller determines the times at which the head assembly is to be brought in contact with the disk surface in order to read or write data. The controller is also responsible for waiting until the head has settled before reading or writing information. Often the controller maintains the head loaded condition for up to 16 disk revolutions (approximately 2 seconds) after a read or write operation has been completed. This feature eliminates the head load time during periods of heavy disk I/O activity.
5) Data Separation—The actual signal recorded on a floppy disk is a combination of timing information (clock) and data. The serial READ DATA input (from the disk drive) must be converted into two signal streams: clock and data. (The READ DATA input operates at 250K bits/second for single-density disks and 500K bits/second for double-density disks.) The serial data must also be assembled into 8-bit bytes for transfer to system memory. A byte must be assembled and transferred every 32 microseconds for single-density disks and every 16 microseconds for double-density.
6) Error-Checking—Information recorded on a floppy disk is subject to both hard and soft errors. Hard (permanent) errors are caused by media defects. Soft errors, on the other hand, are temporary errors caused by electromagnetic noise or mechanical interference. Disk controllers use a standard error checking technique known as a Cyclic Redundancy Check (CRC). As data is written to a disk, a 16-bit CRC character is computed and also stored on the disk. When the data is subsequently read, the CRC character allows the controller to detect data errors. Typically, when CRC errors are detected, the controlling software retries the failed operation (attempting to recover from a soft error). If data cannot reliably be read or written after a number of retries, the system software normally reports the error to the operator. Multiple CRC errors normally indicate unrecoverable media error on the current disk track. Subsequent recovery attempts must be defined by the system deisgners and tailored to meet system interfacing requirements.

Today, single-chip digital LSI floppy disk controllers such as the 8272 perform all the above functions with the exception of data separation. A data separation circuit (a combination of digital and analog electronics) synchronizes itself to the actual data rate of the disk drive. This data rate varies from drive to drive (due to mechanical factors such as motor tolerances) and varies from disk to disk (due to temperature effects). In order to operate reliably with both single- and double-density storage, the data separation circuit must be based on phase-locked loop (PLL) technology. The phase-locked loop data separation logic is described in section 5. The separation logic, after synchronizing with the data stream, supplies a data window to the LSI disk controller. This window differentiates data information from clock information within the serial stream. The controller uses this window to reconstruct the data previously recorded on the floppy disk.

Drive Electronics

Each floppy disk drive contains digital electronic circuits that translate TTL-compatible command signals into electromechanical operations (such as drive selection and head movement/loading) and that sense and report disk or drive status to the controller (e.g., drive ready, write fault, and write protect). In addition, the drive electronics contain analog components to sense, amplify, and shape data pulses read from, or written to, the floppy disk surface by the read/write head.

Controller/Drive Interface

The controller/drive interface consists of high-current line drivers, Schmitt triggered input gates, and flat or twisted pair cable(s) to connect the disk drive electronics to the controller electronics. Each interface signal line is resistively terminated at the end of the cable farthest from the line drivers. Eight-inch drives may be directly interfaced by means of 50-conductor flat cable. Generally, cable lengths should be less than ten feet in order to maintain noise immunity.

Normally, provisions are made for up to four disk drives to share the same interface cable. The controller may operate as many cable assemblies as practical. LSI floppy disk controllers typically operate one to four drives on a single cable.

Processor/Memory Interface

The disk controller must interface to the system processor and memory for two distinct purposes. First, the processor must specify disk control and command parameters to the controller. These parameters include the selection of the recording density and specification of disk formatting information (discussed in section 3). In addition to disk parameter specification, the processor must also send commands (e.g., read, write, seek, and scan) to the controller. These commands require the specification of the command code, drive number, cylinder address, sector address, and head address. Most LSI controllers receive commands and parameters by means of processor I/O instructions.

In addition to this I/O interface, the controller must also be designed for high-speed data transfer between memory and the disk drive. Two implementation methods may be used to coordinate this data transfer. The lower-cost method requires direct processor intervention in the transfer. With this method, the controller issues an interrupt to the processor for each data transfer. (An equivalent method allows the processor to poll an interrupt flag in the controller status word.) In the case of a disk write operation, the processor writes a data byte (to be encoded into the serial output stream) to the disk controller following the receipt of each controller interrupt. During a disk read operation, the processor reads a data byte (previously assembled from the input data stream) from the controller after each interrupt. The processor must transfer a data byte from the controller to memory or transfer a data byte from memory to the disk controller within 16 or 32 microseconds after each interrupt (double-density and single-density response times, respectively).

If the system processor must service a variety of other interrupt sources, this interrupt method may not be practical, especially in double-density systems. In this case, the disk controller may be interfaced to a Direct Memory Access (DMA) controller. When the disk controller requires the transfer of a data byte, it simply activates the DMA request line. The DMA controller interfaces to the processor and, in response to the disk controller's request, gains control of the memory interface for a short period of time—long enough to transfer the requested data byte to/from memory. See section 6 for a detailed DMA interface description.

3.0 DISK FORMAT

New floppy disks must be written with a fixed format by the controller before these disks may be used to store data. Formatting is a method of taking raw media and adding the necessary information to permit the controller to read and write data without error. All formatting is performed by the disk controller on a track-by-track basis under the direction of the system processor. Generally, a track may be formatted at any time. However, since formatting "initializes" a complete disk track, all previously written data is lost (after a format operation). A format operation is normally used only when initializing new floppy disks. Since soft-sectoring in such a predominant formatting technique (due to IBM's influence), the following discussion will limit itself to self-sectored formats.

Data Recording Techniques

Two standard data recording techniques are used to combine clock and data information for storage on a floppy disk. The single-density technique is referred to as FM encoding. In FM encoding (see Figure 3), a double frequency encoding technique is used that inserts a data bit between two adjacent clock bits. (The presence of a data bit represents a binary "one" while the absence of a data bit represents a binary "zero.") The two adjacent clock bits are referred to as a bit cell, and except for a unique field identifiers, all clock bits written on the disk are binary "ones." In FM encoding, each data bit is written at the center of the bit cell and the clock bits are written at the leading edge of the bit cell.

The encoding used for double-density recording is terms MFM encoding (for "Modified FM"). In MFM encoding (Figure 3) the data bits are again written at the center of the bit cell. However, a clock bit is written at the leading edge of the bit cell only if no data bit was written in the previous bit cell and no data bit will be written in the present bit cell.

Sectors

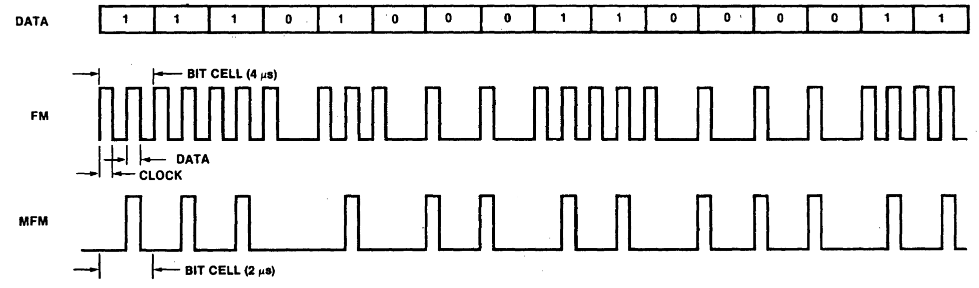
Soft-sectored floppy disks divide each track into a number of data sectors. Typically, sector sizes of 128, 256, 512, or 1024 data bytes are permitted. The sector size is specified when the track initially formatted by the controller. Table 1 lists the single- and double-density data storage capacities for each of the four sector sizes. Each sector within a track is composed of the following four fields (illustrated in Figure 4):

1) Sector ID Field—This field, consisting a seven bytes, is written only when the track is formatted. The ID field provides the sector identification that is used by the controller when a sector must be read or written. The first byte of the field is the ID address mark, a unique coding that specifies the beginning of the ID field. The second, third, and fourth bytes are the cylinder, head, and sector addresses, respectively, and the fifth byte is the sector length code. The last two bytes are the 16-bit CRC character for the ID field. During formatting, the controller supplies the address mark. The cylinder, head, and sector addresses and the sector length code are supplied to the controller by the processor software. The CRC character is derived by the controller from the data in the first five bytes.

2) Post ID Field Gap—The post ID field gap (gap 2) is written initially when the track is formatted. During subsequent write operations, the drive's write circuitry is enabled within the gap and the trailing bytes of the gap are rewritten each time the sector is updated (written). During subsequent read operations, the trailing bytes of the gap are used to synchronize the data separator logic with the upcoming data field.

3) Data Field—The length (number of data bytes) of the data field is determined by software when the track is formatted. The first byte of the data field is the data address mark, a unique coding that specifies the beginning of the data field. When a sector is to be deleted, (e.g., a hard error on the disk), a deleted data address mark is written in place of the data address mark. The last two bytes of the data field comprise the CRC character.

4) Post Data Field Gap—The post data field gap (gap 3) is written when the track is formatted and separates the preceding data field from the next physical ID field on the track. Note that a post data field gap is not written following the last physical sector on a track. The gap itself contains a program-selectable number of bytes. Following a sector update (write) operation, the drive's write logic is disabled during the gap. The actual size of gap 3 is determined by the maximum number of data bits that can be recorded on a track, the number of sectors per track and the total sector size (data plus overhead information). The gap size must be adjusted so that it is large enough to contain the discontinuity generated on the floppy disk when the write current is turned on or off (at the start or completion of a disk write operation) and to contain a synchronization field for the upcoming ID field (of the next sector). On the other hand, the gaps must be small enough so that the total number of data bits required on the track (sectors plus gaps) is less than the maximum number of data bits that can be recorded on the track. The gap



Note:
The FM bit cell is twice the size of the MFM bit cell. Thus, the FM time scale in this figure is 4 μs/bit while the MFM time scale is 2 μs/bit.

Figure 3. FM and MFM Encoding

size must be specified for all read, write, and format operation. The gap size used during disk reads and writes must be smaller than the size used to format the disk to avoid the splice points between contiguous physical sectors. Suggested gap sizes are listed in Table 9.

Tracks

The overall format for a track is illustrated in Figure 4. Each track consists of the following fields:

1) Pre-Index Gap—The pre-index gap (gap 5) is written only when the track is formatted.

2) Index Address Mark—The index address mark consists of a unique code that indicates the beginning of a data track. One index mark is written on each track when the track is formatted.

3) Post Index Gap—The post index gap (gap 1) is used during disk read and write operations to synchronize the data separator logic with the data to be read from the ID field (of the first sector). The post index gap is written only when the disk is formatted.

4) Sectors—The sector information (discussed above) is repeated once for each sector on ths track.

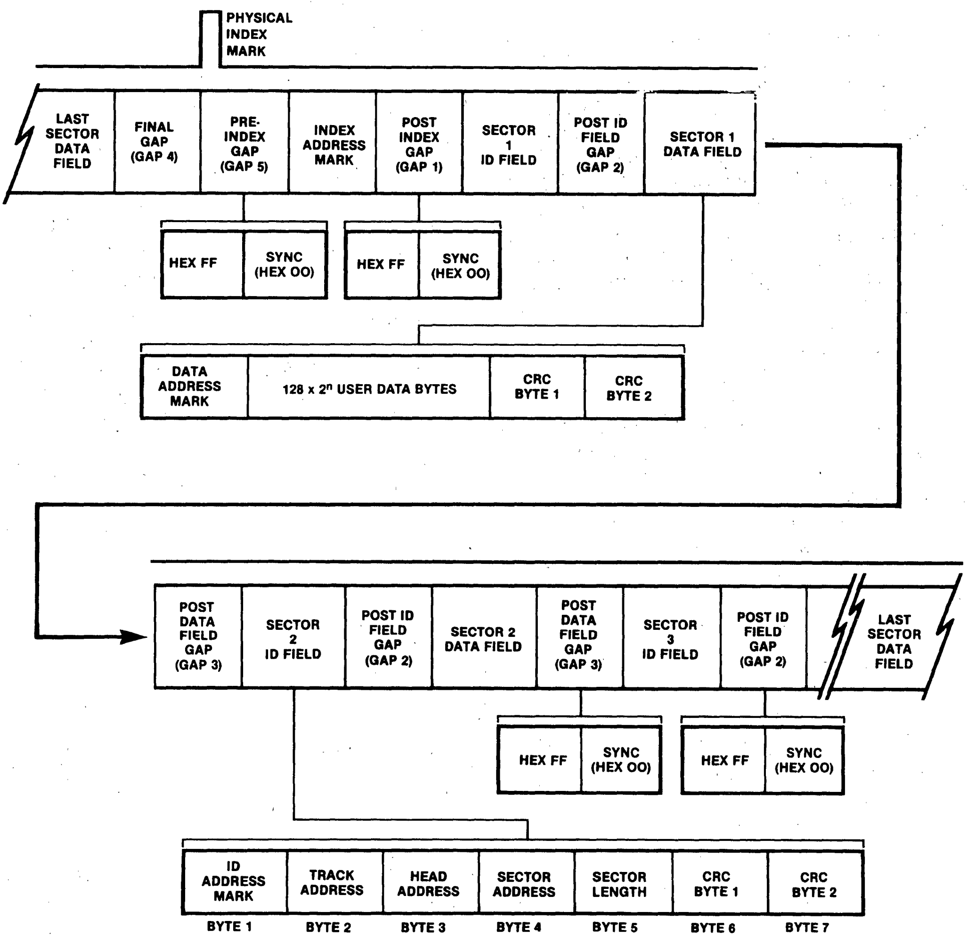


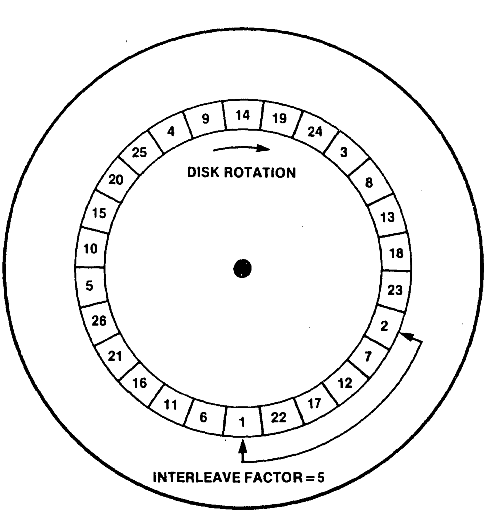
Figure 4. Standard Floppy Diskette Track Format (from SBC 204 Manual)

5) Final Gap—The final gap (gap 4) is written when the track is formatted and extends from the last physical data field on the track to the physical index mark. The length of this gap is dependent on the number of bytes per sector specified, the legnths of the program-selectable gaps specified, and the drive speed.

Sector Interleaving

The initial formatting of a floppy disk determines where sectors are located within a track. It is not necessary to allocate sectors sequentially around the track (i.e., 1,2,3,. . .,26). In fact, it is often advantageous to place the sectors on the track in non-sequential order. Sequential sector ordering optimizes sector access times during multi-sector transfers (e.g., when a program is loaded) by permitting the number of sector specified (up to an entire track) to be transferred within a single revolution of the disk. A technique known as sector interleaving optimizes access times when, although sectors are accessed sequentially, a small amount of processing must be performed between sector reads/writes. For example, an editing program performing a text search reads sectors sequentially, and after each sector is read, performs a software search. If a match is not found, the software issues a read request for the next sector. Since the floppy disk continues to rotate during the time that the software executes, the next physical sector is already passing under the read/write head when the read request is issued, and the processor must wait for another complete revolution of the disk (approximately 166 milliseconds) before the data may actually be input. With interleaving, the sectors are not stored sequentially on a track; rather, each sector is physically removed from the previous sector by some number (known as the interleave factor) of physical sectors as shown in Figure 5. This method of sector allocation provides the processor additional execution time between sectors on the disk. For example, with a 26 sector/track format, an interleave factor of 2 provides 6.4 milliseconds of processing time between sequential 128 byte sector accesses.

To calculate the correct interleave factor, the maximum processor time between sector operations must be divided by the time required for a complete sector to pass under the disk read/write head. After determining the interleave factor, the correct sector numbers are passed to the disk controller (in the exact order that they are to physically appear on the track) during the execution of a format operation.



207875-5

Figure 5. Interleaved Sector Allocation within a Track

4.0 THE 8272 FLEXIBLE DISKETTE CONTROLLER

The 8272 is a single-chip LSI Floppy Disk Controller (FDC) that contains the circuitry necessary to implement both single- and double-density floppy disk storage subsystems (with up to four dual-sided disk drives per FDC). The 8272 supports the IBM 3740 single-density recording format (FM) and the IBM System 34 double-density recording format (MFM). With the 8272, less than 30 ICs are needed to implement a complete disk subsystem. The 8272 accepts and executes high-level disk commands such as format track, seek, read sector, write sector, and read track. All data synchronization and error checking is automatically performed by the FDC to ensure reliable data storage and subsequent retrieval. External logic is required only for the generation of the FDC master clock and write clock (see Section 6) and for data separation (Section 5). The FDC provides signals that control the startup and base frequency selection of the data separator. These signals greatly ease the design of a phase-locked loop data separator.

In addition to the data separator interface signals, the 8272 also provides the necessary signals to interface to

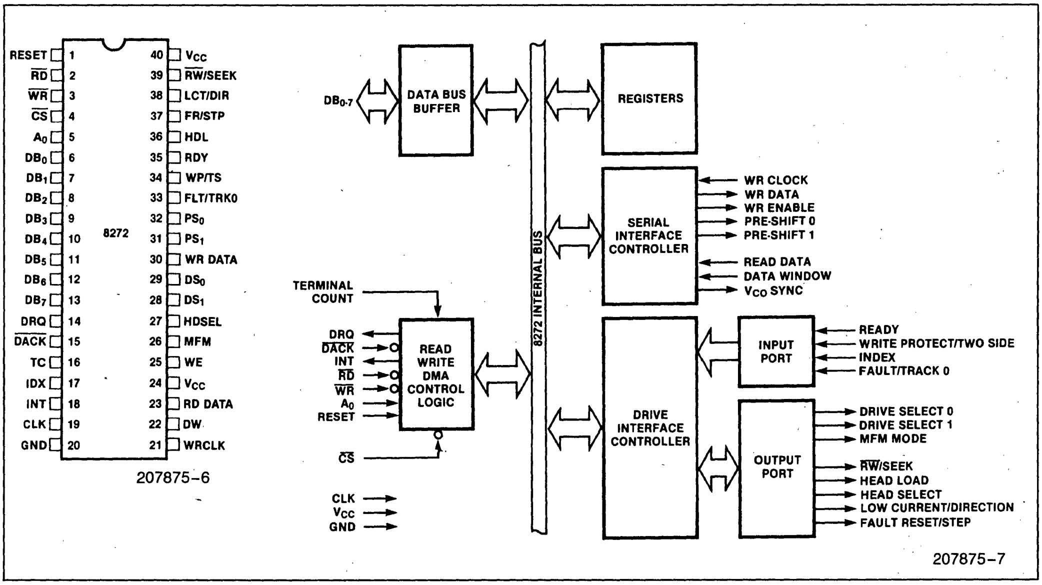


Figure 6. 8272 Pin Configuration and Internal Block Diagram

microprocessor systems with or without Direct Memory Access (DMA) capabilities. In order to interface to a large number of commercially available floppy disk drives, the FDC permits software specification of the track stepping rate, the head load time, and the head unload time.

The pin configuration and internal block diagram of the 8272 is shown in Figure 6. Table 2 contains a description for each FDC interface pin.

Floppy Disk Commands

The 8272 executes fifteen high-level disk interface commands:

Specify	Write Data
Sense Drive Status	Write Deleted Data
Sense Interrupt Status	Read Track
Seek	Read ID
Recalibrate	Scan Equal
Format Track	Scan High or Equal
Read Data	Scan Low or Equal
Read Deleted Data	

Each command is initiated by a multi-byte transfer from the processor to the FDC (the transferred bytes contain command and parameter information). After complete command specification, the FDC automatically executes the command. The command result data (after execution of the command) may require a multi-byte transfer of status information back to the processor. It is convenient to consider each FDC command as consisting of the following three phases:

COMMAND PHASE: The executing program transfers to the FDC all the information required to perform a particular disk operation. The 8272 automatically enters the command phase after RESET and following the completion of the result phase (if any) of a previous command.

EXECUTION PHASE: The FDC performs the operation as instructed. The execution phase is entered immediately after the last command parameter is written to the FDC in the preceding command phase. The execution phase normally ends when the last data byte is transferred to/from the disk (signalled by the TC input to the FDC) or when an error occurs.

RESULT PHASE After completion of the disk operation, status and other housekeeping information are made available to the processor. After the processor reads this information, the FDC re-enters the command phase and is ready to accept another command.

Table 2. 8272 FDC Pin Description

Number	Pin Symbol	I/O	To/From	Description
1	RST	I	μP	**RESET:** Active-high signal that places the FDC in the "idle" state and all disk drive output signals are forced inactive (low). This input must be held active during power on reset while the RD and WR input are active.
2	$\overline{RD}$	I*	μP	**READ:** Active-low control signal that enables data transfer from the FDC to the data bus.
3	$\overline{WR}$	I*	μP	**WRITE:** Active-low control signal that enables data transfer from the data bus into the FDC.
4	$\overline{CS}$	I	μP	**CHIP SELECT:** Active-low control signal that selects the FDC. No reading of writing will occur unless the FDC is selected.
5	A_0	I*	μP	**ADDRESS:** Selects the Data Register or Main Status Register for input/output in conjunction with the RD and WR inputs. (See Table 3.)
6–13	DB_0–DB_7	I/O*	μP	**DATA BUS:** Bidirectional three-state 8-bit data bus.
14	DRQ	O	DMA	**DMA REQUEST:** Active-high output that indicates an FDC request for DMA services.
15	$\overline{DACK}$	1	DMA	**DMA ACKNOWLEDGE:** Active-low control signal indicating that the requested DMA transfer is in progress.
16	TC	I	DMA	**TERMINAL COUNT:** Active-high signal that causes the termination of a command. Normally, the terminal count input is directly connected to the TC/EOP output from the DMA controller, signalling that the DMA transfer has been completed. In a non-DMA environment, the processor must count data transfers and supply a TC signal to the FDC.
17	IDX	I	Drive	**INDEX:** Indicates detection of the physical index mark (the beginning of a track) on the selected disk drive.
18	INT	O	μP	**INTERRUPT REQUEST:** Active-high signal indicating an 8272 interrupt service request.
19	CLK	I		**CLOCK:** Signal phase 8 MHz clock (50% duty cycle).
20	GND			**GROUND:** DC power return.
21	WR CLK	I		**WRITE CLOCK:** 500 kHz (FM) or 1 MHz (MFM) write clock with a constant pulse width of 250 ns (for both FM and MFM recording). The write clock must be present at all times.
22	DW	I	PLL	**DATA WINDOW:** Data sample signal from the phase-locked loop indicating that the FDC should sample input data from the disk drive.
23	RD DATA	I	Drive	**READ DATA:** FDC input data from the selected disk drive.
24	VCO	O	PLL	**VCO SYNC:** Active-high output that enables the phase-locked loop to synchronize with the input data from the disk drive.
25	WE	O	Drive	**WRITE ENABLE:** Active-high output that enables the disk drive write gate.
26	MFM	O	PLL	**MFM MODE:** Active-high output used by external logic to enable the MFM double-density recording mode. When the MFM output is low, single-density FM recording is indicated.
27	HDSEL	O	Drive	**HEAD SELECT:** Selects head 0 or head 1 on a dual-sided disk.

Table 2. 8272 FDC Pin Description (Continued)

Number	Pin Symbol	I/O	To/From	Description
28, 29	DS_1, DS_0	O	Drive	**DRIVE SELECT:** Selects one of four disk drives.
30	WR DATA	O	Drive	**WRITE DATA:** Serial data stream (combination of clock and data bits) to be written on the disk.
31, 32	PS_1, PS_0	O	Drive	**PRECOMPENSATION (PRE-SHIFT) CONTROL:** Write precompensation output control during MFM mode. Specifies early, late, and normal timing signals. See the discussion in Section 5.
33	FLT/TRKO	I	Drive	**FAULT/TRACK 0:** Senses the disk drive fault condition in the Read/Write mode and the Track 0 condition in the Seek mode.
34	WP/TS	I	Drive	**WRITE PROTECT/TWO-SIDED:** Sense the disk write protect status in the Read/Write mode and the dual-sided media status in the Seek mode.
35	RDY	I	Drive	**READY:** Senses the disk drive ready status.
36	HDL	O	Drive	**HEAD LOAD:** Loads the disk drive read/write head. (The head is placed in contact with the disk.)
37	FR/STP	O	Drive	**FAULT RESET/STEP:** Resets the fault flip-flop in the disk drive when operating in the Read/Write mode. Provides head step pulses (to move the head from one cylinder to another cylinder) in the Seek mode.
38	LCT/DIR	O	Drive	**LOW CURRENT/DIRECTION:** Signals that the recording head has been positioned over the inner cylinders (44–77) of the floppy disk in the Read/Write mode. (The write current must be lowered when recording on the physically shorter inner cylinders of the disk. Most drives do not track the actual head position and require that the FDC supply this signal.) Determines the head step direction in the Seek mode. In the Seek mode, a high level on this pin steps the read/write head toward the spindle (step-in); a low level steps the head away from the spindle (step-out).
39	RW/SEEK	O	Drive	**READ, WRITE/SEEK MODE SELECTOR:** A high level selects the Seek mode; a low level selects the Read/Write mode.
40	V_{CC}			+5V DC Power.

*Disabled when CS is high.

Interface Registers

To support information transfer between the FDC and the system processor, the 8272 contains two 8-bit registers: the Main Status Register and the Data Register. The Main Status Register (read only) contains FDC status information and may be accessed at any time. The Main Status Register (Table 4) provides the system processor with the status of each disk drive, the status of the FDC, and the status of the processor interface. The Data Register (read/write) stores data, commands, parameters, and disk drive status information. The Data Register is used to program the FDC during the command phase and to obtain result information after completion of FDC operations. Data is read from, or written to, the FDC registers by the combination of the A0, $\overline{RD}$, $\overline{WR}$, and $\overline{CS}$ signals, as described in Table 3.

In addition to the Main Status Register, the FDC contains four additional status registers (ST0, ST1, ST2, and ST3). These registers are only available during the result phase of a command.

Table 3. FDC Read/Write Interface

$\overline{CS}$	A_0	$\overline{RD}$	$\overline{WR}$	Function
0	0	0	1	Read Main Status Register
0	0	1	0	Illegal
0	0	0	0	Illegal
0	1	0	0	Illegal
0	1	0	1	Read from Data Register
0	1	1	0	Write into Data Register
1	X	X	X	Data Bus is three-stated

Table 4. Main Status Register Bit Definitions

Bit Number	Symbol	Description
0	D_0B	**DISK DRIVE 0 BUSY:** Disk Drive 0 is in the Seek mode.
1	D_1B	**DISK DRIVE 1 BUSY:** Disk Drive 1 is in the Seek mode.
2	D_2B	**DISK DRIVE 2 BUSY:** Disk Drive 2 is in the Seek mode.
3	D_3B	**DISK DRIVE 3 BUSY:** Disk Drive 3 is in the Seek mode.
4	CB	**FDC BUSY:** A read or write command is in process.
5	NDM	**NON-DMA MODE:** The FDC is in the non-DMA mode when this bit is high. This bit is set only during the execution phase of commands in the non-DMA mode. Transition to a low level indicates that the execution phase has ended.
6	DIO	**DATA INPUT/OUTPUT:** Indicates the direction of a data transfer between the FDC and the Data Register. When DIO is high, data is read from the Data Register by the processor; when DIO is low, data is written from the processor to the Data Register.
7	RQM	**REQUEST FOR MASTER:** Indicates that the Data Register is read to send data to, or receive data from, the processor.

Command/Result Phases

Table 5 lists the 8272 command set. For each of the fifteen commands, command and result phase data transfers are listed. A list of abbreviations used in the table is given in Table 6, and the contents of the result status registers (ST0–ST3) are illustrated in Table 7.

The bytes of data which are sent to the 8272 during the command phase, and are read out of the 8272 in the result phase, must occur in the order shown in Table 5. That is, the command code must be sent first and the other bytes sent in the prescribed sequence. All bytes of the command and result phases must be read/written as described. After the last byte of data in the command phase is sent to the 8272 the execution phase automatically starts. In a similar fashion, when the last byte of data is read from the 8272 in the result phase, the command is automatically ended and the 8272 is ready for a new command. A command may be aborted by simply raising the terminal count signal (pin 16). This is a convenient means of ensuring that the processor may always gain control of the 8272 (even if the disk system hangs up in an abnormal manner).

It is important to note that during the result phase all bytes shown in Table 5 must be read. The Read Data command, for example, has seven bytes of data in the result phase. All seven bytes must be read in order to successfully complete the Read Data command. The 8272 will not accept a new command until all seven bytes have been read. The number of command and result bytes varies from command-to-command.

In order to read data from, or write data to, the Data Register during the command and result phases, the system processor must examine the Main Status Register to determine if the Data Register is available. The DIO (bit 6) and RQM (bit 7) flags in the Main Status Register must be low and high, respectively, before each byte of the command word may be written into the 8272. Many of the commands require multiple bytes, and as a result, the Main Status Register must be read prior to each byte transfer to the 8272. To read status bytes during the result phase, DIO and RQM in the Main Status Register must both be high. Note, checking the Main Status Register in this manner before each byte transfer to/from the 8272 is required only in the command and result phases, and is NOT required during the execution phase.

Execution Phase

All data transfers to (or from) the floppy drive occur during the execution phase. The 8272 has two primary modes of operation for data transfers (selected by the specify command):

1. DMA mode
2. non-DMA mode

In the DMA mode, DRQ (DMA Request) is activated for each transfers request. The DMA controller responds to DRQ with $\overline{\text{DACK}}$ (DMA Acknowledge) and $\overline{\text{RD}}$ (for read commands) or $\overline{\text{WR}}$ (for write commands). DRQ is reset by the FDC during the transfer. INT is activated after the last data transfer, indicating the completion of the execution phase, and the beginning of the result phase. In the DMA mode, the terminal count (TC/EOP) output of the DMA controller should be connected to the 8272 TC input to properly terminate disk data transfer commands.

Table 5. 8272 Command Set

PHASE	R/W	DATA BUS								REMARKS
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
READ DATA										
Command	W	MT	MFM	SK	0	0	1	1	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior to Command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and the main-system
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information after command execution
	R	H								
	R	R								
	R	N								
READ DELETED DATA										
Command	W	MT	MFM	SK	0	1	1	0	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior to Command execution
	W	H								
	W	R								
	W	N								
	W	EC 1								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and the main-system
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information after Command execution
	R	H								
	R	R								
	R	N								
WRITE DATA										
Command	W	MT	MFM	0	0	0	1	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior to Command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the main-system and the FDD
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information after Command execution
	R	H								
	R	R								
	R	N								
WRITE DELETED DATA										
Command	W	MT	MFM	0	0	1	0	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior to Command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and the main-system
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information after Command execution
	R	H								
	R	R								
	R	N								

PHASE	R/W	DATA BUS								REMARKS
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
READ A TRACK										
Command	W	0	MFM	SK	0	0	0	1	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior to Command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and the main-system. FDC reads the complete track contents from the physical index mark to EOT
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information after Command execution
	R	H								
	R	R								
	R	N								
READ ID										
Command	W	0	MFM	0	0	1	0	1	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
Execution										The first correct ID information on the track is stored in Data Register
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information during Execution Phase
	R	H								
	R	R								
	R	N								
FORMAT A TRACK										
Command	W	0	MFM	0	0	1	1	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	N								Bytes/Sector
	W	SC								Sectors/Track
	W	GPL								Gap 3
	W	D								Filter Byte
Execution										FDC formats an entire track
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								In this case, the ID information has no meaning
	R	H								
	R	R								
	R	N								
SCAN EQUAL										
Command	W	MT	MFM	SK	1	0	0	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior to Command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	STP								
Execution										Data compared between the FDD and the main-system
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information after Command execution
	R	H								
	R	R								
	R	N								

207875-18

NOTE:
1. A_0 = 1 for all operations.

Table 5. 8272 Command Set (Continued)

PHASE	R/W	DATA BUS D7	D6	D5	D4	D3	D2	D1	D0	REMARKS
SCAN LOW OR EQUAL										
Command	W	MT	MFM	SK	1	1	0	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior Command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	STP								
Execution										Data compared between the FDD and the main-system
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information after Command execution
	R	H								
	R	R								
	R	N								
SCAN HIGH OR EQUAL										
Command	W	MT	MFM	SK	1	1	1	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior Command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	STP								
Execution										Data compared between the FDD and the main-system
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information after Command execution
	R	H								
	R	R								
	R	N								

PHASE	R/W	DATA BUS D7	D6	D5	D4	D3	D2	D1	D0	REMARKS
RECALIBRATE										
Command	W	0	0	0	0	0	1	1	1	Command Codes
	W	0	0	0	0	0	0	DS1	DS0	
Execution										Head retracted to Track 0
SENSE INTERRUPT STATUS										
Command	W	0	0	0	0	1	0	0	0	Command Codes
Result	R	ST 0								Status information at the end of each seek operation about the FDC
	R	C								
SPECIFY										
Command	W	0	0	0	0	0	0	1	1	Command Codes
	W	SPT				HUT				Timer Settings
	W	HLT							ND	
SENSE DRIVE STATUS										
Command	W	0	0	0	0	0	1	0	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
Result	R	ST 3								Status information about the FDD
SEEK										
Command	W	0	0	0	0	1	1	1	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								
Execution										Head is positioned over proper Cylinder on Diskette
INVALID										
Command	W	Invalid Codes								Invalid Command Codes (NoOp — FDC goes into Standby State)
Result	R	ST 0								ST 0 = 80 (16)

207875-19

Table 6. Command/Result Parameter Abbreviations

<table>
<tr><th>Symbol</th><th>Description</th></tr>
<tr><td>C</td><td><b>CYLINDER ADDRESS:</b> The currently selected cylinder address (0 to 76) on the disk.</td></tr>
<tr><td>D</td><td><b>DATA PATTERN:</b> The pattern to be written in each sector data field during formatting.</td></tr>
<tr><td>DS0,DS1</td><td><b>DISK DRIVE SELECT:</b>
<table>
<tr><th>DS1</th><th>DS0</th><th></th></tr>
<tr><td>0</td><td>0</td><td>Drive 0</td></tr>
<tr><td>0</td><td>1</td><td>Drive 1</td></tr>
<tr><td>1</td><td>0</td><td>Drive 2</td></tr>
<tr><td>1</td><td>1</td><td>Drive 3</td></tr>
</table></td></tr>
<tr><td>DTL</td><td><b>SPECIAL SECTOR SIZE:</b> During the execution of disk read/write commands, this parameter is used to temporarily alter the effective disk sector size. By setting N to zero, DTL may be used to specify a sector size from 1 to 256 bytes in length. If the actual sector (on the diskette) is larger than DTL specifies, the remainder of the actual sector is not passed to the system during read commands; during write commands, the remainder of the actual sector is written with all-zeroes bytes. DTL should be set to FF hexadecimal when N is not zero.</td></tr>
<tr><td>EOT</td><td><b>END OF TRACK:</b> The final sector number of the current track.</td></tr>
</table>

Table 6. Command/Result Parameter Abbreviations (Continued)

Symbol	Description
GPL	**GAP LENGTH:** The gap 3 size. (Gap 3 is the space between sectors excluding the VCO synchronization field as defined in section 3.)
H	**HEAD ADDRESS:** Selected head: 0 or 1 (disk side 0 or 1, respectively) as encoded in the sector ID field.
HLT	**HEAD LOAD TIME:** Defines the time interval that the FDC waits after loading the head before initiating a read or write operation. Programmable from 2 to 254 milliseconds (in increments of 2 ms).
HUT	**HEAD UNLOAD TIME:** Defines the time interval from the end of the execution phase (of a read or write command) until the head is unloaded. Programmable from 16 to 240 milliseconds (in increments of 16 ms).
MFM	**MFM/FM MODE SELECTOR:** Selects MFM double-density recording mode when high, FM single-density mode when low.
MT	**MUTLI-TRACK SELECTOR:** When set, this flag selects the multi-track operating mode. In this mode (used only with dual-sided disks), the FDC treats a complete cylinder (under both read/write head 0 and read/write head 1) as a single track. The FDC operates as if this expanded track started at the first sector under head 0 and ended at the last sector under head 1. With this flag set (high), a multi-sector read operation will automatically continue to the first sector under head 1 when FDC finishes operating on the last sector under head 0.
N	**SECTOR SIZE:** The number of data bytes within a sector. (See Table 9.)
ND	**NON-DMA MODE FLAG:** When set (high), this flag indicates that the FDC is to operate in the non-DMA mode. In this mode, the processor is interrupted for each data transfer. When low, the FDC interfaces to a DMA controller by means of the DRQ and DACK signals.
R	**SECTOR ADDRESS:** Specifies the sector number to be read or written. In multi-sector transfers, this parameter specifies the sector number of first sector to be read or written.
SC	**NUMBER OF SECTORS PER TRACK:** Specifies the number of sectors per track to be initialized by the Format Track command.
SK	**SKIP FLAG:** When this flag is set, sectors containing deleted data address marks will automatically be skipped during the execution of multi-sector Read Data or Scan commands. In the same manner, a sector containing a data address mark will automatically be skipped during the execution of a multi-sector Read Delected Data command.
SRT	**STEP RATE INTERVAL:** Defines the time interval between step pulses issued by the FDC (track-to-track access time). Programmable from 1 to 16 milliseconds (in increments of 1 ms).

Table 6. Command/Result Parameter Abbreviations (Continued)

Symbol	Description
ST0,ST3 ST2,ST3	**STATUS REGISTER 0-3:** Registers within the FDC that store status information after a command has been executed. This status information is available to the processor during the Result Phase after command execution. These registers may only be read after a command has been executed (in the exact order shown in Table 5 for each command). These registers should not be confused with the Main Status Register.
STP	**SCAN SECTOR INCREMENT:** During Scan operations, this parameter is added to the current sector number in order to determine the next sector to be scanned.

Table 7. Status Register Definitions

Bit Number	Symbol	Description
Status Register 0		
7,6	IC	**INTERRUPT CODE:** 00— Normal termination of command. The specified command was properly executed and completed without error. 01— Abnormal termination of command. Command execution was started but could not be successfully completed. 10— Invalid command. The requested command could not be executed. 11— Abnormal termination. During command execution, the disk drive ready signal changed state.
5	SE	**SEEK END:** This flag is set (high) when the FDC has completed the Seek command and the read/write head is positioned over the correct cylinder.
4	EC	**EQUIPMENT CHECK ERROR:** This flag is set (high) if a fault signal is received from the disk drive or if the track 0 signal fails to become active after 77 step pulses (Recalibrate command).
3	NR	**NOT READY ERORR:** This flag is set if a read or write command is issued and either the drive is not ready or the command specifies side 1 (head 1) of a single-sided disk.
2	H	**HEAD ADDRESS:** The head address at the time of the interrupt.
1,0	DS1,DS0	**DRIVE SELECT:** The number of the drive selected at the time of the interrupt.
Status Register 1		
7	EN	**END OF TRACK ERROR:** This flag is set if the FDC attempts to access a sector beyond the final sector of the track.
6		Not used. This bit is always low.
5	DE	**DATA ERROR:** Set when the FDC detects a CRC error in either the ID field or the data field of a sector.

Table 7. Status Register Definitions (Continued)

Bit Number	Symbol	Description
Status Register 1 (Continued)		
4	OR	**OVERRUN ERROR:** Set (during data transfers) if the FDC does not receive DMA or processor service within the specified time interval.
3		Not used. This bit is always low.
2	ND	**SECTOR NOT FOUND ERROR:** This flag is set by any of the following conditions. a) The FDC cannot locate the sector specified in the Read Data, Read Delected Data, or Scan command. b) The FDC cannot locate the staring sector specified in the Read Track command. c) The FDC cannot read the ID field without error during a Read ID command.
1	NW	**WRITE PROTECT ERROR:** This flag is set if the FDC detects a write protect signal from the disk drive during the execution of a Write Data, Write Deleted Data, or Format Track command.
0	MA	**MISSING ADDRESS MARK ERROR:** This flag is set by either of the following conditions: a) The FDC cannot detect the ID address mark on the specified track (after two occurrences of the physical index mark). b) The FDC cannot detect the data address mark or deleted data address mark on the specified track. (See also the MD bit of Status Register 2.)
Status Register 2		
7		**NOT USED:** This bit is always low.
6	CM	**CONTROL MARK:** This flag is set when the FDC encounters one of the following conditions: a) A deleted data address mark during the execution of a Read Data or Scan command. b) A data address mark during the execution of a Read Deleted Data command.
5	DD	**DATA ERROR:** Set (high) when the FDC detects a CRC error in a sector data field. This flag is not set when a CRC eror is detected in the ID field.
4	WC	**CYLINDER ADDRESS ERROR:** Set when the cylinder address from the disk sector ID field is different from the current cylinder address maintained within the FDC.
3	SH	**SCAN HIT:** Set during the execution of the Scan command if the scan condition is satisfied.
2	SN	**SCAN NOT SATISFIED:** Set during execution of the Scan command if the FDC cannot locate a sector on the specified cylinder that satisfies the scan condition.
1	BC	**BAD TRACK ERROR:** Set when the cylinder address from the disk sector ID field is FF hexadecimal and this cylinder address is different from the current cylinder address maintained within the FDC. This all "ones" cylinder number indicates a bad track (one containing hard errors) according to the IBM soft-sectored format specifications.
0	MD	**MISSING DATA ADDRESS MARK ERROR:** Set if the FDC cannot detect a data address mark or deleted data address mark on the specified track.

Table 7. Status Register Definitions (Continued)

Bit Number	Symbol	Description
Status Register 3		
7	FT	**FAULT:** This flag indicates the status of the fault signal from the selected disk drive.
6	WP	**WRITE PROTECTED:** This flag indicates the status of the write protect signal from the selected disk drive.
5	RDY	**READY:** This flag indicates the status of the ready signal from the selected disk drive.
4	TO	**TRACK 0:** This flag indicates the status of the track 0 signal from the selected disk drive.
3	TS	**TWO-SIDED:** This flag indicates the status of the two-sided signal from the selected disk drive.
2	H	**HEAD ADDRESS:** This flag indicates the status of the side select signal for the currently selected disk drive.
1,0	DS1,DSO	**DRIVE SELECT:** Indicates the currently selected disk drive number.

In the non-DMA mode, transfers requests are indicated by activation of both the INT output signal and the RQM flag (bit 7) in the Main Status Register. INT can be used for interrupt-driven systems and RQM can be used for polled systems. The system processor must respond to the transfer request by reading data from (activating $\overline{RD}$), or writing data to (activating $\overline{WR}$), the FDC. This response removes the transfer request (INT and RQM are set inactive). After completing the last transfer, the 8272 activates the INT output to indicate the beginning of the result phase. In the non-DMA mode, the processor must activate the TC signal to the FDC (normally by means of an I/O port) after the transfer request for the last data byte has been received (by the processor) and before the appropriate data byte has been read from (or written to) the FDC.

In either mode of operation (DMA or non-DMA), the execution phase ends when a terminal count signal is sensed or when the last sector on a track (the EOT parameter—Table 5) has been read or written. In addition, if the disk drive is in a "not ready" state at the beginning of the execution phase, the "not ready" flag (bit 3 in Status Register 0) is set (high) and the command is terminated.

If a fault signal is received from the disk drive at the end of a write operation (Write Data, Write Deleted Data, or Format), the FDC sets the "equipment check" flag (bit 4 in Status Register 0), and terminates the command after setting the interrupt code (bits 7 and 6 of Status Register 0) to "01" (bit 7 low, bit 6 high).

Multi-Sector and Multi-Track Transfers

During disk read/write transfers (Read Data, Write Data, Read Deleted Data, and Write Deleted Data), the FDC will continue to transfer data from sequential sectors until the TC input is sensed. In the DMA mode, the TC input is normally connected to the TC/EOP (terminal count) output of the DMA controller. In the non-DMA mode, the processor directly controls the FDC TC input as previously described. Once the TC input is received, the FDC stops requesting data transfers (from the system processor or DMA controller). The FDC, however, continues to read data from, or write data to, the floppy disk until the end of the current disk sector. During a disk read operation, the data read from the disk (after reception of the TC input) is discarded, but the data CRC is checked for errors; during a disk write operation, the remainder of the sector is filled with all-zero bytes.

If the TC signal is not received before the last byte of the current sector has been transferred to/from the system, the FDC increments the sector number by one and initiates a read or write command for this new disk sector.

The FDC is also designed to operate in a multi-track mode for dual-sided disks. In the multi-track mode (specified by means of the MT flag in the command byte—Table 5) the FDC will automatically increment the head address (from 0 to 1) when the last sector (on the track under head 0) has been read or written. Reading or writing is then continued on the first sector (sector 1) of head 1.

Drive Status Polling

After the power-on reset, the 8272 automatically enters a drive status polling mode. If a change in drive status is detected (all drives are assumed to be "not ready" at power-on), an interrupt is generated. The 8272 continues this status polling between command executions (and between step pulses in the Seek command). In this manner, the 8272 automatically notifies the system processor when a floppy disk is inserted, removed, or changed by the operator.

Command Details

During the command phase, the Main Status Register must be polled by the CPU before each byte is written into the Data Register. The DI0 (bit 6) and RQM (bit 7) flags in the Main Status Register must be low and high, respectively, before each byte of the command may be written into the 8272. The beginning of the execution phase for any of these commands will cause DI0 to be set high and RQM to be set low.

The following paragraphs describe the fifteen FDC commands in detail.

Specify

The Specify command is used prior to performing any disk operations (including the formatting of a new disk) to define drive/FDC operating characteristics. The Specify command parameters set the values for three internal timers:

1. Head Load Time (HLT)—This seven-bit value defines the time interval that the FDC waits after loading the head before initiating a read or write operation. This timer is programmable from 2 to 254 milliseconds in increments of 2 ms.
2. Head Unload Time (HUT)—This four-bit value defines the time from the end of the execution phase (of a read or write command) until the head is unloaded. This timer is programmable from 16 to 240 milliseconds in increments of 16 ms. If the processor issues another command before the head unloads, the head will remain loaded and the head load wait will be eliminated.
3. Step Rate Time (SRT)—This four-bit value defines the time interval between step pulses issued by the FDC (track-to-track access time). This timer is programmable from 1 to 16 milliseconds in increments of 1 ms.

The time intervals mentioned above are a direct function of the FDC clock (CLK on pin 19). Times indicated above are for an 8 MHz clock.

The Specify command also indicates the choice of DMA or non-DMA operation (by means of the ND bit). When this bit is high the non-DMA mode is selected; when ND is low, the DMA mode is selected.

Sense Drive Status

This command may be used by the processor whenever it wishes to obtain the status of the disk drives. Status Register 3 (returned during the result phase) contains the drive status information as described in Table 7.

Sense Interrupt Status

An interrupt signal is generated by the FDC when one or more of the following events occurs:

1) The FDC enters the result phase for:
   a) Read Data command
   b) Read Track command
   c) Read ID command
   d) Read Deleted Data command
   e) Write Data command
   f) Format Track command
   g) Write Deleted Data command
   h) Scan commands
2) The ready signal from one of the disk drives changes state.
3) A Seek or Recalibrate command completes operation.
4) The FDC requires a data transfer during the execution phase of a command in the non-DMA mode.

Interrupts caused by reasons (1) and (4) above occur during normal command operations and are easily discernible by the processor. However, interrupts caused by reasons (2) and (3) above are uniquely identified with the aid of the Sense Interrupt Status command. This command, when issued, resets the interrupt signal and by means of bits 5, 6, and 7 of Status Register 0 (returned during the result phase) identifies the cause of the interrupt (see Table 8).

Table 8. Interrupt Codes

Seek End Bit 5	Interrupt Code		Cause
	Bit 6	Bit 7	
0	1	1	Read Line changed state, either polarity
1	0	0	Normal Termination of Seek or Recalibrate Command
1	1	0	Abnormal Termination of Seek or Recalibrate Command

Neither the Seek nor the Recalibrate command has a result phase. Therefore, it is mandatory to use the Sense Interrupt Status Command after these commands to effectively terminate them and to provide verification of the disk head position.

When an interrupt is received by the processor, the FDC busy flag (bit 4) and the non-DMA (bit 5) may be used to distinguish the above interrupt causes:

bit 5	bit 4	
0	0	Asynchronous event-(2) or (3) above
0	1	Result phase-(1) above
1	1	Data transfer required-(4) above

A single interrupt request to the processor may, in fact, be caused by more than one of the above events. The processor should continue to issue Sense Interrupt Status commands (and service the resulting conditions) until an invalid command code is received. In this manner, all "hidden" interrupts are serviced.

Seek

The Seek command causes the drive's read/write head to be positioned over the specified cylinder. The FDC determines the difference between the current cylinder address and the desired (specified) address, and issues the appropriate number of step pulses. If the desired cylinder address is larger than the current address, the direction signal (LCT/DIR, pin 38) is set high (step-in); the direction signal is set low (step-out) if the desired cylinder address is less than the current address. No head movement occurs (no step pulses are issued) if the desired cylinder is the same as the current cylinder.

The rate at which step pulses are issued is controlled by the step rate time (SRT) in the Specify command. After each step pulse is issued, the desired cylinder address is compared against the current cylinder address. When the cylinder addresses are equal, the "seek end" flag (bit 5 in Status Register 0) is set (high) and the command is terminated. If the disk drive becomes "not ready" during the seek operation, the "not ready" flag (in Status Register 0) is set (high) and the command is terminated.

During the command phase of the Seek operation the FDC is in the FDC busy state, but during the execution phase it is in the non-busy state. While the FDC is in the non-busy state, another Seek command may be issued.. In this manner parallel seek operations may be in operation on up to four floppy disk drives at once. The Main Status Register contains a flag for each drive (Table 4) that indicates whether the associated drive is currently operating in the seek mode. When a drive has completed a seek operation, the FDC generates an interrupt. In response to this interrupt, the system software must issue a Sense Interrupt Status command. During the result phase of this command, Status Register 0 (containing the drive number in bits 0 and 1) is read by the processor.

Recalibrate

This command causes the read/write head of the disk drive to retract to the track 0 position. The FDC clears the contents of its internal cylinder counter, and checks the status of the track 0 signal from the disk drive. As long as the track 0 signal is low, the direction signal remains high and step pulses are issued. When the track 0 signal goes high, the seek end flag (in Status Register 0) is set (high) and the command is terminated. If the track 0 signal is still low after 77 step pulses have been issued, the seek end and equipment check flags (in Status Register 0) are both set and the Recalibrate command is terminated.

Recalibrate commands for multiple drives can be overlapped in the same manner that Seek commands are overlapped.

Format Track

The Format Track command formats or "initializes" a track on a floppy disk by writing the ID field, gaps, and address marks for each sector. Before issuing the Format command, the Seek command must be used to position the read/write head over the correct cylinder. In addition, a table of ID field values (cylinder, head, and sector addresses and sector length code) must be prepared before the command is executed. During command execution, the FDC accesses the table and, using the values supplied, writes each sector on the track. The ID field address mark originates from the FDC and is written automatically as the first byte of each sector's ID field. The cylinder, head, and sector addresses are taken, in order, from the table. The ID field CRC character (derived from the data written in the first five bytes) is written as the last two bytes of the ID field. Gaps are written automatically by the FDC, with the length of the variable gap determined by one of the Format command parameters.

The data field address mark is generated by the FDC and is written automatically as the first byte of the data field. The data pattern specified in the command phase is written into each data byte of each sector. A CRC character is derived from the data address mark and the data written in the sector's data field. The two CRC bytes are appended to the last data byte.

The formatting of a track begins at the physical index mark. As previously mentioned, the order of sector assignment is taken directly from the formatting table. Four entries are required for each sector: a cylinder address, a head address, a sector address, and a sector length code. The cylinder address in the ID field should be equal to the cylinder address of the track currently being formatted.

The sector addresses must be unique (no two equal). The order of the sector entries in the table is the sequence in which sector numbers appear on the track when it is formatted. The number of entry sets (cylinder, head, and sector address and sector length code) must equal the number of sectors allocated to the track (specified in the command phase).

Since the sector address is supplied, in order, for each sector, tracks can be formatted sequentially (the first sector following the index mark is assigned sector address 1, the adjacent sector is assigned sector address 2, and so on) or sector numbers can be interleaved (see section 3) on a track.

Table 9 lists recommended gap sizes and sectors/track for various sector sizes.

Read Data

Nine (9) bytes are required to complete the command phase specification for the Read Data command. During the execution phase, the FDC loads the head (if it is in the unloaded state), waits the specified head load time (defined in the Specify command), and begins reading ID address marks and ID fields. When the requested sector address compares with the sector address read from the disk, the FDC outputs data (from the data field) byte-by-byte to the system. The Read Data command automatically operates in the multi-sector mode described earlier. In addition, multi-track operation may be specified by means of the MT command flag (Table 5). The amount of data that can be transferred with a single command to the FDC depends on the multi-track flag, the recording density flag, and the number of bytes per sector.

During the execution of read and write commands, the special sector size parameter (DTL) is used to temporarily alter the effective disk sector size. By setting the sector size code (N) to zero, DTL may be used to specify a sector size from 1 to 256 bytes in length. If the actual sector (on the disk) is larger than DTL specifies, only the number of bytes specified by the DTL parameter are passed to the system; the remainder of the actual disk sector is not transferred (although the data is checked for CRC errors). Multi-sector read operations are performed in the same manner as they are when the sector size code is non-zero. (The N and DTL parameters are always present in the command sequence. DTL should be set to FF hexadecimal when N is not zero.)

If the FDC detects the physical index mark twice without finding the requested sector, the FDC sets the "sector not found error" flag (bit 2 in Status Register 1) and terminates the Read Data command. The interrupt code (bits 7 and 6 of Status Register 0) is set to "01." Note that the FDC searches for each sector in a multi-sector operation. Therefore, a "sector not found'" error may occur after successful transfer of one or more preceding sectors. This error could occur if a particular sector number was not included when the track was first formatted or if a hard error on the disk has invalidated a sector ID field.

After reading the ID field and data field in each sector, the FDC checks the CRC bytes. If a read error is detected (incorrect CRC in the ID field), the FDC sets the "data error" flag in Status Register 1; if a CRC error occurs in the data field, the FDC sets the "data error" flag in Status Register 2. In either error condition, the FDC terminates the Read Data command. The interrupt code (bits 7 and 6 in Status Register 0) is set to "01."

If the FDC reads a deleted data address mark from the disk, and the skip flag (specified during the command phase) is not set, the FDC sets the "control mark" flag (bit 6 in Status Register 2) and terminates the Read Data command (after reading all the data in the sector). If the skip flag is set, the FDC skips the sector with the deleted data address mark and reads the next sector. Thus, the skip flag may be used to cause the FDC to ignore deleted data sectors during a multi-sector read operation.

Table 9. Sector Size Relationships

Format	Sector Size	N Sector Size Code	SC Sectors/ Track	GPL(1) Gap 3 Length	GPL(2) Gap 3 Length	Remarks
FM Mode	128 bytes/Sector	00	$1A_{(16)}$	$07_{(16)}$	$1B_{(16)}$	IBM Diskette 1
	256	01	$0F_{(16)}$	$0E_{(16)}$	$2A_{(16)}$	IBM Diskette 2
	512	02	08	$1B_{(16)}$	$3A_{(16)}$	
MFM Mode	256	01	$1A_{(16)}$	$0E_{(16)}$	$36_{(16)}$	iBM Diskette 2D
	512	02	$0F_{(16)}$	$1B_{(16)}$	$54_{(16)}$	
	1024	03	08	$35_{(16)}$	$74_{(16)}$	IBM Diskette 2D

NOTES:

1. Suggested values of GPL in Read or Write commands to avoid splice point between data field and ID field of contiguous sectors.
2. Suggested values of GPL in Format command.

During disk data transfers between the FDC and the system, the FDC must be serviced by the system (processor or DMA controller) every 27 μs in the FM mode, and every 13 μs in the MFM mode. If the FDC is not serviced within this interval, the "overrun error" flag (bit 4 in Status Register 1) is set and the Read Data command is terminated.

If the processor terminates a read (or write) operation in the FDC, the ID information in the result phase is dependent upon the state of the multi-track flag and end of track byte. Table 11 shows the values for C, H, R, and N, when the processor terminates the command.

Write Data

Nine (9) bytes are required to complete the command phase specification for the Write Data command. During the execution phase the FDC loads the head (if it is in the unloaded state), waits the specified head load time (defined by the Specify command), and begins reading sector ID fields. When the requested sector address compares with the sector address read from the disk, the FDC reads data from the processor one byte at a time via the data bus and outputs the data to the data field of that sector. The CRC is computed on this data and two CRC bytes are written at the end of the data field.

The FDC reads the ID field of each sector and checks the CRC bytes. If the FDC detects a read error (incorrect CRC) in one of the ID fields, it sets the "data error" flag (bit 5 in Status Register 1) and terminates the Write Data command. The interrupt code (bits 7 and 6 in Status Register 0) is set to "01."

The Write Data command operates in much the same manner as the Read Data command. The following items are the same; refer to the Read Data command for details:

- Multi-sector and Multi-track operation
- Data transfer capacity
- "End of track error" flag
- "Sector not found error" flag
- "Data error" flag
- Head unload time interval
- ID information when the processor terminates the command (see Table 11)
- Definition of DTL when N = 0 and when N ≠ 0

During the Write Data execution phase, data transfers between the processor and FDC must occur every 31 μs in the FM mode, and every 15 μs in the MFM mode. If the time interval between data transfers is longer than this, the FDC sets the "overrun error" flag (bit 4 in Status Register 1) and terminates the Write Data command.

Read Deleted Data

This command operates in almost the same manner as the Read Data command operates. The only difference involves the treatment of the data address mark and the skip flag. When the FDC detects a data address mark at the beginning of a data field (and the skip flag is not set), the FDC reads all the data in the sector, sets the "control mark" flag (bit 6 in Status Register 2), and terminates the command. If the skip flag is set, the FDC skips the sector with the data address mark and continues reading at the next sector. Thus, the skip flag may be used to cause the FDC to read only deleted data sectors during a multi-sector read operation.

Write Deleted Data

This command operates in the same manner as the Write Data command operates except that a deleted data address mark is written at the beginning of the data field instead of the normal data address mark. This command is used to mark a bad sector (containing a hard error) on the floppy disk.

Read Track

The Read Track command is similar to the Read Data command except that the entire data field is read continuously from each of the sectors of a track. Immediately after encountering the physical index mark, the FDC starts reading all data fields on the track as continuous blocks of data. If the FDC finds an error in the ID field or data field CRC check bytes, it continues to read data from the track. The FDC compares the ID information read from each sector with the values specified during the command phase. If the specified ID field information is not found on the track, the "sector not found error" flag (in Status Register 1) is set. Multitrack and skip operations are not allowed with this command.

This command terminates when the last sector on the track has been read. (The number of sectors on the track is specified by the end of track parameter byte during the command phase.) If the FDC does not find an ID address mark on the disk after it encounters the physical index mark for the second time, it sets the "missing address mark error" flag (bit 0 in Status Register 1) and terminates the command. The interrupt code (bits 7 and 6 of Status Register 0) is set to "01."

Read ID

The Read ID command transfers (reads) the first correct ID field from the current disk track (following the physical index mark) to the processor. If no correct ID address mark is found on the track, the "missing address mark error" flag is set (bit 0 in Status Register 1). If no data mark is found on the track, the "sector not found error" flag is also set (bit 2 in Status Register 1). Either error condition causes the command to be terminated.

Scan Commands

The Scan commands allow the data being read from the disk to be compared against data supplied by the system (by the processor in non-DMA mode, and by the DMA controller in DMA mode). The FDC compares the data on a byte-by-byte basis, and searches for a sector of data that meets the conditions of "disk data equal to system data", "disk data less than or equal to system data", or "disk data greater than or equal to system data". Simple binary (ones complement) arithmetic is used for comparison (FF = largest number, 00 = smallest number). If, after a complete sector of data is compared, the conditions are not met, the sector number is incremented by the scan sector increment (specified in the command phase), and the scan operation is continued. The scan operation continues until one of the following conditions occurs; the conditions for scan are met (equal, low, or high), the last sector on the track is reached, or the terminal count signal is received.

If the conditions for scan are met, the FDC sets the "scan hit" flag (bit 3 in Status Register 2) and terminates the Scan command. If the conditions for scan are not met between the starting sector and the last sector on the track (specified in the command phase), the FDC sets the "scan not satisfied" flag (bit 2 in Status Register 2) and terminates the Scan command. The receipt of a terminal count signal from the processor or DMA controller during the scan operation will cause the FDC to complete the comparison of the particular byte which is in process, and to termiante the command. Table 10 shows the status of the "scan hit" and "scan not satisfied" flags under various scan termination conditions.

If the FDC encounters a deleted data address mark in one of the sectors and the skip flag is low, it regards the sector as the last sector on the cylinder, sets the "control mark" flag (bit 6 in Status Register 2) and terminates the command. If the skip flag is high, the FDC skips the sector with the deleted address mark, and reads the next sector. In this case, the FDC also sets the 'control mark" flag (bit 6 in Stauts Register 2) in order to show that a deleted sector had been encountered.

Table 10. Scan Status Codes

Command	Status Register 2		Comments
	Bit 2 = SN	Bit 3 = SH	
Scan Equal	0	1	$D_{FDD} = D_{Processor}$
	1	0	$D_{FDD} \neq D_{Processor}$
Scan Low or Equal	0	1	$D_{FDD} = D_{Processor}$
	0	0	$D_{FDD} < D_{Processor}$
	1	0	$D_{FDD} \not> D_{Processor}$
Scan High or Equal	0	1	$D_{FDD} = D_{Processor}$
	0	0	$D_{FDD} \not> D_{Processor}$
	1	0	$D_{FDD} \not< D_{Processor}$

NOTE:

During scan command execution, the last sector on the track must be read for the command to terminate properly. For example if the scan sector increment is set to 2, the end of track parameter is set to 26, and the scan begins at sector 21, sectors 21, 23, and 25 will be scanned. The next sector, 27 will not be found on the track and an abnormal command termination will occur. The command would be completed in a normal manner if either a) the scan had started at sector 20 or b) the end of track parameter had been set to 25.

During the Scan command, data is supplied by the processor or DMA controller for comparison against the data read from the disk. In order to avoid having the "overrun error" flag set (bit 4 in Status Register 1), it is necessary to have the data available in less than 27 μs (FM Mode) or 13 μs (MFM Mode). If an overrun error occurs, the FDC terminates the command.

Invalid Commands

If an invalid (undefined) command is sent to the FDC, the FDC will terminate the command. No interrupt is generated by the 8272 during this condition. Bit 6 and bit 7 (DIO and RQM) in the Main Status Register are both set indicating to the processor that the 8272 is in the result phase and the contents of Status Register 0 must be read. When the processor reads Status Register 0 it will find an 80H code indicating that an invalid command was received.

A Sense Interrupt Status command must be sent after a Seek or Recalibrate interrupt; otherwise the FDC will consider the next command to be an invalid command. Also, when the last "hidden" interrupt has been serviced, further Sense Interrupt Status commands will result in invalid command codes.

Table 11. ID Information When Processor Terminates Command

MT	EOT	Final Sector Transferred to Processor	ID Information at Result Phase			
			C	H	R	N
0	1A 0F 08	Sector 1 to 25 at Side 0 Sector 1 to 14 at Side 0 Sector 1 to 7 at Side 0	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 0 Sector 15 at Side 0 Sector 8 at Side 0	C + 1	NC	R = 01	NC
	1A 0F 08	Sector 1 to 25 at Side 1 Sector 1 to 14 at Side 1 Sector 1 to 7 at Side 1	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 1 Sector 15 at Side 1 Sector 8 at Side 1	C + 1	NC	R = 01	NC
1	1A 0F 08	Sector 1 to 25 at Side 0 Sector 1 to 14 at Side 0 Sector 1 to 7 at Side 0	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 0 Sector 15 at Side 0 Sector 8 at Side 0	NC	$\overline{\text{LSB}}$	R = 01	NC
	1A 0F 08	Sector 1 to 25 at Side 1 Sector 1 to 14 at Side 1 Sector 1 to 7 at Side 1	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 1 Sector 15 at Side 1 Sector 8 at Side 1	C + 1	$\overline{\text{LSB}}$	R = 01	NC

NOTES:
1. NC (No Change): The same value as the one at the beginning of command execution.
2. $\overline{\text{LSB}}$ (Least Significant Bit): The least significant bit of H is complemented.

In some applications the user may wish to use this command as a No-Op command to place the FDC in a stand-by or no operation state.

5.0 THE DATA SEPARATOR

As briefly discussed in Section 2, LSI disk controllers such as the 8272 require external circuitry to generate a data window signal. This signal is used within the FDC to isolate the data bits contained within the READ DATA input signal from the disk drive. (The disk READ DATA signal is a composite signal constructed from both clock and data information.) After isolating the data bits from this input signal, the FDC assembles the data bits into 8-bit bytes for transfer to the system processor or memory.

Single Density

In a single-density (FM) recording (Figure 3), the bit cell is 4 microseconds wide. Each bit cell contains a clock bit at the leading edge of the cell. The data bit (if present) is always located at the center of the cell. The job of data separation is relatively straightforward for single-density; simply generate a data window 2 μs wide starting 1 μs after each clock bit. Since every cell has a clock bit, a fixed window reference is available for every data bit and because the window is 2 μs wide, a slightly shifted data bit will still remain within the data window.

A single-density data separator with these specifications may be easily generated using a digital or analog one-shot triggered by the clock bit.

Double-Density

Double-density (MFM) bit cells are reduced to 2 μs (in order to double the disk data storage capacity). Clock bits are inserted into the data stream only if data bits are not present in both the current and preceding bit cells (Figure 3). The data bit (if present) still occurs at the center of the bit cell and the clock bit (if present) still occurs at the leading edge of the bit cell.

MFM data separation has two problems. First, only some bit cells contain a clock bit. In this manner, MFM encoding loses the fixed bit cell reference pulse present in FM encoding. Second, the bit cell for MFM is one-half the size of the bit cell for FM. This shorter bit cell means that MRM cannot tolerate as large a playback data-shift (as FM can tolerate) without errors.

Since most playback data-shift is predictable, the FDC can precompensate the write data stream so that data/clock pulses will be correctly positioned for subsequent playback. This function is completely controlled by the FDC and is only required for MFM recording. During write operations, the FDC specifies an early, normal, or late bit positioning. This timing information is specified with respect to the FDC write clock. Early and late timing is typically 125 ns to 250 ns before or after the write clock transition (depending on disk drive requirements).

The data separator circuitry for double-density recording must continuously analyze the total READ DATA stream, synchronizing its operation (window generation) with the actual clock/data bits of the data stream. The data separation circuit must track the disk input data frequency very closely—unpredictable bit shifts leave less than 50 ns margin to the window edges.

Phase-Locked Loop

Only an analog phase-locked loop (PLL) can provide the reliability required for a double-density data separation circuit. (A phase-locked loop is an electronic circuit that constantly analyzes the frequency of an input signal and locks another oscillator to that frequency.) Using analog PLL techniques, a data separator can be designed with ±1 ns resolution (this would require a 100 MHz clock in a digital phase-locked loop). The analog PLL determines the clock and data bit positions by sampling each bit in the serial data stream. The phase relationship between a data bit and the PLL generated data window is constantly fed back to adjust the position of the data window, enabling the PLL to track input data frequency changes, and thereby reliably read previously recorded data from a floppy disk.

PLL Design

A block diagram of the phase-locked loop described in this application note is shown in Figure 7. Basically, the phase-locked loop operates by comparing the frequency of the input data (from the disk drive) against the frequency of a local oscillator. The difference of these frequencies is used to increase or decrease the frequency of the local oscillator in order to bring its frequency closer to that of the input. The PLL synchronizes the local

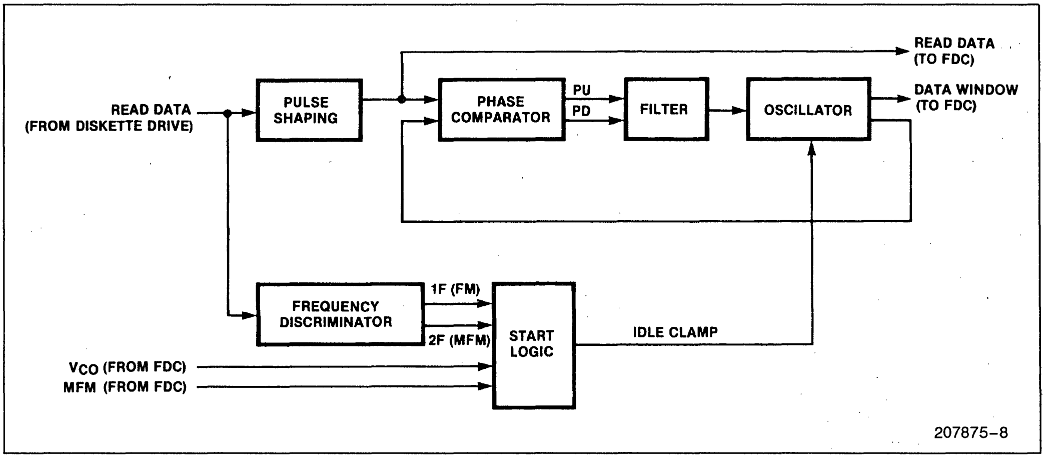


Figure 7. Phase-Locked Loop Data Separator

oscillator to the frequency of the input during the all "zeroes" synchronization field on the floppy disk (immediately preceding both the ID field and the data field.

The PLL consists of nine ICs and is located on page 3 of the schematics in the Appendix. The 8272 VCO output essentially turns the PLL circuitry on and off. When the PLL is off, it "idles" at its center frequency. The VCO turns the PLL on only when valid data is being received from the disk drive. The VCO turns the PLL on after the read/write head has been loaded and the head load time has elapsed. The PLL is turned off in the gap between the ID field and the data field and in the gap after the data field (before the next sector ID field). The GPL parameter in the FDC read and write commands specifies the elapsed time (number of data bytes) that the PLL is turned off in order to blank out discontinuities that appear in the gaps when the write current is turned on and off. The PLL operates with either MFM or FM input data. The MFM output from the FDC controls the PLL operation frequency.

The PLL consists of six functional blocks as follows:

1) Pulse Shaping—A 96LS02 senses a READ DATA pulse and provides a clean output signal to the FDC and to the PLL Phase Comparator and Frequency Discriminator circuitry.

2) Phase Comparator—The phase difference between the PLL oscillator and the READ DATA input is compared. Pump up (PU) and pump down (PD) error signals are derived from this phase difference and output to the filter. If there is no phase difference between the PLL oscillator and the READ DATA input, the PU and PD pulse widths are equal. If the READ DATA pulse occurs early, the PU duration is shorter than the PD duration. If the data pulse occurs late, the PU duration is longer than the PD duration.

3) Filter—This analog circuit filters the PU and PD pulses into an error voltage. This error voltage is buffered by an LM358 operational amplifier.

4. PLL Oscillator—This oscillator is composed of a 74LS393, 74LS74, and 96LS02. The oscillator frequency is controlled by the error voltage output by the filter. This oscillator also generates the data window signal to the FDC.

5. Frequency Discriminator—This logic tracks the READ DATA input from the disk drive and discriminates between the synchronization gap for FM recording (250 KHz) and the gap for MFM recording (500 KHz). Synchronization gaps immediately precede address marks.

6. Start Logic—The function of this logic is to clamp the PLL oscillator to its center frequency (2 MHz) until the FDC VCO signal is enabled and a valid data pattern is sensed by the frequency discriminator. The start logic (consisting of a 74LS393 and 74LS74) ensures that the PLL oscillator is started with zero phase error.

PLL Adjustments

The PLL must be initially adjusted to operate at its center frequency with the VCO output off and the adjustment jumper removed. The 5K timepot should be adjusted until the frequency at the test point (Q output of the 96LS02) is 2 MHz. The jumper should then be replaced for normal operation.

PLL Design Details

The following paragraphs describe the operational and design details of the phase-locked loop data separator illustrated in the appendix. Note that the analog section is operated from a separately filtered +5V supply.

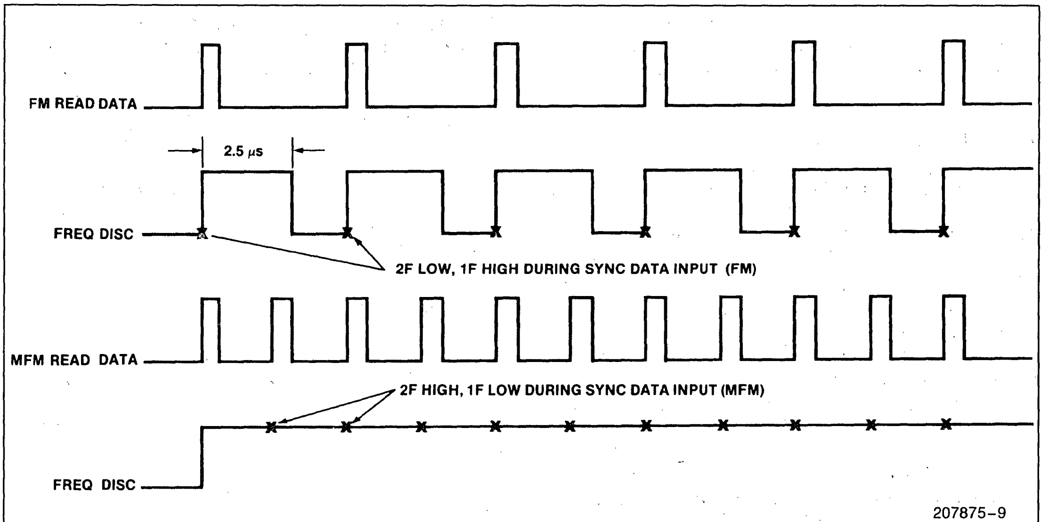
Initialization

As long as the 8272 maintains a low VCO signal, the data separator logic is "turned off". In this state, the PLL oscillator (96LS02) is not oscillating and therefore the 2XBR signal is constantly low. In addition, the pump up (PU) and pump down (PD) signals are inactive (PU low and PD high), the CNT8 signal is inactive (low), and the filter input voltage is held at 2.5 volts by two 1 MΩ resistors between ground and +5 volts.

Floppy Disk Data

The data separator freqeuncy discriminator, the input pulse shaping circuitry, and the start logic are always enabled and respond to rising edges of the READ DATA signal. The rising edge of every data bit from the disk drive triggers two pulse shaping one-shots. The first pulse shaper generates a stable and well-defined 200 ns read data pulse for input to the 8272 and other portions of the data separator logic. The second one-shot generates a 2.5 μs data pulse that is used for input data frequency discrimination.

The frequency discriminator operates as illustrated in Figure 8. The 2F output signal is active (high) during reception of valid MFM (double-density) sync fields on the disk while the 1F signal is active (high) during reception of valid FM (single-density) sync fields. A multiplexer (controlled by the 8272 MFM signal) selects the appropriate 1F or 2F signal depending on the programmed mode.


FM READ DATA
2.5 μs
FREQ DISC
2F LOW, 1F HIGH DURING SYNC DATA INPUT (FM)
MFM READ DATA
2F HIGH, 1F LOW DURING SYNC DATA INPUT (MFM)
FREQ DISC
207875-9

X = Frequency Discriminator Sample Points to Generate 1F and 2F Signals
(a) FM Operation: One-Shot Times Out between Clock Pulses

Figure 8. Input Data Frequency Discrimination

Startup

The data separator is designed to require reception of eight valid sync bits (one sync byte) before enabling the PLL oscillator and attempting to synchronize with the input data stream (see Figure 9). This delay ensures that the PLL will not erroneously synchronize outside a valid sync field in the data stream if the VCO signal is enabled slightly early. The sync bit counter is asynchronously reset by the CNTEN signal when valid sync data is not being recevied by the drive.

Once the VCO signal is active and eight sync bits have been counted, the CNT8 signal is enabled. This signal turns on the PLL oscillator. Note that this oscillator starts synchronously with the rising edge of the disk input data (becasue CNT8 is synchronous with the data rising edge) and the oscillator also starts at its center frequency of 2 MHz (because the LM348 filter input is held at its center voltage of approximately 2.5 volts). This frequency is divided by two and four to generate the 2XBR signal (1 MHz for MFM and 500 KHz for FM).

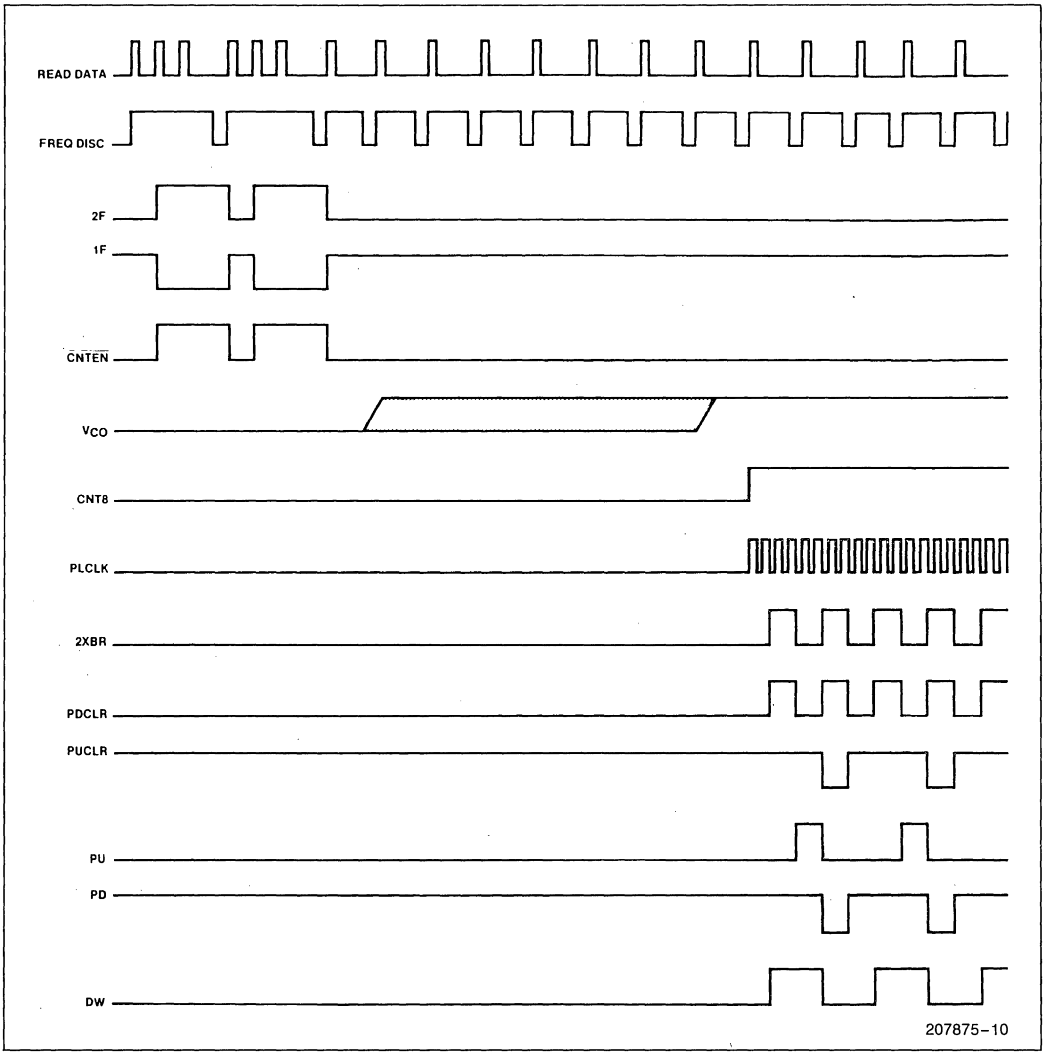


Figure 9. Typical Data Separator Startup Timing Diagram

PLL Synchronization

At this point, the PLL is enabled and begins to synchronize with the input data stream. This synchronization is accomplished very simply in the following manner. The pump up (PU) signal is enabled on the rising edge of the READ DATA from the disk drive. (When the PLL is synchronized with the data stream, this point will occur at the same time as the falling edge of the 2XBR signal as shown in Figure 9). The PU Signal is turned off and the PD signal is activated on the next rising edge of the 2XBR clock. With this scheme, the difference between PU active time and the PD active time is equal to the difference between the input bit rate and the PLL clock rate. Thus, if PU is turned on longer than PD is on, the input bit rate is faster than the PLL clock.

As long as PU and PD are both inactive, no charge is transferred to or from the LM358 input holding capacitor, and the PLL output frequency is maintained (the LM358 operational amplifier has a very high input impedance). Whenver PU is turned on, current flows from the +5 volt supply through a 20K resistor into the holding capacitor. When the PD signal is turned on, current flows from the holding capacitor to ground through a 20K resistor. In this manner, both the pump up and pump down charging rates are balanced.

The change in capacitor charge (and therefore voltage) after a complete PU/PD cycle is proportional to the difference between the PU and PD pulse widths and is also proportional to the frequency difference between the incoming data stream and the PLL oscillator. As the capacitor voltage is raised (PU active longer than PD), the PLL oscillator time constant (RC of the 96LS02) is modified by the filter output (LM358) to raise the oscillator frequency. As the capacitor voltage is lowered (PD active longer than PD), the oscillator frequency is lowered. If both frequencies are equal, the voltage on the holding capacitor does not change, and the PLL oscillator frequency remains constant.

6.0 AN INTELLIGENT DISKETTE DATA BASE SYSTEM

The system described in this application note is designed to function as an intelligent data base controller. The schematics for this data base unit are presented in Appendix A; a block diagram of the unit is illustrated in Figure 10. As designed, the unit can access over four million bytes of mass storage on four floppy disk drives (using a single 8272 FDC); the system can easily be expanded to four FDC devices (and 16 megabytes of on-line disk storage). Three serial data links are also included. These data links may be used by CRT terminals or other microprocessor systems to access the data base.

Processor and Memory

A high-performance 8088 eight-bit microprocessor (operating at 5 MHz with no wait states) controls system operation. The 8088 was selected because of its memory addressing capabilities and its sophisticated string handling instructions. These features improve the speed of data base search operations. In addition, these capabilities allow the system to be easily upgraded with additional memory, disk drives, and if required, a bubble memory or Winchester disk unit.

The schematics for the basic design provide 8K bytes of 2732A high-speed EPROM program storage and 8K bytes of disk directory and file buffer RAM. This memory can easily be expanded to 1 megabyte for performance upgrades.

An 8259A Programmable Interrupt Controller (PIC) is also included in the design to field interrupts from both the serial port and the FDC. This interrupt controller provides a large degree of programming flexibility for the implementation of data base function in an asynchronous, demand driven environment. The PIC allows the system to accumulate asynchronous data base requests from all serial I/O ports while previously specified data base operations are currently in progress. This feature is made possible by the ability of the 8251A RXRDY signal to cause a processor interrupt. After receiving this interrupt, the processor can temporarily halt work on existing requests and enter the incoming information into a data base request buffer. Once the information has been entered into the buffer, the system can resume its previous processing.

In addition, the PIC permits some portions of data base requests to be processed in parallel. For example, once a disk record has been loaded into a memory buffer, a memory search can proceed in parallel with the loading of the next record. After the FDC completes the record transfer, the memory search will be interrupted and the processor can begin another disk transfer before resuming the memory search.

The bus structure of the system is split into three functional buffered units. A 20-bit address from the processor is latched by three-state transparent 74LS373 devices. When the processor is in control of the address and data busses, these devices are output enabled to the system buffered address bus. All I/O devices are placed directly on the local data bus. Finally, the memory data bus is isolated from the local data bus by an 8286 octal transceiver. The direction of this transceiver is determined by the Memory Read signal, while its output enable is activated by a Memory Read or Memory Write command.

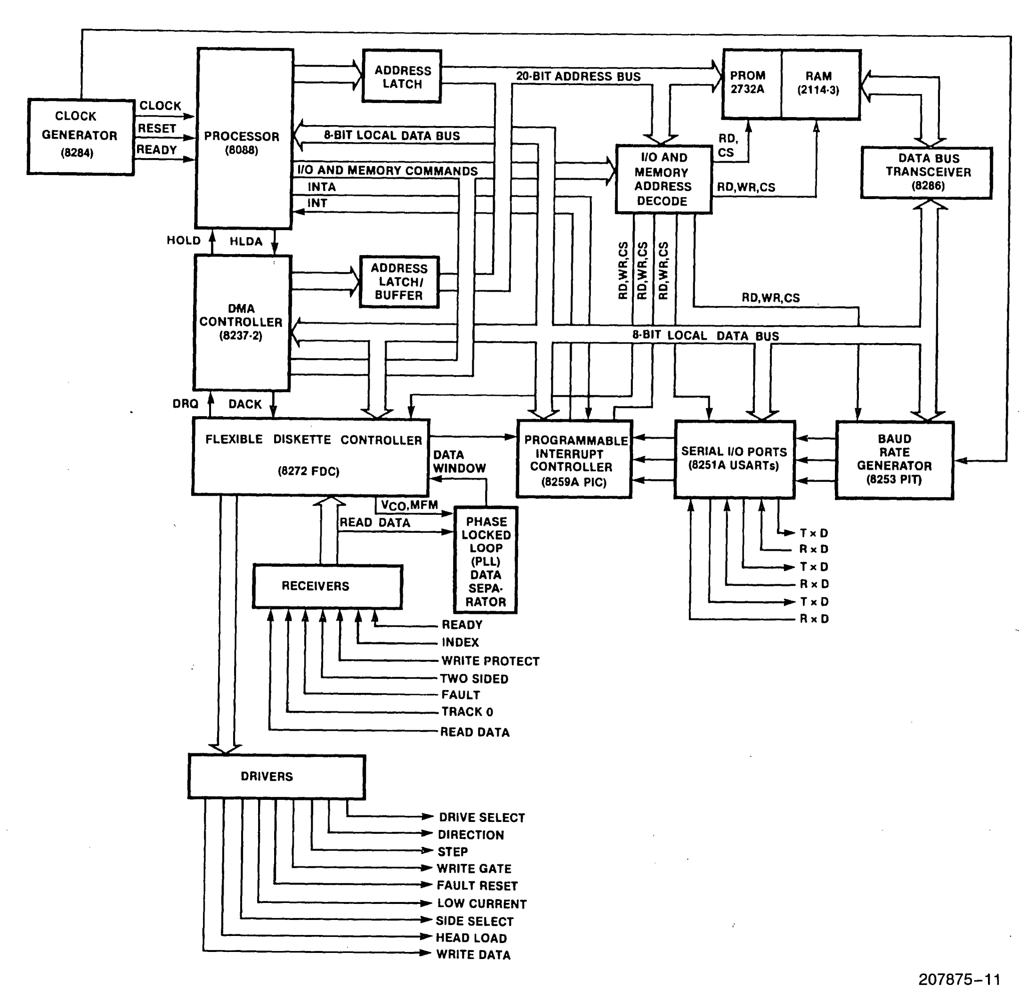


Figure 10. Intelligent Data Base Block Diagram

Serial I/O

The three RS-232-C compatible serial I/O ports operate at software-programmable baud rates to 19.2K. Each I/O port is controlled by an 8251A USART (Universal Synchronous/Asynchronous Receiver/Transmitter). Each USART is individually programmable for operation in many synchronous and asynchronous serial data transmission formats (including IBM Bi-sync). In operation, USART error detection circuits can check for parity, data overrun, and framing errors. An 8253 Programmable Interval Timer is employed to generate the baud rates for the serial I/O ports.

The Transmitter Read and Receiver Ready output signals of the 8251As are routed to the interrupt inputs of the 8259A interrupt controller. These signals interrupt processor execution when a data byte is received by a USART and also when the USART is ready to accept another data byte for transmission.

DMA

The 8272 FDC interfaces to system memory by means of an 8237-2 high-speed DMA controller. Transfers between the disk controller and memory also operate with no wait states when 2114-3 (150 ns) or faster static RAM is used. In operation, the 8272 presents a DMA request to the 8237 for every byte of data to be transferred. This request causes the 8273 to present a HOLD request to the 8088. As soon as the 8088 is able to relinquish data/address bus control, the processor signals a HOLD acknowledge to the 8237. The 8237 then assumes control over the data and address busses. After latching the address for the DMA transfer, the 8237 generates simultaneous I/O Read and Memory Write commands (for a disk read) or simultaneous I/O Write and Memory Read commands (for a disk write). At the same time, the 8272 is selected as the I/O device by means of the DMA acknowledge signal from the 8237. After this single byte has been transferred between the FDC and memory, the DMA controller releases the data/address busses to the 8088 by deactivating the HOLD request. In a short period of time (13 μs for double-density and 27 μs for single-density) the FDC requests a subsequent data transfer. This transfer occurs in exactly the same manner as the previous transfer. After all data transfers have been completed (specified by the word count programmed into the 8237 before the FDC operation was initiated), the 8237 signals a terminal count (EOP pin). This terminal count signal informs the 8272 that the data transfer is complete. Upon reception of this terminal count signal, the 8272 halts DMA requests and initiates an "operation complete" interrupt.

Since the system is designed for a 20-bit addressing, a four-bit DMA-address latch is included as a processor addressable I/O port. The processor writes the upper four DMA address bits before a data transfer. When the DMA controller assumes bus control, the contents of this latch are output enabled on the upper four bits of the address bus. The only restriction in the use of this address latch is that a single disk read or write transfer cannot cross a 64K memory boundary.

Disk Drive Interface

The 8272 FDC may be interfaced to a maximum of four eight-inch floppy disk drives. Both single- and double-density drives are accommodated using the data separation circuit described in section 5. In addition, single- or dual-sided disk drives may be used. The 8272 is driven by an 8 MHz crystal controller clock produced by an 8224 clock generator.

Drive select signals are decoded by means of a 74LS139 from the DS0, DS1 outputs of the FDC. The fault reset, step, low current, and direction outputs to the disk drives are generated from the FR/STEP, LCT/DIR, and RW/SEEK FDC output signals by means of a 74LS240. The other half of the 74LS240 functions as an input multiplexer for the disk write protect, two-sided, fault, and track zero status signals. These signals are multiplexed into the WP/TS and FLT/TRK0 inputs to the 8272.

The 8272 write clock (WR CLK) is generated by a ring counter/multiplexer combination. The write clock frequency is 1 MHz for MFM recording and 500 KHz for FM recording (selected by the MFM output of the 8272). The pulse width is a constant 250 ns. The write clock is constantly generated and input to the FDC (during both read and write operations). The FDC write enable output (WE) is transmitted directly to the write gate disk drive input.

Write data to the disk drive is preshifted (according to the PS0, PS1 FDC outputs) by the combination of a 74LS175 four-bit latch and a 74LS153 multiplexer. The amount of preshift is completely controlled within the 8272 FDC. Three cases are possible: the data may be written one clock cycle early, one clock cycle late, or with no preshift. The data preshift circuit is activated by the FDC only in the double-density mode. The preshift is required to cancel predictable playback data shifts when recorded data is later read from the floppy disk.

A single 50-conductor flat cable connects the board to the floppy disk drives. FDC outputs are driven by 7438 open collector high-current line-drivers. These drivers are resistively terminated on the last disk drive by means of a 150Ω resistor to +5V. The line receivers are 7414 Schmitt triggered inverters with 150Ω pull-up resistors on board.

7.0 SPECIAL CONSIDERATIONS

This section contains a quick review of key features and issues, most of which have been mentioned in other sections of this application note. Before designing with the 8272 FDC, it is advisable that the information in this section be completely understood.

7.1 Multi-Sector Transfers

The 8272 always operates in a multi-sector transfer mode. The 8272 continues to transfer data until the TC input is activated. In a DMA configuration, the TC input of the 8272 must always be connected to the EOP/TC output of the DMA controller. When multiple DMA channels are used on a single DMA controller, EOP must be gated with the select signal for the proper FDC. If the TCD signal is not gated, a terminal count on another channel will abort FDC operation.

In a processor driven configuration with no DMA controller, the system must count the transfers and supply a TC signal to the FDC. In a DMA environment, ORing a programmable TC with the TC from the DMA controller is a convenient means of ensuring that the processor may always gain control of the FDC (even if the diskette system hangs up in an abnormal manner).

7.2 Processor Command/Result Phase Interface

In the command phase, the processor must write the exact number of parameters in the exact order shown in Table 5. During the result phase, the processor must read the complete result status. For example, the Format Track command requires six command bytes and presents seven result bytes. The 8272 will not accept a new command until all result bytes are read. Note that the number of command and result bytes varies from command-to-command. Command and result phases cannot be shortened.

During both the command and result phases, the Main Status Register must be read by the processor before each byte of information is read from, or written to, the FDC Data Register. Before each command byte is written, DIO (bit 6) must be low (indicating a data transfer from the processor) and RQM (bit 7) must be high (indicating that the FDC is ready for data). During the result phase, DIO must be high (indicating a data transfer to the processor) and RQM must also be high (indicating that data is ready for the processor).

NOTE:
After the 8272 receives a command byte, the RQM flag may remain set for 12 microseconds (with an 8 MHz clock). Software should not attempt to read the Main Status Register before this time interval has elapsed; otherwise, the software will erroneously assume that the FDC is ready to accept the next byte.

7.3 Sector Sizes

The 8272 does not support 128 byte sectors in the MFM (double-density) mode.

7.4 Write Clock

The FDC Write Clock input (WR CLK) must be present at all times.

7.5 Reset

The FDC Reset input (RST) must be held active during power-on reset while the RD and WR inputs are active. If the reset input becomes inactive while RD and WR are still active, the 8272 enters the test mode. Once activated, the test mode can only be deactivated by a power-down condition.

7.6 Drive Status

The 8272 constantly polls (starting after the power-on reset) all drives for changes in the drive ready status. At power-on, the FDC assumes that all drives are not ready. If a drive application requires that the ready line be strapped active, the FDC will generate an interrupt immediately after power is applied.

7.7 Gap Length

Only the gap 3 size is software programmable. All other gap sizes are fixed. In addition, different gap 3 sizes must be specified in format, read, write, and scan commands. Refer to Section 3 and Table 9 for gap size recommendations.

7.8 Seek Command

The drive busy flag in the Main Status Register remains set after a Seek command is issued until the Sense Interrupt Status command is issued (following reception of the seek complete interrupt).

The FDC does not perform implied seeks. Before issuing data read or write commands, the read/write head must be positioned over the correct cylinder. If the head is not positioned correctly, a cylinder address error is generated.

After issuing a step pulse, the 8272 resumes drive status polling. For correct stepper operation in this mode, the stepper motor must be constantly enabled. (Most drives provide a jumper to permit the stepper motor to be constantly enabled.)

7.9 Step Rate

The 8272 can emit a step pulse that is one millisecond faster than the rate programmed by the SRT parameter in the Specify command. This action may cause subsequent sector not found errors. The step rate time should be programmed to be 1 ms longer than the step rate time required by the drive.

7.10 Cable Length

A cable length of less than 10 feet is recommended for drive interfacing.

7.11 Scan Commands

The current 8272 has several problems when using the scan commands. These commands should not be used at this time.

7.12 Interrupts

When the processor receives an interrupt from the FDC, the FDC may be reporting one of two distinct events:

a) The beginning of the result phase of a previously requested read, write, or scan command.

b) An asynchronous event such as a seek/recalibrate completion, an attention, an abnormal command termination, or an invalid command.

These two cases are distinguished by the FDC busy flag (bit 4) in the Main Status Register. If the FDC busy flag is high, the interrupt is of type (a). If the FDC busy flag is low, the interrupt was caused by an asynchronous event (b).

A single interrupt from the FDC may signal more than one of the above events. After receiving an interrupt, the processor must continue to issue Sense Interrupt Status commands (and service the resulting conditions) until an invalid command code is received. In this manner, all "hidden" interrupts are ferreted out and serviced.

7.13 Skip Flag (SK)

The skip flag is used during the execution of Read Data, Read Deleted Data, Read Track, and various Scan commands. This flag permits the FDC to skip unwanted sectors on a disk track.

When performing a Read Data, Read Track, or Scan command, a high SK flag indicates that the FDC is to skip over (not transfer) any sector containing a deleted data address mark. A low SK flag indicates that the FDC is to terminate the command (after reading all the data in the sector) when a deleted data address mark is encountered.

When performing a Read Deleted Data command, a high SK flag indicates that sectors containing normal data address marks are to be skipped. Note that his is just the opposite situation from that described in the last paragraph. When a data address mark is encountered during a Read Deleted Data command (and the SK flag is low), the FDC terminates the command after reading all the data in the sector.

7.14 Bad Track Maintenance

The 8272 does not internally maintain bad track information. The maintenance of this information must be performed by system software. As an example of typical bad track operation, assume that a media test determines that track 31 and track 66 of a given floppy disk are bad. When the disk is formatted for use, the system software formats physcial track 0 as logical cylinder 0 (C = 0 in the command phase parameters), physical track 1 as logical track 1 (C = 1), and so on, until physical track 30 is formatted as logical cylinder 30 (C = 30). Physical track 31 is bad and should be formatted as logical cylinder FF (indicating a bad track). Next, physical track 32 is formatted as logical cylinder 31, and so on, until physical track 67 is formatted as logical cylinder 64. Next, bad physical track 66 is formatted as logcial cylinder FF (another bad track marker), and physical track 67 is formatted as logical cylinder 65. This formatting continues until the last physical track (77) is formatted as logical cylinder 75. Normally, after this formatting is complete, the bad track information is stored in a prespecified area on the floppy disk (typically in a sector on track 0) so that the system will be able to recreate the bad track information when the disk is removed from the drive and reinserted at some later time.

To illustrate how the system software performs a transfer operation disk with bad tracks, assume that the disk drive head is positioned at track 0 and the disk described above is loaded into the drive. If a command to read track 36 is issued by an application program, the system software translates this read command into a seek to physical track 37 (since there is one bad track between 0 and 36, namely 31) followed by a read of logical cylinder 36. Thus, the cylinder parameter C is set to 37 for the Seek command and 36 for the Read Sector command.

7.15 Head Load versus Head Settle Times

The 8272 does not permit separate specification of the head load time and the head settle time. When the Specify command is issued for a given disk drive, the proper value for the HLT parameter is the maximum of the head load time and the head settle time.

APPENDIX A SCHEMATICS

Power Distribution

Part	Ref Desig	+5	GND	+12	−12
8088	A2	40	1, 20		
8224	I6	9, 16	8		
8237-2	A6	31	20		
8251A	A9, B9, C9	26	4		
8253-5	A10	24	12		
8259A	B10	28	14		
8272	D10	40	20		
8284	A1	18	9		
8286	B6, F4	20	10		
2114	F1, F2, G1, G2, H1, H2, I1, I2	18	9		
2732A	D1, D2	24	12		
74LS00	E1	14	7		
74LS04	B2, E6, E8, F8	14	7		
74LS27	E2, E5	14	7		
74LS32	B1	14	7		
74LS74	A4, G5, H6	14	7		
74LS138	F3	16	8		
74LS139	E10	16	8		
74LS153	I3	16	8		
74LS157	F6	16	8		
74LS164	F5	14	7		
74LS173	G3	16	8		
74LS175	G4	16	8		
74LS240	G10	20	10		
74LS257	D3	16	8		
74LS367	C3, E9	16	8		
74LS373	B4, C4, D4, C6	20	10		
74LS393	I5, F7	14	7		
74S08	E4	14	7		
74S138	D6, E3	16	8		
7414	H7	14	7		
7438	H8, H9, H10	14	7		
1488	H3		7	14	1
1489	H4	14	7		
96LS02	G7	16	8		
96LS02	G6			16	8
LM358	H5			8	4

REFERENCES

1) Intel, "8272 Single/Double Density Floppy Disk Controller Data Sheet," Intel Corporation, 1980.

2) Intel, *iSBC 208 Hardware Reference Manual,* Manual Order No. 143078, Intel Corporation, 1980.

3) Intel, *iSBC 204 Flexible Diskette Controller Hardware Reference Manual,* Manual Order No. 9800568A, Intel Corporation, 1978.

4) Shugart, *SA800/801 Diskette Storage Drive OEM Manual,* Part No. 50574, Shugart Associates, 1977.

5) Shugart, *SA800/801 Diskette Storage Drive Theory of Operations,* Part No. 50664, Shugart Associates, 1977.

6) Shugart, *SA800 Series Diskette Storage Drive Double Density Design Guide,* Part No. 39000, Shugart Associates, 1977.

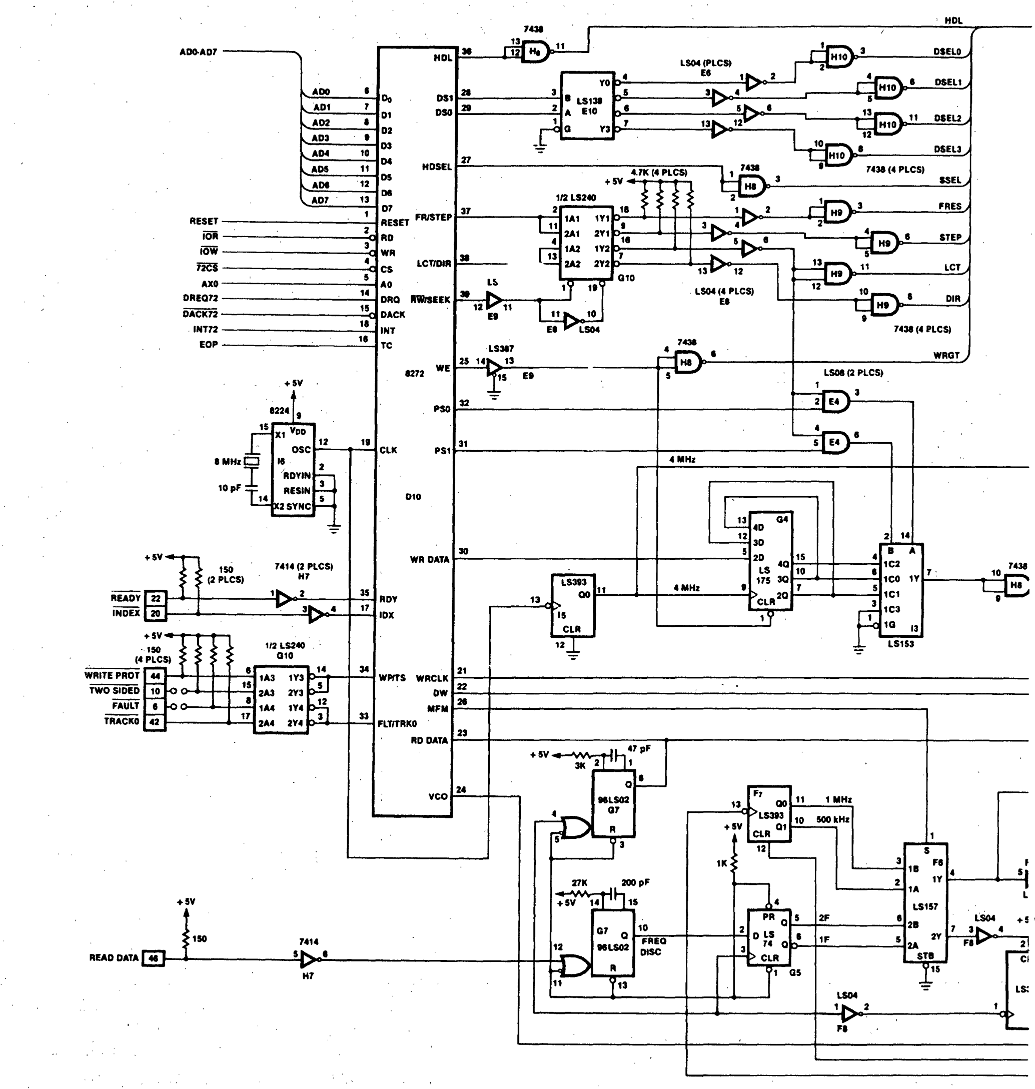
7) Shugart, "Application Notes for Shugart Dual VFO," Part No. 39101, Shugart Associates, 1980.

8) Pertec, "Soft-sector Formatting for PERTEC Flexible Disk Drives," Pertec Application Note, 1977.

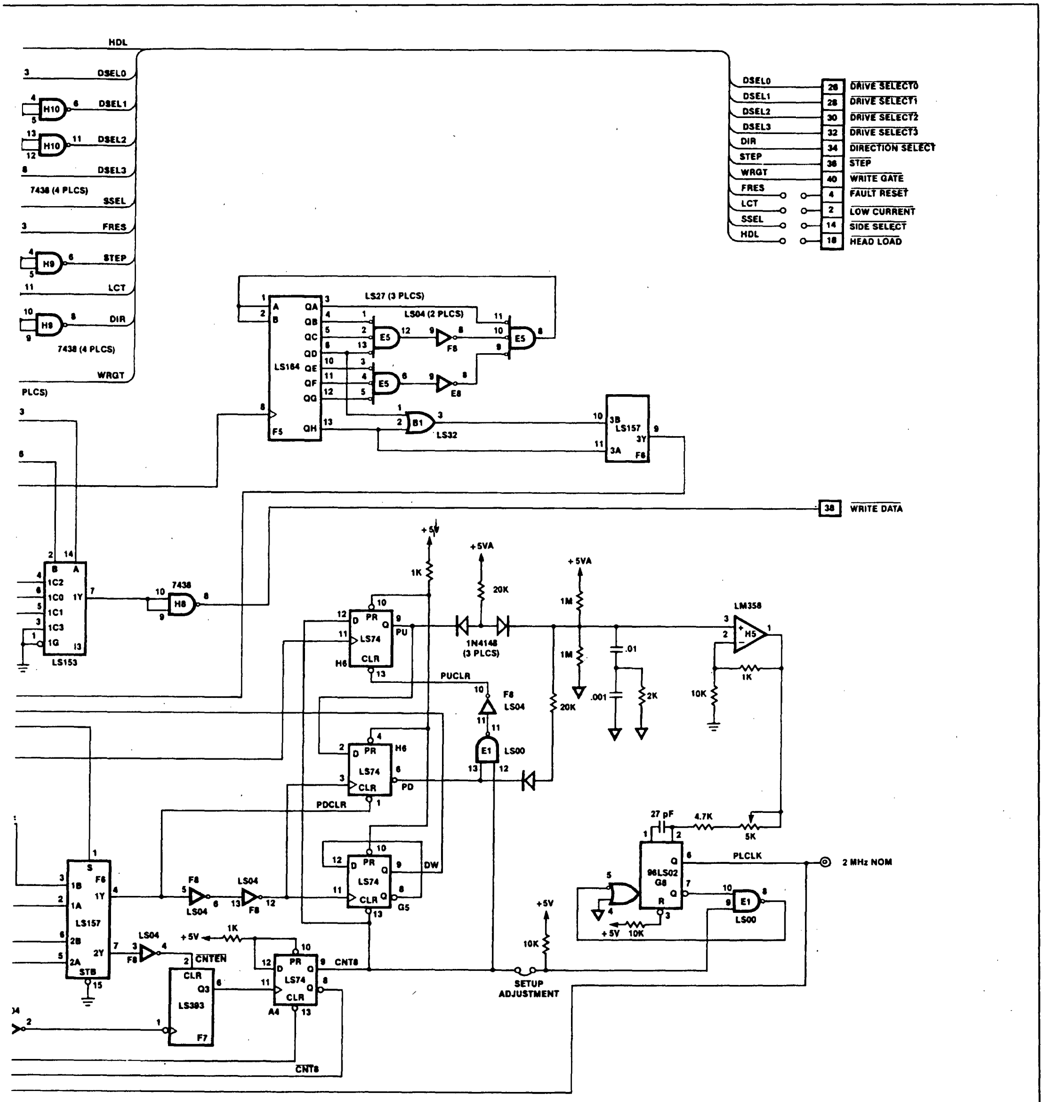
9) Ausing Lesea and Rodnay Zaks, "Floppy-disc Controller Design Must Begin With the Basics," EDN, May 20, 1978.

10) John Hoeppner and Larry Wall, "Encoding/Decoding Techniques Double Floppy Disc Capacity," Computer Design, Feb 1980.

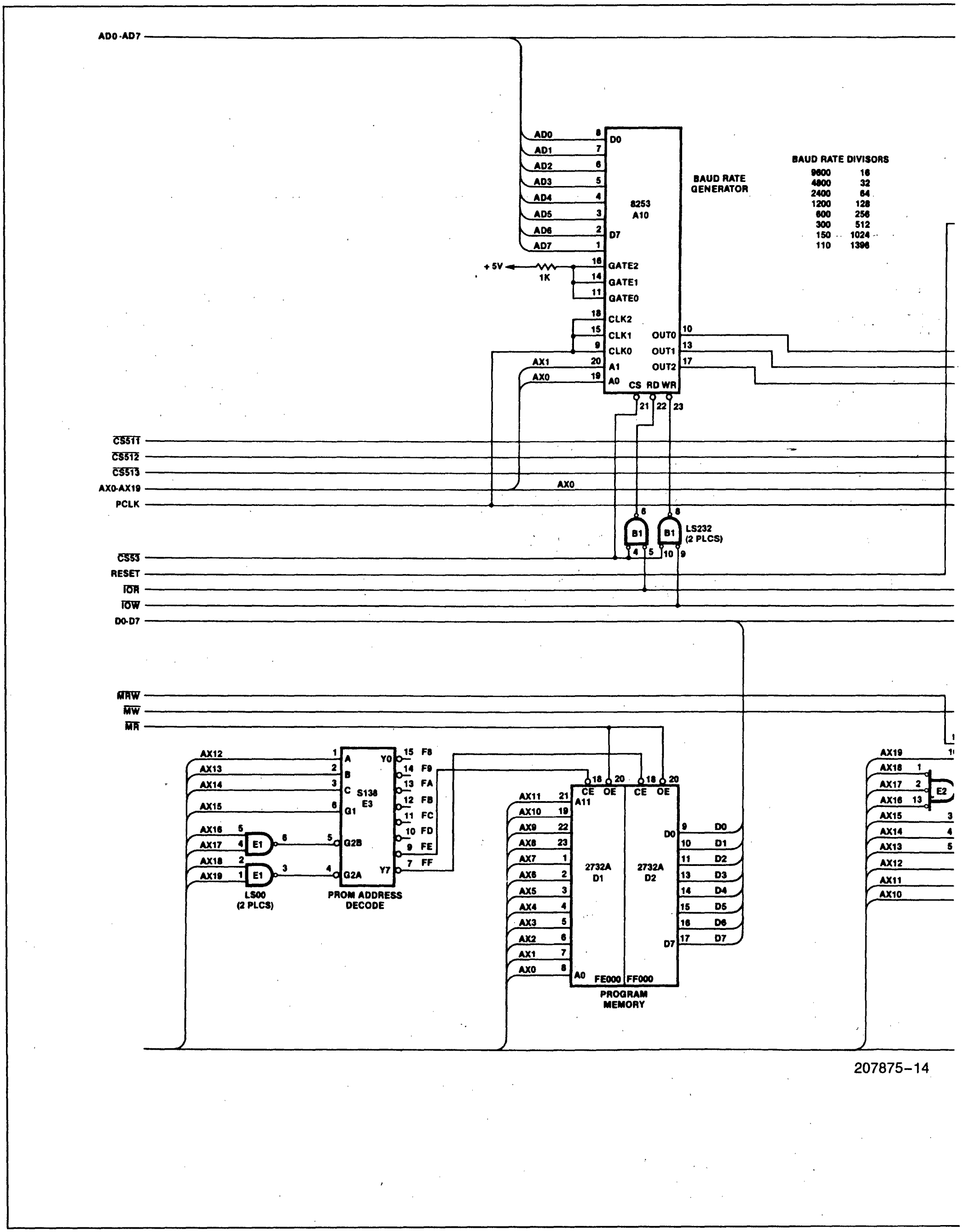
11) John Zarrella, *System Architecture,* Microcomputer Applications, 1980.



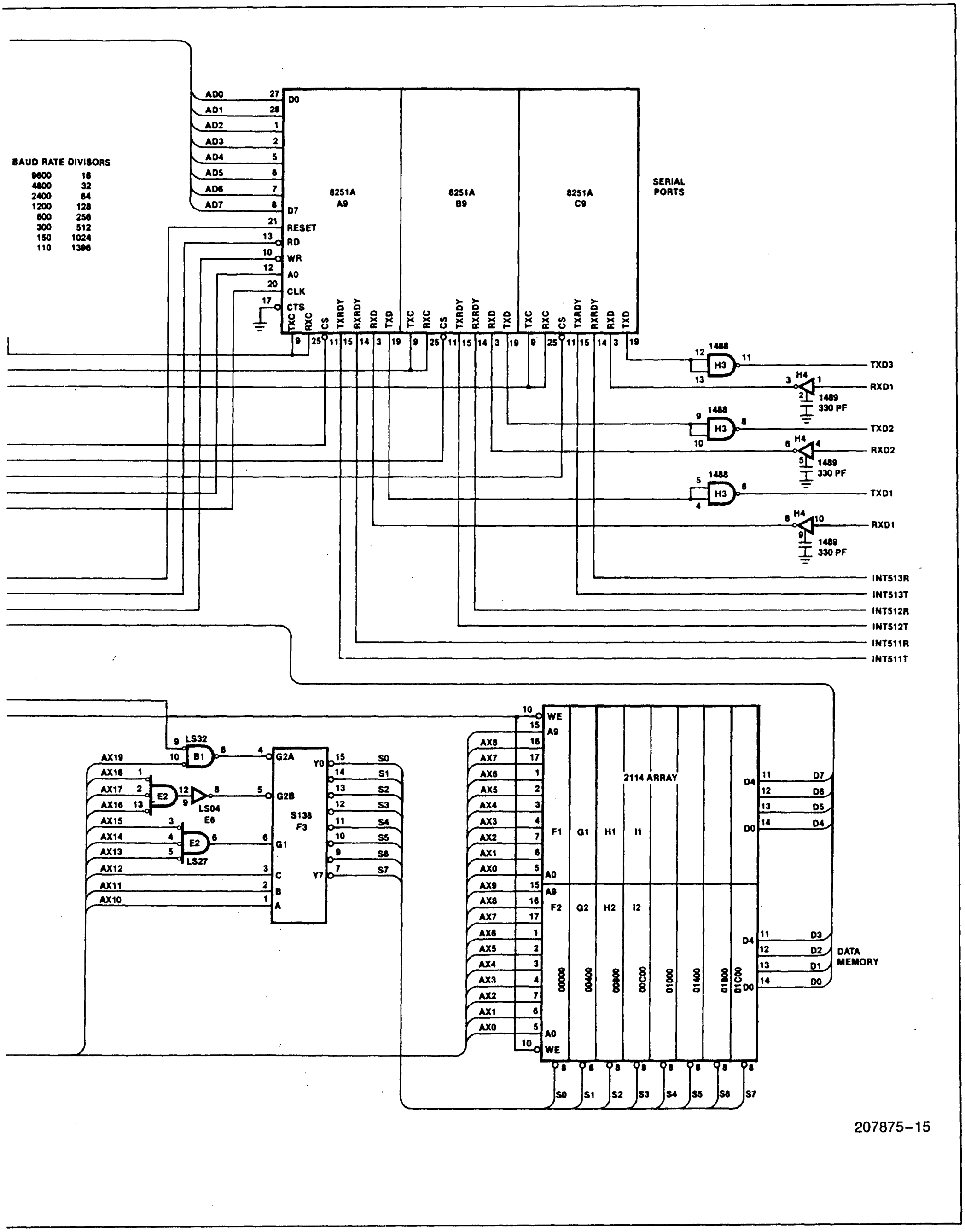
207875-12


HDL
DSEL0
DSEL1
DSEL2
DSEL3
H10
7438 (4 PLCS)
SSEL
FRES
STEP
H9
LCT
DIR
WRGT
PLCS)
DSEL0
DSEL1
DSEL2
DSEL3
DIR
STEP
WRGT
FRES
LCT
SSEL
HDL
26 DRIVE SELECT0
28 DRIVE SELECT1
30 DRIVE SELECT2
32 DRIVE SELECT3
34 DIRECTION SELECT
36 STEP
40 WRITE GATE
4 FAULT RESET
2 LOW CURRENT
14 SIDE SELECT
18 HEAD LOAD
LS27 (3 PLCS)
LS04 (2 PLCS)
LS164
F5
E5
F8
E8
B1
LS32
LS157
F6
38 WRITE DATA
LS153
7438
H8
+5V
1K
+5VA
20K
1M
1N4148
(3 PLCS)
LS74
H6
PU
PUCLR
F8
LS04
E1
LS00
.01
.001
2K
LM358
H5
10K
PD
PDCLR
DW
G5
27 pF
4.7K
5K
96LS02
G8
PLCLK
2 MHz NOM
LS04
F8
1K
CNTEN
CLR
Q3
LS393
F7
A4
CNT8
SETUP
ADJUSTMENT

207875–13



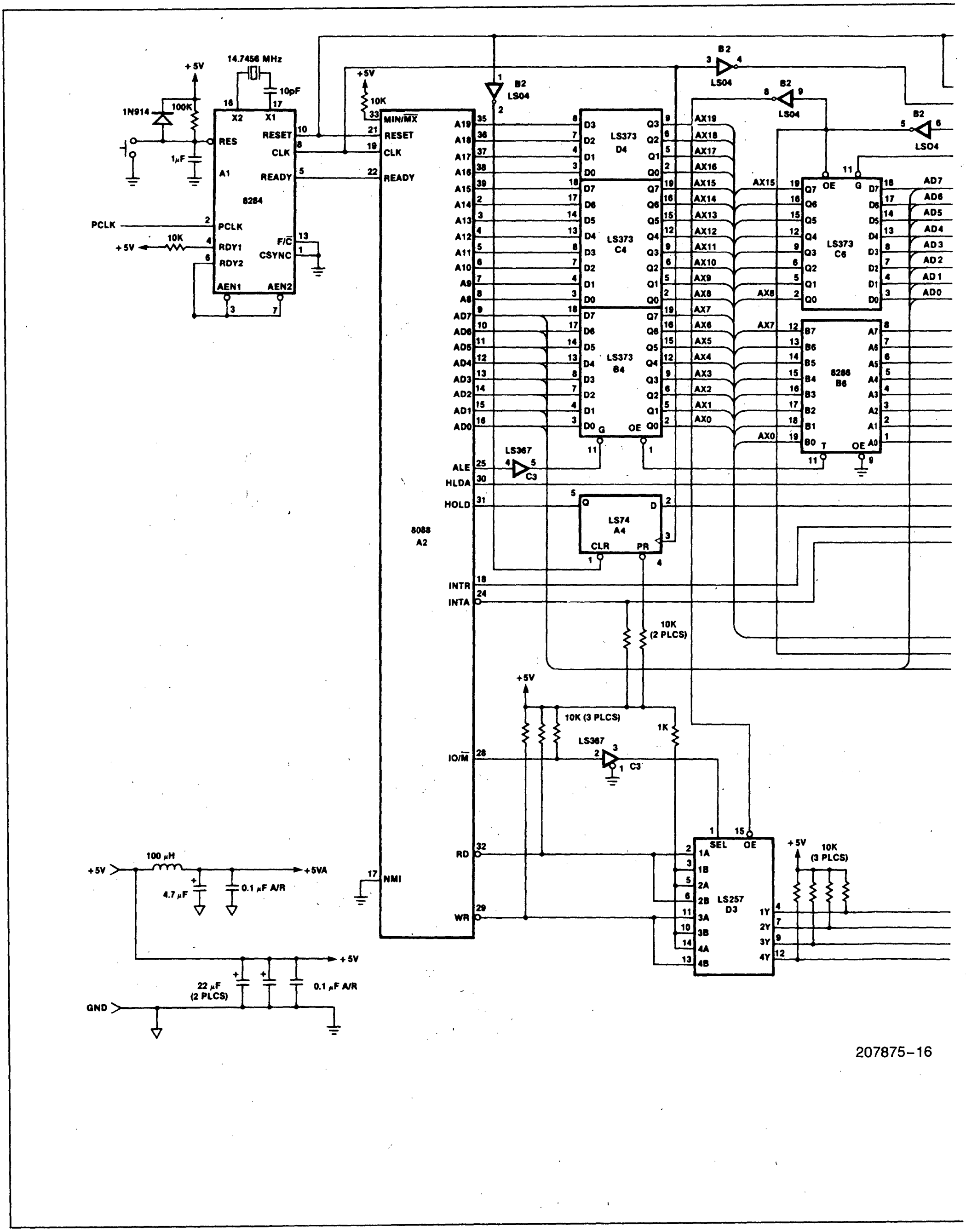
BAUD RATE DIVISORS	
9600	16
4800	32
2400	64
1200	128
600	256
300	512
150	1024
110	1396



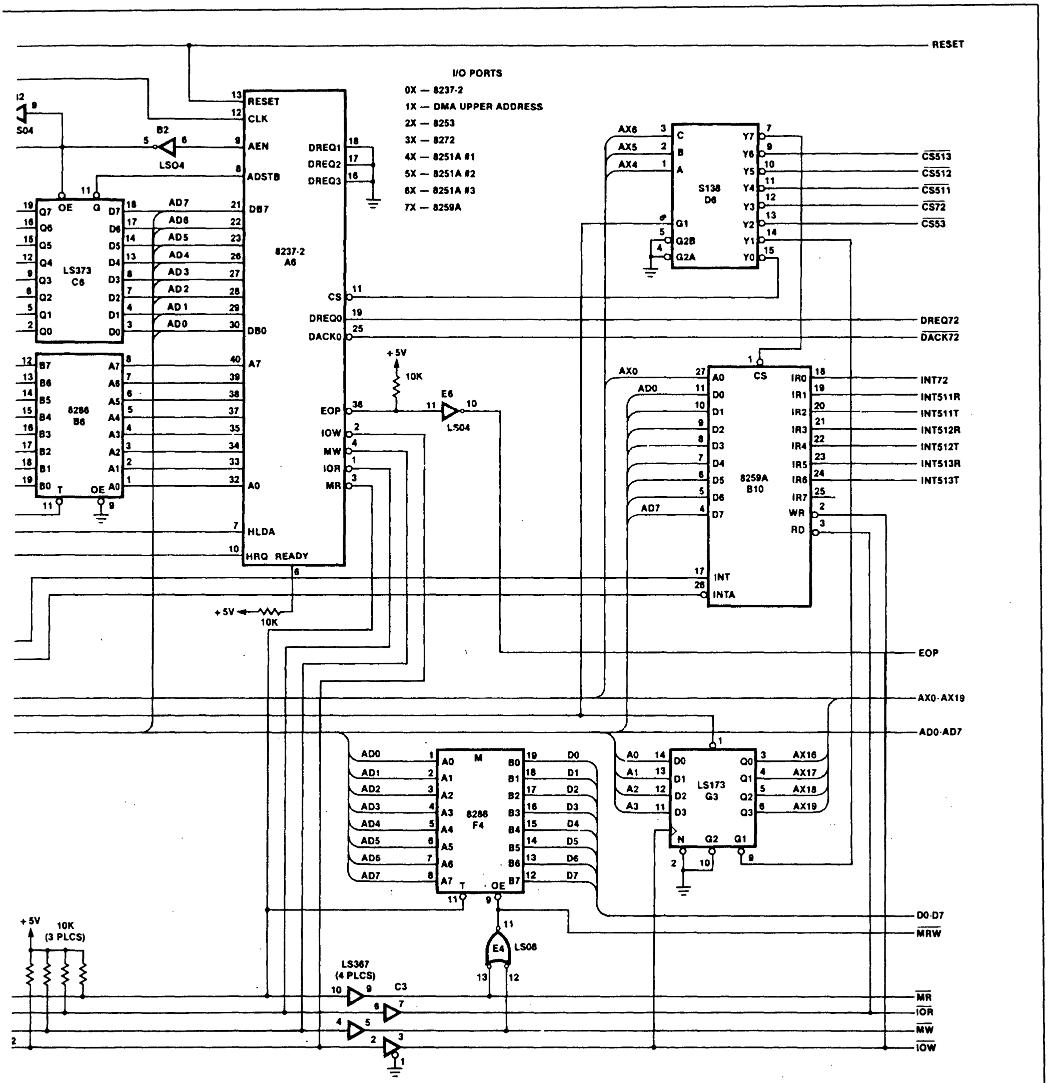
BAUD RATE DIVISORS

9600	16
4800	32
2400	64
1200	128
600	256
300	512
150	1024
110	1396

207875-15


14.7456 MHz
+5V
10pF
1N914
100K
1μF
X2
X1
RES
RESET
CLK
A1
READY
8284
PCLK
RDY1
RDY2
F/C̅
CSYNC
AEN1
AEN2
10K
MIN/M̅X̅
8088
A2
A19
A18
A17
A16
A15
A14
A13
A12
A11
A10
A9
A8
AD7
AD6
AD5
AD4
AD3
AD2
AD1
AD0
ALE
HLDA
HOLD
INTR
INTA
IO/M̅
RD
WR
NMI
B2
LS04
LS373
D4
C4
B4
LS367
C3
LS74
A4
CLR
PR
AX19
AX18
AX17
AX16
AX15
AX14
AX13
AX12
AX11
AX10
AX9
AX8
AX7
AX6
AX5
AX4
AX3
AX2
AX1
AX0
LS373
C6
OE
G
8286
B6
T
10K (2 PLCS)
10K (3 PLCS)
1K
LS257
D3
SEL
1A
1B
2A
2B
3A
3B
4A
4B
1Y
2Y
3Y
4Y
10K (3 PLCS)
100 μH
+5VA
4.7 μF
0.1 μF A/R
22 μF (2 PLCS)
GND

207875-16


RESET
I/O PORTS
0X — 8237-2
1X — DMA UPPER ADDRESS
2X — 8253
3X — 8272
4X — 8251A #1
5X — 8251A #2
6X — 8251A #3
7X — 8259A
8237-2
A6
RESET
CLK
AEN
ADSTB
DB7
DB0
A7
A0
HLDA
HRQ
READY
DREQ1
DREQ2
DREQ3
CS
DREQ0
DACK0
EOP
IOW
MW
IOR
MR
B2
LS04
S04
LS373
C6
OE
Q
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Q0
D7
D6
D5
D4
D3
D2
D1
D0
AD7
AD6
AD5
AD4
AD3
AD2
AD1
AD0
8286
B6
B7
B6
B5
B4
B3
B2
B1
B0
A7
A6
A5
A4
A3
A2
A1
A0
T
OE
+5V
10K
E6
LS04
S138
D6
AX6
AX5
AX4
C
B
A
G1
G2B
G2A
Y7
Y6
Y5
Y4
Y3
Y2
Y1
Y0
CS513
CS512
CS511
CS72
CS53
DREQ72
DACK72
8259A
B10
AX0
AD0
AD7
A0
D0
D1
D2
D3
D4
D5
D6
D7
INT
INTA
IR0
IR1
IR2
IR3
IR4
IR5
IR6
IR7
WR
RD
INT72
INT511R
INT511T
INT512R
INT512T
INT513R
INT513T
EOP
AX0-AX19
AD0-AD7
8286
F4
M
A0
A1
A2
A3
A4
A5
A6
A7
B0
B1
B2
B3
B4
B5
B6
B7
D0
D1
D2
D3
D4
D5
D6
D7
T
OE
LS173
G3
D0
D1
D2
D3
Q0
Q1
Q2
Q3
N
G2
G1
AX16
AX17
AX18
AX19
D0-D7
MRW
E4
LS08
+5V
10K
(3 PLCS)
LS367
(4 PLCS)
C3
MR
IOR
MW
IOW

207875–17

APPLICATION NOTE

AP-121

November 1986

Software Design and Implementation of Floppy Disk Subsystems

Order Number: 207885-001

1.0 INTRODUCTION

Disk interface software is a major contributor to the efficient and reliable operation of a floppy disk subsystem. This software must be a well-designed compromise between the needs of the application software modules and the capabilities of the floppy disk controller (FDC). In an effort to meet these requirements, the implementation of disk interface software is often divided into several levels of abstraction. The purpose of this application note is to define these software interface levels and describe the design and implementation of a modular and flexible software driver for the 8272 FDC. This note is a companion to AP-116, "An Intelligent Data Base System Using the 8272".

The Physical Interface Level

The software interface level closest to the FDC hardware is referred to as the physical interface level. At this level, interface modules (often called disk drivers or disk handlers) communicate directly with the FDC device. Disk drivers accept floppy disk commands from other software modules, control and monitor the FDC execution of the commands, and finally return operational status information (at command termination) to the requesting modules.

In order to perform these functions, the drivers must support the bit/byte level FDC interface for status and data transfers. In addition, the drivers must field, classify, and service a variety of FDC interrupts.

The Logical Interface Level

System and application software modules often specify disk operation parameters that are not directly compatible with the FDC device. This software incompatibility is typically caused by one of the following:

1) The change from an existing FDC to a functionally equivalent design. Replacing a TTL based controller with an LSI device is an example of a change that may result in software incompatibilities.
2) The upgrade of an existing FDC subsystem to a higher capability design. An expansion from a single-sided, single-density system to a dual-sided, double-density system to increase data storage capacity is an example of such a system change.
3) The abstraction of the disk software interface to avoid redundancy. Many FDC parameters (in particular the density, gap size, number of sectors per track and number of bytes per sector) are fixed for a floppy disk (after formatting). In fact, in many systems these parameters are never changed during the life of the system.
4) The requirement to support a software interface that is independent of the type of disk attached to the system. In this case, a system generated ("logical") disk address (drive, head, cylinder, and sector numbers) must be mapped into a physical floppy disk address. For example, to switch between single- and dual-sided disks, it may be easier and more cost-effective for the software to treat the dual-sided disk as containing twice as many sectors per track (52) rather than as having two sides. With this technique, accesses to sectors 1 through 26 are mapped onto head 0 while accesses to sectors 27 through 52 are mapped onto head 1.
5) The necessity of supporting a bad track map. Since bad tracks depend on the disk media, the bad track mapping varies from disk to disk. In general, the system and application software should not be concerned with calculating bad track parameters. Instead, these software modules should refer to cylinders logically (0 through 76). The logical interface level procedures must map these cylinders into physical cylinder positions in order to avoid the bad tracks.

The key to logical interface software design is the mapping of the "logical disk interface" (as seen by the application software) into the "physical disk interface" (as implemented by the floppy disk drivers). This logical to physical mapping is tightly coupled to system software design and the mapping serves to isolate both applications and system software from the peculiarities of the FDC device. Typical logical interface procedures are described in Table 1.

The File System Interface Level

The file system typically comprises the highest level of disk interface software used by application programs. The file system is designed to treat the disk as a collection of named data areas (known as files). These files are cataloged in the disk directory. File system interface software permits the creation of new files and the deletion of existing files under software control. When a file is created, its name and disk address are entered into the directory; when a file is deleted, its name is removed from the directory. Application software requests the use of a file by executing an OPEN function. Once opened, a file is normally reserved for use by the requesting program or task and the file cannot be reopened by other tasks. When a task no longer needs to use an open file, the task closes the file, releasing it for use by other tasks.

Most file systems also support a set of file attributes that can be specified for each file. File attributes may be used to protect files (e.g., the WRITE PROTECT attribute ensures that an existing file cannot accidentally be overwritten) and to supply system configuration information (e.g., a FORMAT attribute may specify that a file should automatically be created on a new disk when the disk is formatted).

At the file system interface level, application programs need not be explicitly aware of disk storage allocation techniques, block sizes, or file coding strategies. Only a "file name" must be presented in order to open, read or write, and subsequently close a file. Typical file system functions are listed in Table 2.

Table 1. Examples of Logical Interface Procedures

Name	Description
FORMAT DISK	Controls physical disk formatting for all tracks on a disk. Formatting adds FDC recognized cylinder, head, and sector addresses as well as address marks and data synchronization fields (gaps) to the floppy disk media.
RECALIBRATE	Moves the disk read/write head to track 0 (at the outside edge of the disk).
SEEK	Moves the disk read/write head to a specified logical cylinder. The logical and physical cylinder numbers may be different if bad track mapping is used.
READ STATUS	Indicates the status of the floppy disk drive and media. One important use of this procedure is to determine whether a floppy disk is dual-sided.
READ SECTOR	Reads one or more complete sectors starting at a specified disk address (drive, head, cylinder, and sector).
WRITE SECTOR	Writes one or more complete sectors starting at a specified disk address (drive, head, cylinder, and sector).

Table 2. Disk File System Functions

Name	Description
OPEN	Prepare a file for processing. If the file is to be opened for input and the file name is not found in the directory, an error is generated. If the file is opened for output and the file name is not found in the directory, the file is automatically created.
CLOSE	Terminate processing of an open file.
READ	Transfer Data from an open file to memory. The READ function is often designed to buffer one or more sectors of data from the disk drive and supply this data to the requesting program, as required.
WRITE	Transfer data from memory to an open file. The WRITE function is often designed to buffer data from the application program until enough data is available to fill a disk sector.
CREATE	Initialize a file and enter its name and attributes into the file directory.
DELETE	Remove a file from the directory and release its storage space.
RENAME	Change the name of a file in the directory.
ATTRIBUTE	Change the attributes of a file.
LOAD	Read a file of executable code into memory.
INITDISK	Initialize a disk by formatting the media and establishing the directory file, the bit map file, and other system files.

Scope of this Note

This application note directly addresses the logical and physical interface levels. A complete 8272 driver (including interrupt service software) is listed in Appendix A. In addition, examples of recalibrate, seek, format, read, and write logical interface level procedures are included as part of the exerciser program found in Appendix B. Wherever possible, specific hardware configuration dependencies are parametized to provide maximum flexibility without requiring major software changes.

2.0 Disk I/O Techniques

One of the most important software aspects of disk interfacing is the fixed sector size. (Sector sizes are fixed when the disk is formatted.) Individual bytes of disk storage cannot be read/written; instead, complete sectors must be transferred between the floppy disk and system memory.

Selection of the appropriate sector size involves a tradeoff between memory size, disk storage efficiency, and disk transfer efficiency. Basically, the following factors must be weighed:

1) Memory size. The larger the sector size, the larger the memory area that must be reserved for use during disk I/O transfers. For example, a 1K byte disk sector size requires that at least one 1K memory block be reserved for disk I/O.
2) Disk Storage efficiency. Both very large and very small sectors can waste disk storage space as follows. In disk file systems, space must be allocated somewhere on the disk to link the sectors of each file together. If most files are composed of many small sectors, a large amount of linkage overhead information is required. At the other extreme, when most files are smaller than a single disk sector, a large amount of space is wasted at the end of each sector.
3) Disk transfer efficiency. A file composed of a few large sectors can be transferred to/from memory more efficiently (faster and with less overhead) than a file composed of many small sectors.

Balancing these considerations requires knowledge of the intended system applications. Typically, for general purpose systems, sector sizes from 128 bytes to 1K bytes are used. For compatibility between single-density and double-density recording with the 8272 floppy disk controller, 256 byte sectors or 512 byte sectors are most useful.

FDC Data Transfer Interface

Three distinct software interface techniques may be used to interface system memory to the FDC device during sector data transfers:

1) DMA—In a DMA implementation, the software is only required to set up the DMA controller memory address and transfer count, and to initiate the data transfer. The DMA controller hardware handshakes with the processor/system bus in order to perform each data transfer.
2) Interrupt Driven—The FDC generates an interrupt when a data byte is ready to be transferred to memory, or when a data byte is needed from memory. It is the software's responsibility to perform appropriate memory reads/writes in order to transfer data from/to the FDC upon receipt of the interrupt.
3) Polling—Software responsibilities in the polling mode are identical to the responsibilities in the interrupt driven mode. The polling mode, however, is used when interrupt service overhead (context switching) is too large to support the disk data rate. In this mode, the software determines when to transfer data by continually polling a data request status flag in the FDC status register.

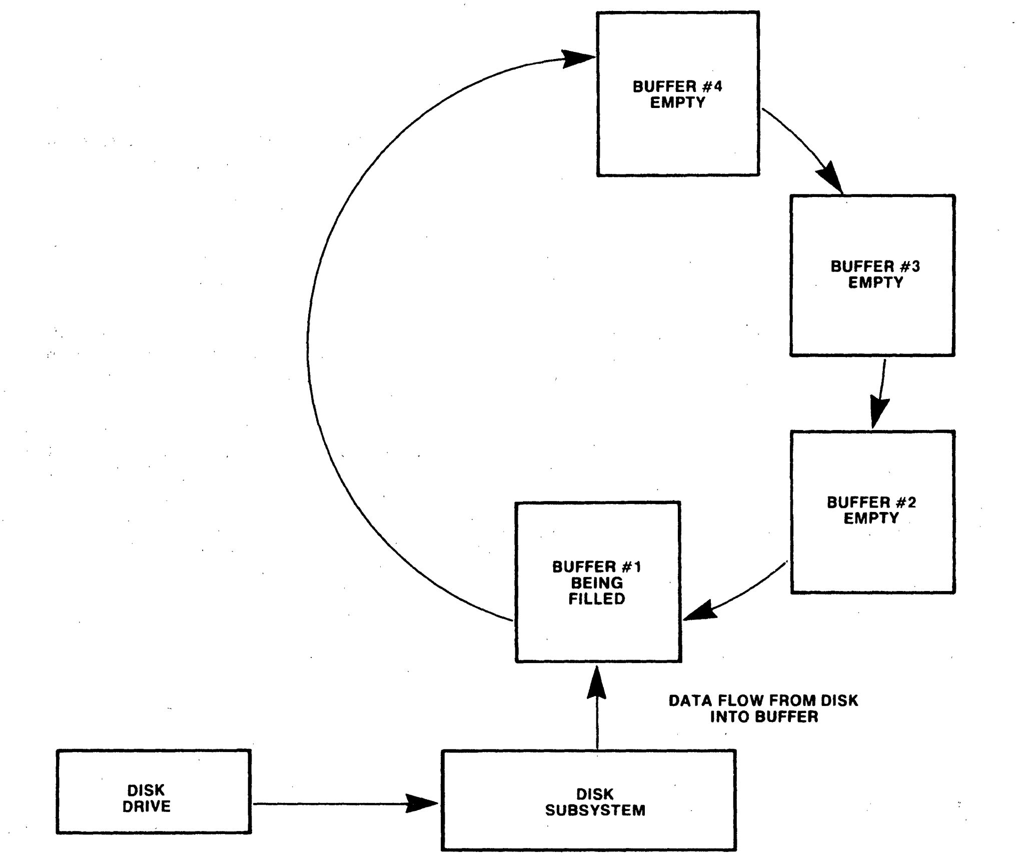
The DMA mode has the advantage of permitting the processor to continue executing instructions while a disk transfer is in progress. (This capability is especially useful in multiprogramming environments when the operating system is designed to permit other tasks to execute while a program is waiting for I/O.) Modes 2 and 3 are often combined and described as non-DMA operating modes. Non-DMA modes have the advantage of significantly lower system cost, but are often performance limited for double-density systems (where data bytes must be transferred to/from the FDC every 16 microseconds).

Overlapped Operations

Some FDC devices support simultaneous disk operations on more than one disk drive. Normally seek and recalibrate operations can be overlapped in this manner. Since seek operations on most floppy drives are extremely slow, this mode of operation can often be used by the system software to reduce overall disk access times.

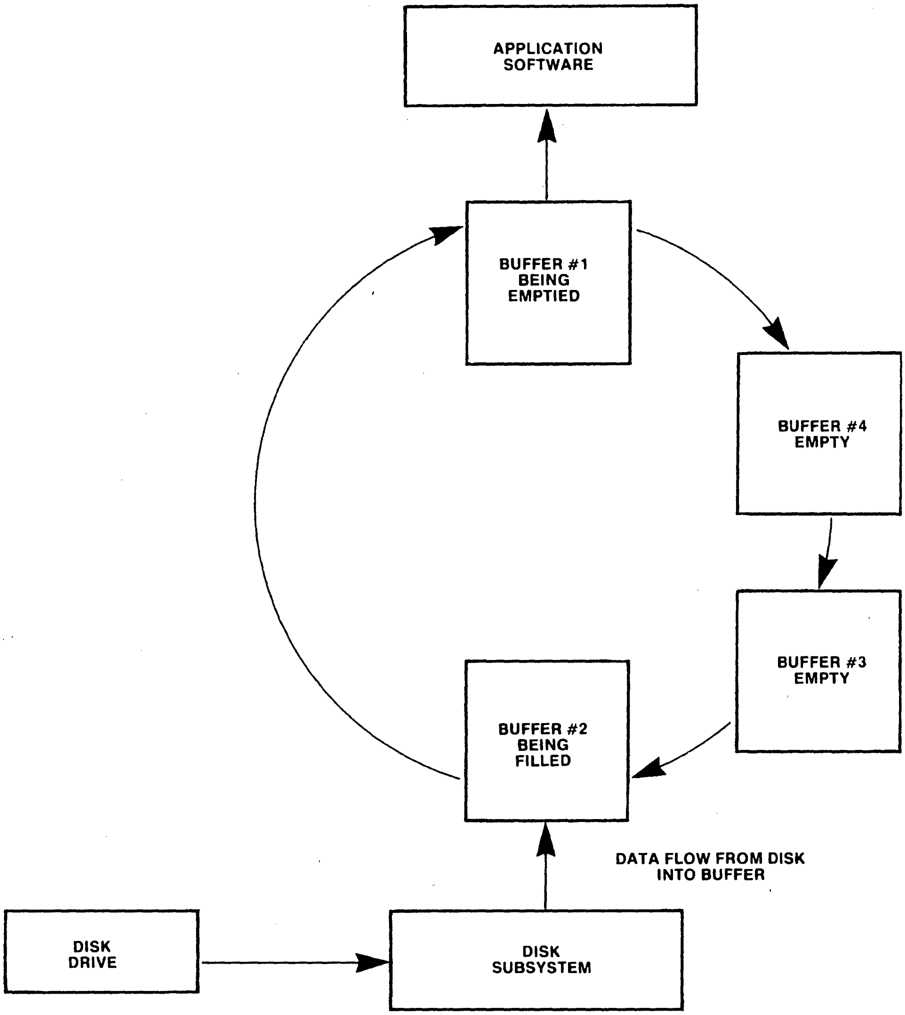
Buffers

The buffer concept is an extremely important element in advanced disk I/O strategies. A buffer is nothing more than a memory area containing the same amount of data as a disk sector contains. Generally, when an application program requests data from a disk, the system software allocates a buffer (memory area) and transfers the data from the appropriate disk sector into the buffer. The address of the buffer is then returned to the application software. In the same manner, after the application program has filled a buffer for output, the buffer address is passed to the system software, which writes data from the buffer into a disk sector. In multitasking systems, multiple buffers may be allocated from a buffer pool. In these systems, the disk controller is often requested to read ahead and fill additional data buffers while the application software is processing a previous buffer. Using this technique, system software attempts to fill buffers before they are needed by the application programs, thereby eliminating program waits during I/O transfers. Figure 1 illustrates the use of multiple buffers in a ring configuration.



NOTE:
a) The first disk read request by the application software causes the disk subsystem to begin filling the first empty buffer. The application software must wait until the buffer is filled before it may continue execution.

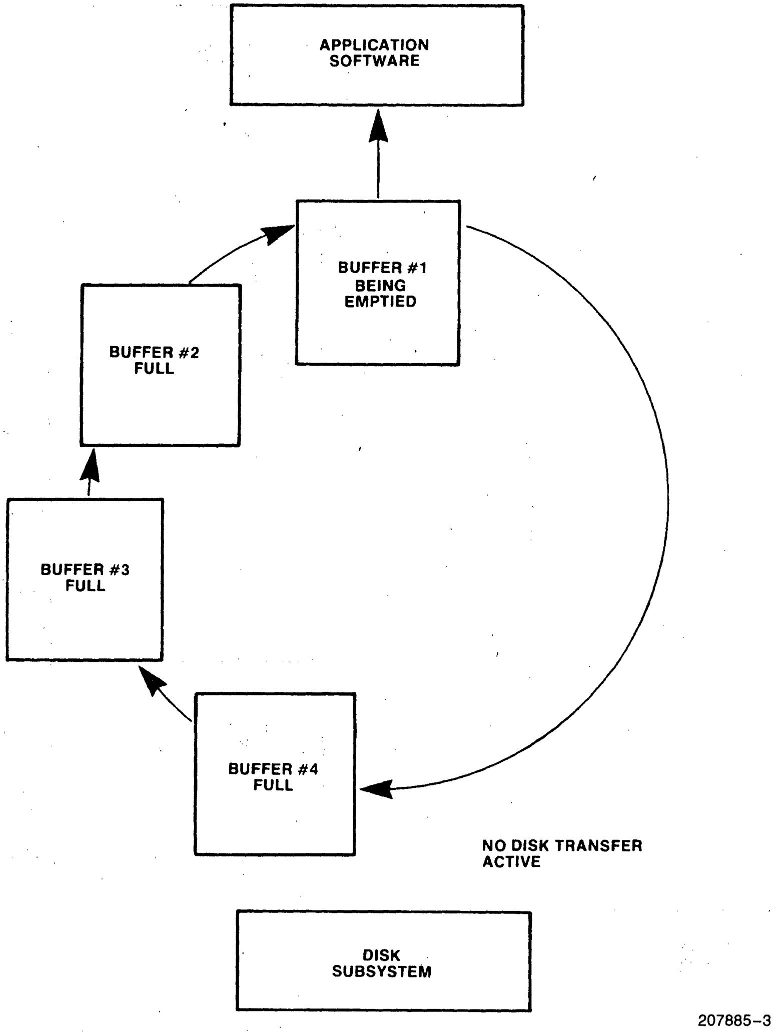
Figure 1. Using Multiple Memory Buffers for Disk I/O



207885–2

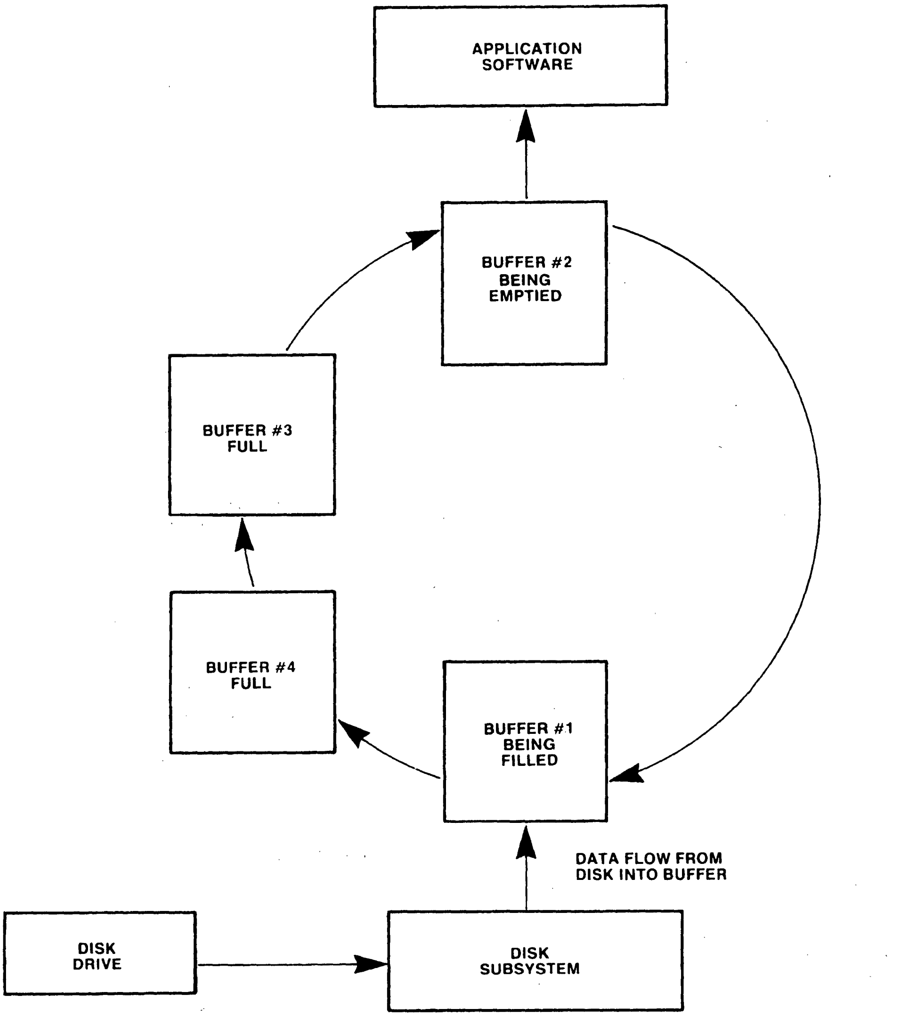
NOTE:
b) After the first buffer is filled, the disk system continues to transfer disk data into the next buffer while the application software begins operating on the first full buffer.

Figure 1. Using Multiple Memory Buffers for Disk I/O (Continued)



NOTE:
c) When all empty buffers have been filled, disk activity is stopped until the application software releases one or more buffers for reuse.

Figure 1. Using Multiple Memory Buffers for Disk I/O (Continued)



NOTE:
d) When the application software releases a buffer (for reuse), the disk subsystem begins a disk sector read to refill the buffer. This strategy attempts to anticipate application software needs by maintaining a sufficient number of full data buffers in order to minimize data transfer delays. If disk data is already in memory when the application software requests it, no disk transfer delays are incurred.

Figure 1. Using Multiple Memory Buffers for Disk I/O (Continued)

3.0 THE 8272 FLOPPY DISK CONTROLLER

The 8272 is a single-chip LSI Floppy Disk Controller (FDC) that implements both single- and double-density floppy disk storage subsystems (with up to four dual-sided disk drives per FDC). The 8272 supports the IBM 3740 single-density recording format (FM) and the IBM System 34 double-density recording format (MFM). The 8272 accepts and executes high-level disk commands such as format track, seek, read sector, and write sector. All data synchronization and error checking is automatically performed by the FDC to ensure reliable data storage and subsequent retrieval. The 8272 interfaces to microprocessor systems with or without Direct Memory Access (DMA) capabilities and also interfaces to a large number of commercially available floppy disk drives.

Floppy Disk Commands

The 8272 executes fifteen high-level disk interface commands:

Specify
Sense Drive Status
Sense Interrupt Status
Seek
Recalibrate
Format Track
Read Data
Read Deleted Data
Write Data
Write Deleted Data
Read Track
Read ID
Scan Equal
Scan High or Equal
Scan Low or Equal

Each command is initiated by a multi-byte transfer from the driver software to the FDC (the transferred bytes contain command and parameter information). After complete command specification, the FDC automatically executes the command. The command result data (after execution of the command) may require a multi-byte transfer of status information back to the driver. It is convenient to consider each FDC command as consisting of the following three phases:

Command Phase: The driver transfers to the FDC all the information required to perform a particular disk operation. The 8272 automatically enters the command phase after RESET and following the completion of the result phase (if any) of a previous command.

Execution Phase: The FDC performs the operation as instructed. The execution phase is entered immediately after the last command parameter is written to the FDC in the preceding command phase. The execution phase normally ends when the last data byte is transferred to/from the disk or when an error occurs.

Result Phase: After completion of the disk operation, status and other housekeeping information are made available to the driver software. After this information is read, the FDC reenters the command phase and is ready to accept another command.

Interface Registers

To support information transfer between the FDC and the system software, the 8272 contains two 8-bit registers: the Main Status Register and the Data Register. The Main Status Register (read only) contains FCD status information and may be accessed at any time. The Main Status Register (Table 3) provides the system processor with the status of each disk drive, the status of the FDC, and the status of the processor interface. The Data Register (read/write) stores data, commands, parameters, and disk drive status information. The Data Register is used to program the FDC during the command phase and to obtain result information after completion of FDC operations.

In addition to the Main Status Register, the FDC contains four additional status registers (ST0, ST1, ST2, and ST3). These registers are only available during the result phase of a command.

Command/Result Phases

Table 4 lists the 8272 command set. For each of the fifteen commands, command and result phase data transfers are listed. A list of abbreviations used in the table is given in Table 5, and the contents of the result status registers (ST0–ST3) are illustrated in Table 6.

The bytes of data which are sent to the 8272 by the drivers during the command phase, and are read out of the 8272 in the result phase, must occur in the order shown in Table 4. That is, the command code must be sent first and the other bytes sent in the prescribed sequence. All bytes of the command and result phases must be read/written as described. After the last byte of data in the command phase is sent to the 8272 the execution phase automatically starts. In a similar fashion, when the last byte of data is read from the 8272 in the result phase, the result phase is automatically ended and the 8272 reenters the command phase.

It is important to note that during the result phase all bytes shown in Table 4 must be read. The Read Data command, for example, has seven bytes of data in the result phase. All seven bytes must be read in order to successfully complete the Read Data command. The 8272 will not accept a new command until all seven bytes have been read. The number of command and result bytes varies from command-to-command.

In order to read data from, or write data to, the Data Register during the command and result phases, the software driver must examine the Main Status Register to determine if the Data Register is available. The DIO (bit 6) and RQM (bit 7) flags in the Main Status Register must be low and high, respectively, before each byte of the command word may be written into the 8272. Many of the commands require multiple bytes, and as a result, the Main Status Register must be read prior to each byte transfer to the 8272. To read status bytes during the result phase, DIO and RQM in the Main Status Register must both be high. Note, checking the Main Status Register in this manner before each byte transfer to/from the 8272 is required only in the command and result phases, and is NOT required during the execution phase.

Table 3. Main Status Register Bit Definitions

Bit Number	Symbol	Description
0	D_0B	Disk Drive 0 Busy. Disk Drive 0 is seeking.
1	D_1B	Disk Drive 1 Busy. Disk Drive 1 is seeking.
2	D_2B	Disk Drive 2 Busy. Disk Drive 2 is seeking.
3	D_3B	Disk Drive 3 Busy. Disk Drive 3 is seeking.
4	CB	FDC Busy. A read or write command is in progress.
5	NDM	Non-DMA Mode. The FDC is in the non-DMA mode when this flag is set (1). This flag is set only during the execution phase of commands in the non-DMA mode. Transition of this flag to a zero (0) indicates that the execution phase has ended.
6	DIO	Data Input/Output. Indicates the direction of a data transfer between the FDC and the Data Register. When DIO is set (1), data is read from the Data Register by the processor; when DIO is reset (0), data is written from the processor to the Data Register.
7	RQM	Request for Master. When set (1), this flag indicates that the Data Register is ready to send data to, or receive data from, the processor.

Table 4. 8272 Command Set

		DATA BUS								
PHASE	R/W	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	REMARKS
READ DATA										
Command	W	MT	MFM	SK	0	0	1	1	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior to Command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and the main-system
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information after command execution
	R	H								
	R	R								
	R	N								
READ DELETED DATA										
Command	W	MT	MFM	SK	0	1	1	0	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior to Command execution
	W	H								
	W	R								
	W	N								
	W	EC 1								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and the main-system
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information after Command execution
	R	H								
	R	R								
	R	N								
WRITE DATA										
Command	W	MT	MFM	0	0	0	1	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior to Command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the main-system and the FDD
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information after Command execution
	R	H								
	R	R								
	R	N								
WRITE DELETED DATA										
Command	W	MT	MFM	0	0	1	0	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior to Command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and the main-system
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information after Command execution
	R	H								
	R	R								
	R	N								

		DATA BUS								
PHASE	R/W	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	REMARKS
READ A TRACK										
Command	W	0	MFM	SK	0	0	0	1	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior to Command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data transfer between the FDD and the main-system. FDC reads the complete track contents from the physical index mark to EOT
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information after Command execution
	R	H								
	R	R								
	R	N								
READ ID										
Command	W	0	MFM	0	0	1	0	1	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
Execution										The first correct ID information on the track is stored in Data Register
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information during Execution Phase
	R	H								
	R	R								
	R	N								
FORMAT A TRACK										
Command	W	0	MFM	0	0	1	1	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	N								Bytes/Sector
	W	SC								Sectors/Track
	W	GPL								Gap 3
	W	D								Filler Byte
Execution										FDC formats an entire track
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								In this case, the ID information has no meaning
	R	H								
	R	R								
	R	N								
SCAN EQUAL										
Command	W	MT	MFM	SK	1	0	0	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior to Command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	STP								
Execution										Data compared between the FDD and the main-system
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information after Command execution
	R	H								
	R	R								
	R	N								

207885-34

NOTE:
1. A_0 = 1 for all operations.

Table 4. 8272 Command Set (Continued)

PHASE	R/W	DATA BUS								REMARKS
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
SCAN LOW OR EQUAL										
Command	W	MT	MFM	SK	1	1	0	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior Command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	STP								
Execution										Data compared between the FDD and the main-system
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information after Command execution
	R	H								
	R	R								
	R	N								
SCAN HIGH OR EQUAL										
Command	W	MT	MFM	SK	1	1	1	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID information prior Command execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	STP								
Execution										Data compared between the FDD and the main-system
Result	R	ST 0								Status information after Command execution
	R	ST 1								
	R	ST 2								
	R	C								Sector ID information after Command execution
	R	H								
	R	R								
	R	N								

PHASE	R/W	DATA BUS								REMARKS
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
RECALIBRATE										
Command	W	0	0	0	0	0	1	1	1	Command Codes
	W	0	0	0	0	0	0	DS1	DS0	
Execution										Head retracted to Track 0
SENSE INTERRUPT STATUS										
Command	W	0	0	0	0	1	0	0	0	Command Codes
Result	R	ST 0								Status information at the end of each seek operation about the FDC
	R	C								
SPECIFY										
Command	W	0	0	0	0	0	0	1	1	Command Codes
	W	SPT				HUT				Timer Settings
	W	HLT							ND	
SENSE DRIVE STATUS										
Command	W	0	0	0	0	0	1	0	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
Result	R	ST 3								Status information about the FDD
SEEK										
Command	W	0	0	0	0	1	1	1	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								
Execution										Head is positioned over proper Cylinder on Diskette
INVALID										
Command	W	Invalid Codes								Invalid Command Codes (NoOp— FDC goes into Standby State)
Result	R	ST 0								ST 0 = 80 (16)

207885-35

NOTE:
1. A_0 = 1 for all operations.

Table 5. Command/Result Parameter Abbreviations

Symbol	Description
C	**CYLINDER ADDRESS.** The currently selected cylinder address (0 to 76) on the disk.
D	**DATA PATTERN.** The pattern to be written in each sector data field during formatting.
DS0, DS1	**DISK DRIVE SELECT.** **DS1 DS0** 0 0 Drive 0 0 1 Drive 1 1 0 Drive 2 1 1 Drive 3
DTL	**SPECIAL SECTOR SIZE.** During the execution of disk read/write commands, this parameter is used to temporarily alter the effective disk sector size. By setting N to zero, DTL may be used to specify a sector size from 1 to 256 bytes in length. If the actual sector (on the disk) is larger than DTL specifies, the remainder of the actual sector is not passed to the system during read commands; during write commands, the remainder of the actual sector is written with all-zeroes bytes. DTL should be set to FF hexadecimal when N is not zero.
EOT	**END OF TRACK.** The final sector number of the current track.
GPL	**GAP LENGTH.** The gap 3 size. (Gap 3 is the space between sectors).
H	**HEAD ADDRESS.** Selected head: 0 or 1 (disk side 0 or 1, respectively) as encoded in the sector ID field.
HLT	**HEAD LOAD TIME.** Defines the time interval that the FDC waits after loading the head before initiating the read or write operation. Programmable from 2 to 254 milliseconds (in increments of 2 ms).
HUT	**HEAD UNLOAD TIME.** Defines the time interval from the end of the execution phase (of a read or write command) until the head is unloaded. Programmable from 16 to 240 milliseconds (in increments of 16 ms).
MFM	**MFM/FM MODE SELECTOR.** Selects MFM double-density recording mode when high, FM single-density mode when low.
MT	**MULTI-TRACK SELECTOR.** When set, this flag selects the multi-track operating mode. In this mode (used only with dual-sided disks), the FDC treats a complete cylinder (under both read/write head 0 and read/write head 1) as a single track. The FDC operates as if this expanded track started at the first sector under head 0 and ended at the last sector under head 1. With this flag set (high), a multi-sector read operation will automatically continue to the first sector under head 1 when the FDC finishes operating on the last sector under head 0.
N	**SECTOR SIZE CODE.** The number of data bytes within a sector.
ND	**NON-DMA MODE FLAG.** When set (1), this flag inidcates that the FDC is to operate in the non-DMA mode. In this mode, the processor participates in each data transfer (by means of an interrupt or by polling the RQM flag in the Main Status Register). When reset (0), the FDC interfaces to a DMA controller.
R	**SECTOR ADDRESS.** Specifies the sector number to be read or written. In multi-sector transfers, this parameter specifies the sector number of the first sector to be read or written.
SC	**NUMBER OF SECTORS PER TRACK.** Specifies the number of sectors per track to be initialized by the Format Track command.
SK	**SKIP FLAG.** When this flag is set, sectors containing deleted data address marks will automatically be skipped during the execution of multi-sector Read Data or Scan commands. In the same manner, a sector containing a data address mark will automatically be skipped during the execution of a multi-sector Read Deleted Data command.
SRT	**STEP RATE INTERVAL.** Defines the time interval between step pulses issued by the FDC (track-to-track access time). Programmable from 1 to 16 milliseconds (in increments of 1 ms).
ST0, ST1, ST2, ST3	**STATUS REGISTER 0–3.** Registers within the FDC that store status information after a command has been executed. This status information is available to the processor during the Result Phase after command execution. These registers may only be read after a command has been executed (in the exact order shown in Table 4 for each command). These registers should not be confused with the Main Status Register.
STP	**SCAN SECTOR INCREMENT.** During Scan operations, this parameter is added to the current sector number in order to determine the next sector to be scanned.

Table 6. Status Register Definitions

Bit Number	Symbol	Description
STATUS REGISTER 0		
7, 6	IC	**INTERRUPT CODE.** 00— Normal termination of command. The specified command was properly executed and completed without error. 01— Abnormal termination of command. Command execution was started but could not be successfully completed. 10— Invalid command. The requested command could not be executed. 11— Abnormal termination. During command execution, the disk drive ready signal changed state.
5	SE	**SEEK END.** This flag is set (1) when the FDC has completed the Seek command and the read/write head is positioned over the correct cylinder.
4	EC	**EQUIPMENT CHECK ERROR.** This flag is set (1) if a fault signal is received from the disk drive or if the track 0 signal is not received from the disk drive after 77 step pulses (Recalibrate command).
3	NR	**NOT READY ERROR.** This flag is set if a read or write command is issued and either the drive is not ready or the command specifies side 1 (head 1) of a single-sided disk.
2	H	**HEAD ADDRESS.** The head address at the time of the interrupt.
1, 0	DS1, DS0	**DRIVE SELECT.** The number of the drive selected at the time of the interrupt.
STATUS REGISTER 1		
7	EN	**END OF TRACK ERROR.** This flag is set if the FDC attempts to access a sector beyond the final sector of the track.
6		**UNDEFINED**
5	DE	**DATA ERROR.** Set when the FDC detects a CRC error in either the ID field or the data field of a sector.
4	OR	**OVERRUN ERROR.** Set (during data transfers) if the FDC does not receive DMA or processor service within the specified time interval.
3		**UNDEFINED**
2	ND	**SECTOR NOT FOUND ERROR.** This flag is set by any of the following conditions. a) The FDC cannot locate the sector specified in the Read Data, Read Deleted Data, or Scan command. b) The FDC cannot locate the starting sector specified in the Read Track command. c) The FDC cannot read the ID field without error during a Read ID command.
1	NW	**WRITE PROTECT ERROR.** This flag is set if the FDC detects a write protect signal from the disk drive during the execution of a Write Data, Write Deleted Data, or Format Track command.
0	MA	**MISSING ADDRESS MARK ERROR.** This flag is set by either of the following conditions: a) The FDC cannot detect the ID address mark on the specified track (after two rotations of the disk). b) The FDC cannot detect the data address mark or deleted data address mark on the specified track. (See also the MD bit of Status Register 2.)

Table 6. Status Register Definitions (Continued)

Bit Number	Symbol	Description
STATUS REGISTER 2		
7		**UNDEFINED**
6	CM	**CONTROL MARK.** This flag is set when the FDC encounters one of the following conditions: a) A deleted data address mark during the execution of a Read Data or Scan command. b) A data address mark during the execution of a Read Deleted Data command.
5	DD	**DATA ERROR.** Set (1) when the FDC detects a CRC error in a sector data field. This flag is not set when a CRC error is detected in the ID field.
4	WC	**CYLINDER ADDRESS ERROR.** Set when the cylinder address from the disk sector ID field is different from the current cylinder address maintained within the FDC.
3	SH	**SCAN HIT.** Set during the execution of the Scan command if the scan condition is satisfied.
2	SN	**SCAN NOT SATISFIED.** Set during execution of the Scan command if the FDC cannot locate a sector on the specified cylinder that satisfies the scan condition.
1	BC	**BAD TRACK ERROR.** Set when the cylinder address from the disk sector ID field is FF hexadecimal and this cylinder address is different from the current cylinder address maintained within the FDC. This all "ones" cylinder number indicates a bad track (one containing hard errors) according to the IBM soft-sectored format specifications.
0	MD	**MISSING DATA ADDRESS MARK ERROR.** Set if the FDC cannot detect a data address mark or deleted data address mark on the specified track.
STATUS REGISTER 3		
7	FT	**FAULT.** This flag indicates the status of the fault signal from the selected disk drive.
6	WP	**WRITE PROTECTED.** This flag indicates the status of the write protect signal from the selected disk drive.
5	RDY	**READY.** This flag indicates the status of the ready signal from the selected disk drive.
4	T0	**TRACK 0.** This flag indicates the status of the track 0 signal from the selected disk drive.
3	TS	**TWO-SIDED.** This flag indicates the status of the two-sided signal from the selected disk drive.
2	H	**HEAD ADDRESS.** This flag indicates the status of the side select signal for the currently selected disk drive.
1, 0	DS1, DS0	**DRIVE SELECT.** Indicates the currently selected disk drive number.

Execution Phase

All data transfers to (or from) the floppy drive occur during the execution phase. The 8272 has two primary modes of operation for data transfers (selected by the specify command):

1) DMA mode

2) non-DMA mode

In the DMA mode, execution phase data transfers are handled by the DMA controller hardware (invisible to the driver software). The driver software, however, must set all appropriate DMA controller registers prior to the beginning of the disk operation. An interrupt is generated by the 8272 after the last data transfer, indicating the completion of the execution phase, and the beginning of the result phase.

In the non-DMA mode, transfer requests are indicated by generation of an interrupt and by activation of the RQM flag (bit 7 in the Main Status Register). The interrupt signal can be used for interrupt-driven systems and RQM can be used for polled systems. The driver software must respond to the transfer request by reading data from, or writing data to, the FDC. After completing the last transfer, the 8272 generates an interrupt to indicate the beginning of the result phase. In the non-DMA mode, the processor must activate the "terminal count" (TC) signal to the FDC (normally by means of an I/O port) after the transfer request for the last data byte has been received (by the driver) and before the appropriate data byte has been read from (or written to) the FDC.

In either mode of operation (DMA or non-DMA), the execution phase ends when a "terminal count" signal is sensed by the FDC, when the last sector on a track (the EOT parameter—Table 4) has been read or written, or when an error occurs.

Multi-Sector and Multi-Track Transfers

During disk read/write transfers (Read Data, Write Data, Read Deleted Data, and Write Deleted Data), the FDC will continue to transfer data from sequential sectors until the TC input is sensed. In the DMA mode, the TC input is normally set by the DMA controller. In the non-DMA mode, the processor directly controls the FDC TC input as previously described. Once the TC input is received, the FDC stops requesting data transfers (from the system software or DMA controller). The FDC, however, continues to read data from, or write data to, the floppy disk until the end of the current disk sector. During a disk read operation, the data read from the disk (after reception of the TC input) is discarded, but the data CRC is checked for errors; during a disk write operation, the remainder of the sector is filled with all-zero bytes.

If the TC signal is not received before the last byte of the current sector has been transferred to/from the system, the FDC increments the sector number by one and initiates a read or write command for this new disk sector.

The FDC is also designed to operate in a multi-track mode for dual-sided disks. In the multi-track mode (specified by means of the MT flag in the command byte—Table 4) the FDC will automatically increment the head address (from 0 to 1) when the last sector (on the track under head 0) has been read or written. Reading or writing is then continued on the first sector (sector 1) of head 1.

Drive Status Polling

After the power-on reset, the 8272 automatically enters a drive status polling mode. If a change in drive status is detected (all drives are assumed to be "not ready" at power-on), an interrupt is generated. The 8272 continues this status polling between command executions (and between step pulses in the Seek command). In this manner, the 8272 automatically notifies the system software whenever a floppy disk is inserted, removed, or changed by the operator.

Command Details

During the command phase, the Main Status Register must be polled by the driver software before each byte is written into the Data Register. The DIO (bit 6) and RQM (bit 7) flags in the Main Status Register must be low and high, respectively, before each byte of the command may be written into the 8272. The beginning of the execution phase for any of these commands will cause DIO to be set high and RQM to be set low.

Operation of the FDC commands is described in detail in Application Note AP-116, "An Intelligent Data Base System Using the 8272".

Invalid Commands

If an invalid (undefined) command is sent to the FDC, the FDC will terminate the command. No interrupt is generated by the 8272 during this condition. Bit 6 and bit 7 (DIO and RQM) in the Main Status Register are both set indicating to the processor that the 8272 is in the result phase and the contents of Status Register 0 must be read. When the processor reads Status Register 0 it will find an 80H code indicating that an invalid command was received. The driver software in Appendix B checks each requested command and will not issue an invalid command to the 8272.

A Sense Interrupt Status command must be sent after a Seek or Recalibrate interrupt; otherwise the FDC will consider the next command to be an invalid command. Also, when the last "hidden" interrupt has been serviced, further Sense Interrupt Status commands will result in invalid command codes.

4.0 8272 PHYSICAL INTERFACE SOFTWARE

PL/M software driver listings for the 8272 FDC are contained in Appendix A. These drivers have been designed to operate in a DMA environment (as described in Application Note AP-116, "An Intelligent Data Base System Using the 8272"). In the following paragraphs, each driver procedure is described. (A description of the driver data base variables is given in Table 7.) In addition, the modifications necessary to reconfigure the drivers for operation in a polled environment are discussed.

INITIALIZE$DRIVERS

This initialization procedure must be called before any FDC operations are attempted. This module initializes the DRIVE$READY, DRIVE$STATUS$CHANGE, OPERATION$IN$PROGRESS, and OPERATION$COMPLETE arrays as well as the GLOBAL$DRIVE$NO variable.

EXECUTE$DOCB

This procedure contains the main 8272 driver control software and handles the execution of a complete FDC command. EXECUTE$DOCB is called with two parameters: a) a pointer to a disk operation control block and b) a pointer to a result status byte. The format of the disk operation control block is illustrated in Figure 2 and the result status codes are described in Table 8.

Before starting the command phase for the specified disk operation, the command is checked for validity and to determine whether the FDC is busy. (For an overlapped operation, if the FDC BUSY flag is set—in the Main Status Register—the command cannot be started; non-overlapped operations cannot be started if the FDC BUSY flag is set, if any drive is in the process of seeking/recalibrating, or if an operation is currently in progress on the specified drive.)

After these checks are made, interrupts are disabled in order to set the OPERATION$IN$PROGRESS flag, reset the OPERATION$COMPLETE flag, load a pointer to the current operation control block into the OPERATION$DOCB$PTR array and set GLOBAL$DRIVE$NO (if a non-overlapped operation is to be started).

At this point, parameters from the operation control block are output to the DMA controller and the FDC command phase is initiated. After completion of the command phase, a test is made to determine the type of result phase required for the current operation. If no result phase is needed, control is immediately returned to the calling program. If an immediate result phase is required, the result bytes are input from the FDC. Otherwise, the CPU waits until the OPERATION$COMPLETE flag is set (by the interrupt service procedure).

Finally, if an error is detected in the result status code (from the FDC), an FDC operation error is reported to the calling program.

Table 7. Driver Data Base

Name	Description
DRIVE$READY	A public array containing the current "ready" status of each drive.
DRIVE$STATUS$CHANGE	A public array containing a flag for each drive. The appropriate flag is set whenever the ready status of a drive changes.
OPERATION$DOCB$PTR	An internal array of pointers to the operation control block currently in progress for each drive.
OPERATION$IN$PROGRESS	An internal array used by the driver procedures to determine if a disk operation is in progress on a given drive.
OPERATION$COMPLETE	An internal array used by the driver procedures to determine when the execution phase of a disk operation is complete.
GLOBAL$DRIVE$NO	A data byte that records the current drive number for non-overlapped disk operations.
VALID$COMMAND	A constant flag array that indicates whether a specified FDC command code is valid.
COMMAND$LENGTH	A constant byte array specifying the number of command/parameter bytes to be transferred to the FDC during the command phase.
DRIVE$NO$PRESENT	A constant flag array that indicates whether a drive number is encoded into an FDC command.
OVERLAP$OPERATION	A constant flag array that indicates whether an FDC command can be overlapped with other commands.
NO$RESULT	A constant flag array that is used to determine when an FDC operation does not have a result phase.
IMMED$RESULT	A constant flag array that indicates that an FDC operation has a result phase beginning immediately after the command phase is complete.
POSSIBLE$ERROR	A constant flag array that indicates if an FDC operation should be checked for an error status indication during the result phase.

Address Offset	Disk Operation Control Block (DOCB)
0	DMA$OP
1	DMA$ADDR
3	DMA$ADDR$EXT
4	DMA$COUNT
6	DISK$COMMAND (0)
7	DISK$COMMAND (1)
8	DISK$COMMAND (2)
9	DISK$COMMAND (3)
10	DISK$COMMAND (4)
11	DISK$COMMAND (5)
12	DISK$COMMAND (6)
13	DISK$COMMAND (7)
14	DISK$COMMAND (8)
15	DISK$RESULT (0)
16	DISK$RESULT (1)
17	DISK$RESULT (2)
18	DISK$RESULT (3)
19	DISK$RESULT (4)
20	DISK$RESULT (5)
21	DISK$RESULT (6)
22	MISC

Figure 2. Disk Operation Control Block (DOCB) Format

Table 8. EXECUTE$DOCB Return Status Codes

Code	Description
0	**NO ERRORS.** The specified operation was completed without error.
1	**FDC BUSY.** The requested operation cannot be started. This error occurs if an attempt is made to start an operation before the previous operation is completed.
2	**FDC ERROR.** An error was detected by the FDC during the execution phase of a disk operation. Additional error information is contained in the result data portion of the disk operation control block (DOCB.DISK$RESULT) as described in the 8272 data sheet. This error occurs whenever the 8272 reports an execution phase error (e.g., missing address mark).
3	**8272 COMMAND INTERFACE ERROR.** An 8272 interfacing error was detected during the command phase. This error occurs when the command phase of a disk operation cannot be successfully completed (e.g., incorrect setting of the DIO flag in the Main Status Register).
4	**8272 RESULT INTERFACE ERROR.** An 8272 interfacing error was detected during the result phase. This error occurs when the result phase of a disk operation cannot be successfully completed (e.g., incorrect setting of the DIO flag in the Main Status Register).
5	**INVALID FDC COMMAND.**

FDCINT

This procedure performs all interrupt processing for the 8272 interface drivers. Basically, two types of interrupts are generated by the 8272: (a) an interrupt that signals the end of a command execution phase and the beginning of the result phase and (b) an interrupt that signals the completion of an overlapped operation or the occurrence of an unexpected event (e.g., change in the drive "ready" status).

An interrupt of type (a) is indicated when the FDC BUSY flag is set (in the Main Status Register). When a type (a) interrupt is sensed, the result bytes are read from the 8272 and placed in the result portion of the disk operation control block, the appropriate OPERATION$COMPLETE flag is set, and the OPERATION$IN$PROGRESS flag is reset.

When an interrupt of type (b) is indicated (FDC not busy), a sense interrupt status command is issued (to the FDC). The upper two bits of the result status register (Status Register Zero—ST0) are used to determine the cause of the interrupt. The following four cases are possible:

1) Operation Complete. An overlapped operation is complete. The drive number is found in the lower two bits of ST0. The ST0 data is transferred to the active operation control block, the OPERATION$COMPLETE flag is set, and the OPERATION$IN$PROGRESS flag is reset.
2) Abnormal Termination. A disk operation has abnormally terminated. The drive number is found in the lower two bits of ST0. The ST0 data is transferred to the active control block, the OPERATION$COMPLETE flag is set, and the OPERATION$IN$PROGRESS flag is reset.
3) Invalid Command. The execution of an invalid command (i.e., a sense interrupt command with no interrupt pending) has been attempted. This interrupt signals the successful completion of all interrupt processing.
4) Drive Status Change. A change has occurred in the "ready" status of a disk drive. The drive number is found in the lower two bits of ST0. The DRIVE$READY flag for this disk drive is set to the new drive "ready" status and the DRIVE$STATUS$CHANGE flag for the drive is also set. In addition, if a command is currently in progress, the ST0 data is transferred to the active control block, the OPERATION$COMPLETE flag is set, and the OPERATION$IN$PROGRESS flag is reset.

After processing a type (b) interrupt, additional sense interrupt status commands must be issued and processed until an "invalid command" result is returned from the FDC. This action guarantees that all "hidden" interrupts are serviced.

In addition to the major driver procedures described above, a number of support procedures are required. These support routines are briefly described in the following paragraphs.

OUTPUT$CONTROLS$TO$DMA

This procedure outputs the DMA mode, the DMA address, and the DMA word count to the 8237 DMA controller. In addition, the upper four bits of the 20-bit DMA address are output to the address extension latch. Finally, the disk DMA channel is started.

OUTPUT$COMMAND$TO$FDC

This software module outputs a complete disk command to the 8272 FDC. The number of required command/parameter bytes is found in the COMMAND$LENGTH table. The appropriate bytes are output one at a time (by calls to OUTPUT$BYTE$TO$FDC) from the command portion of the disk operation control block.

INPUT$RESULT$FROM$FDC

This procedure is used to read result phase status information from the disk controller. At most, seven bytes are read. In order to read each byte, a call is made to INPUT$BYTE$FROM$FDC. When the last byte has been read, a check is made to insure that the FDC is no longer busy.

OUTPUT$BYTE$TO$FDC

This software is used to output a single command/parameter byte to the FDC. This procedure waits until the FDC is ready for a command byte and then outputs the byte to the FDC data port.

INPUT$BYTE$FROM$FDC

This procedure inputs a single result byte from the FDC. The software waits until the FDC is ready to transfer a result byte and then reads the byte from the FDC data port.

FDC$READY$FOR$COMMAND

This procedure assures that the FDC is ready to accept a command/parameter byte by performing the following three steps. First, a small time interval (more than 20 microseconds) is inserted to assure that the RQM flag has time to become valid (after the last byte transfer). Second, the master request flag (RQM) is polled until it is activated by the FDC. Finally, the DIO flag is checked to ensure that it is properly set for FDC input (from the processor).

FDC$READY$FOR$RESULT

The operation of this procedure is similar to the FDC$READY$FOR$COMMAND with the following exception. If the FDC BUSY flag (in the Main Status Register) is not set, the result phase is complete and no more data is available from the FDC. Otherwise, the procedure waits for the RQM flag and checks the DIO flag for FDC output (to the processor).

OPERATION$CLEAN$UP

This procedure is called after the execution of a disk operation that has no result phase. OPERATION$CLEAN$UP resets the OPERATION$IN$PROGRESS flag and the GLOBAL$DRIVE$NO variable if appropriate. This procedure is also called to clean up after some disk operation errors.

Modifications for Polling Operation

To operate in the polling mode, the following modifications should be made to the previous routines:

1) The OUTPUT$CONTROLS$TO$DMA routine should be deleted.

2) In EXECUTE$DOCB, immediately prior to WAIT$FOR$OP$COMPLETE, a polling loop should be inserted into the code. The loop should test the RQM flag (in the Main Status Register). When RQM is set, a data byte should be written to, or read from, the 8272. The buffer address may be computed from the base address contained in DOCB.DMA$ADDR and DOCB.DMA$ADDR$EXT. After the correct number of bytes have been transferred, an operation complete interrupt will be issued by the FDC. During data transfer in the non-DMA mode, the NON-DMA MODE flag (bit 5 of the Main Status Register) will be set. This flag will remain set for the complete execution phase. When the transfer is finished, the NON-DMA MODE flag is reset and the result phase interrupt is issued by the FDC.

5.0 8272 LOGICAL INTERFACE SOFTWARE

Appendix B of this Application Note contains a PL/M listing of an exerciser program for the 8272 drivers. This program illustrates the design of logical interface level procedures to specify disk parameters, recalibrate a drive, seek to a cylinder, format a disk, read data, and write data.

The exerciser program is written to operate a standard single-sided 8" floppy disk drive in either the single- or double-density recording mode. Only the eight parameters listed in Table 9 must be specified. All other parameters are derived from these 8 basic variables.

Each of these logical interface procedures is described in the following paragraphs (refer to the listing in Appendix B).

SPECIFY

This procedure sets the FDC signal timing so that the FDC will interface correctly to the attached disk drive. The SPECIFY procedure requires four parameters, the step rate (SRT), head load time (HLT), head unload time (HUT), and the non-DMA mode flag (ND). This procedure builds a disk operation control block (SPECIFY$DOCB) and passes the control block to the FDC driver module (EXECUTE$DOCB) for execution. (Note carefully the computation required to transform the step rate (SRT) into the correct 8272 parameter byte.)

RECALIBRATE

This procedure causes the floppy disk read/write head to retract to track 0. The RECALIBRATE procedure requires only one parameter—the drive number on which the recalibrate operation is to be performed. This procedure builds a disk operation control block (RECALIBRATE$DOCB) and passes the control block to the FDC driver for execution.

SEEK

This procedure causes the disk read/write head (on the selected drive) to move to the desired cylinder position. The SEEK procedure is called with three parameters: drive number (DRV), head/side number (HD), and cylinder number (CYL). This software module builds a disk operation control block (SEEK$DOCB) that is executed by the FDC driver.

FORMAT

The FORMAT procedure is designed to initialize a complete floppy disk so that sectors can subsequently be read and written by system and application programs. Three parameters must be supplied to this procedure: the drive number (DRV), the recording density (DENS), and the interleave factor (INTLVE). The FORMAT procedure generates a data block (FMTBLK) and a disk operation control block (FORMAT$DOCB) for each track on the floppy disk (normally 77).

Table 9. Basic Disk Parameters

Name	Description
DENSITY	The recording mode (FM or MFM).
FILLER$BYTE	The data byte to be written in all sectors during formatting.
TRACKS$PER$DISK	The number of cylinders on the floppy disk.
BYTES$PER$SECTOR	The number of bytes in each disk sector. The exerciser accepts 128, 256, and 512 in FM mode, and 256, 512, and 1024 in MFM mode.
INTERLEAVE	The sector interleave factor for each disk track.
STEP$RATE	The disk drive step rate (1–16 milliseconds).
HEAD$LOAD$TIME	The disk drive head load time (2–254 milliseconds).
HEAD$UNLOAD$TIME	The head unload time (16–240 milliseconds).

The format data block specifies the four sector ID field parameters (cylinder, head, sector, and bytes per sector) for each sector on the track. The sector numbers need not be sequential; the interleave factor (INTLVE parameter) is used to compute the logical to physical sector mapping.

After both the format data block and the operation control block are generated for a given cylinder, control is passed to the 8272 drivers for execution. After the format operation is complete, a SEEK to the next cylinder is performed, a new format table is generated, and another track formatting operation is executed by the drivers. This track formatting continues until all tracks on the diskette are formatted.

In some systems, bad tracks must also be specified when a disk is formatted. For these systems, the existing FORMAT procedure should be modified to format bad tracks with a cylinder number of 0FFH.

Write

The WRITE procedure transfers a complete sector of data to the disk drive. Five parameters must be supplied to this software module: the drive number (DRV), the cylinder number (CYL), the head/side number (HD), the sector number (SEC) and the recording density (DENS). This procedure generates a disk operation control block (WRITE$DOCB) from these parameters and passes the control block to the 8272 driver for execution. When control returns to the calling program, the data has been transferred to disk.

Read

This procedure is identical to the WRITE procedure except the direction of data transfer is reversed. The READ procedure transfers a sector of data from the floppy disk to system memory.

Coping with Errors

In actual practice all logical disk interface routines would contain error processing mechanisms. (Errors have been ignored for the sake of simplicity in the exerciser programs listed in Appendix B.) A typical error recovery technique consists of a two-stage procedure. First, when an error is detected, a recalibrate operation is performed followed by a retry of the failed operation. This procedure forces the drive to seek directly to the requested cylinder (lowering the probability of a seek error) and attempts to perform the requested operation an additional time. Soft (temporary) errors caused by mechanical or electrical interference do not normally recur during the retry operation; hard errors (caused by media or drive failures), on the other hand, will continue to occur during retry operations. If, after a number of retries (approximately 10), the operation continues to fail, an error message is displayed to the system operator. This error message lists the drive number, type of operation, and failure status (from the FDC). It is the operator's responsibility to take additional action as required.

6.0 FILE SYSTEMS

The file system provides the disk I/O interface level most familiar to users of interactive microcomputer and minicomputer systems. In a file system, all data is stored in named disk areas called files. The user and applications programs need not be concerned with the exact location of a file on the disk—the disk file system automatically determines the file location from the file name. Files may be created, read, written, modified, and finally deleted (destroyed) when they are no longer needed. Each floppy disk typically contains a directory that lists all the files existing on the disk. A directory entry for a file contains information such as file name, file size, and the disk address (track and sector) of the beginning of the file.

File Allocation

File storage is actually allocated on the disk (by the file system) in fixed size areas called blocks. Normally a block is the same size as a disk sector. Files are created by finding and reserving enough unused blocks to contain the data in the file. Two file allocation methods are currently in widespread use. The first method allocates blocks (for a file) from a sequential pool of unused blocks. Thus, a file is always contained in a set of sequential blocks on the disk. Unfortunately, as files are created, updated, and deleted, these free-block pools become fragmented (separated from one another). When this fragmentation occurs, it often becomes impossible for the file system to create a file even though there is a sufficient number of free blocks on the disk. At this point, special programs must be run to "squeeze" or compact the disk, in order to re-create a single contiguous free-block pool.

The second file allocation method uses a more flexible technique in which individual data blocks may be located anywhere on the disk (with no restrictions). With this technique, a file directory entry contains the disk address of a file pointer block rather than the disk address of the first data block of the file. This file pointer block contains pointers (disk addresses) for each data block in the file. For example, the first pointer in the file pointer block contains the track and sector address of the first data block in the file; the second pointer contains the disk address of the second data block, etc.

In practice, pointer blocks are usually the same size as data blocks. Therefore, some files will require multiple pointer blocks. To accommodate this requirement without loss of flexibility, pointer blocks are linked together, that is, each pointer block contains the disk address of the following pointer block. The last pointer block of the file is signaled by an illegal disk address (e.g., track 0, sector 0 or track 0FFH, sector 0FFH).

The Intel File System

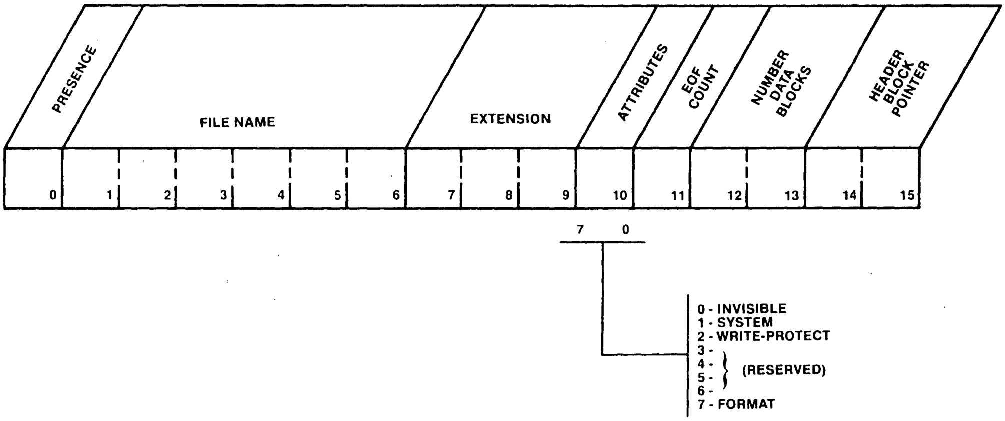
The Intel file system (described in detail in the RMX-80 Users Guide) uses the second disk file allocation method (previously discussed). In order to lower the system overhead involved in finding free data blocks, the Intel file system incorporates a free space management data structure known as a bit map. Each disk sector is represented by a single bit in the bit map. If a bit in the bit map is set to 1, the corresponding disk sector has been allocated. A zero in the bit map indicates that the corresponding sector is free. With this technique, the process of allocating or freeing a sector is accomplished by simply altering the bit map.

File names consist of a basic file name (up to six characters) and a file extension (up to three characters). The basic file name and the file extension are separated by a period (.). Examples of valid file names are: DRIV72.OBJ, XX.TMP, and FILE.CS. In addition, four file attributes are supported (see Figure 3 for attribute definitions).

The bit map and the file directory are placed on prespecified disk tracks (reserved for system use) beginning at track zero.

Disk File System Functions

Table 2 illustrates the typical functions implemented by a disk file system. As an example, the disk directory function (DIR) lists disk file information on the console display terminal. Figure 3 details the contents of a display entry in the Intel file system. The PL/M procedure outlined in Figure 4 illustrates a disk directory algorithm that displays the file name, the file attributes, and the file size (in blocks) for each file in the directory.



207885-5

Directory Entry

Presence is a flag that can contain one of three values:

000H— The file associated with this entry is present on the disk.

07FH— No file is associated with this entry; the content of the rest of the entry is undefined. The first entry with its flag set to 07FH marks the current logical end of the directory and directory searches stop at this entry.

0FFH— The file named in this entry once existed on the disk but is currently deleted. The next file added to the directory will be placed in the first entry marked 0FFH. This flag cannot, therefore, be used to (reliably) find a file that has been deleted. A value of 0FFH should be thought of as simply marking an open directory entry.

File Name is a string of up to 6 non-blank ASCII characters specifying the name of the file associated with the directory entry. If the file name is shorter than six characters, the remaining bytes contain binary zeros. For example, the name ALPHA would be stored as: 414C50484100H.

Extension is a string of up to 3 non-blank ASCII characters that specifies an extension to the file name. Extensions often identify the type of data in the file such as OBJ (object module), or PLM (PL/M source module). As with the file name, unused positions in the extension field are filled with binary zeros.

Attributes are bits that identify certain characteristics of the file. A 1 bit indicates that the file has the attribute, while a 0 bit means that the file does not have the attribute. The bit positions and their corresponding attributes are listed below (bit 0 is the low-order or rightmost bit, bit 7 is the leftmost bit):

0: **Invisible.** Files with this attribute are not listed by the ISIS-II DIR command unless the I switch is used. All system files are invisible.

1: **System.** Files with this attribute are copied to the disk in drive 1 when the S switch is specified with the ISIS-II FORMAT command.

2: **Write-Protect.** Files with this attribute cannot be opened for output or update, nor can they be deleted or renamed.

3–6: These positions are reserved for future use.

7: **Format.** Files with this attribute are treated as though they are write-protected. In addition, these files are created on a new diskette when the ISIS-II FORMAT command is issued. The system files all have the FORMAT attribute and it should not be given to any other files.

Figure 3. Intel Directory Entry Format

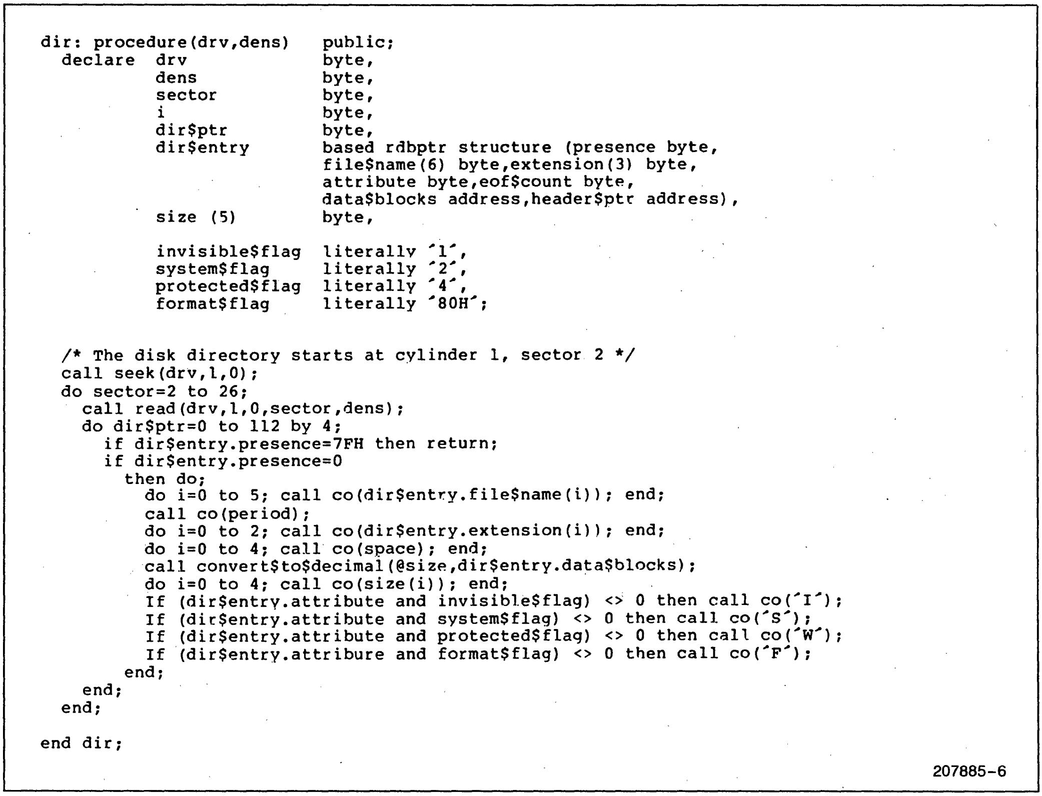
EOF Count contains the number of the last byte in the last data block of the file. If the value of this field is 080H, for example, the last byte in the file is byte number 128 in the last data block (the last block is full).

Number of Data Blocks is an address variable that indicates the number of data blocks currently used by the file. ISIS-II and the RMX/80 Disk File system both maintain a counter called LENGTH that is the current number of bytes in the file. This is calculated as:

(Number of Data Blocks − 1) × 128 + EOF Count.

Header Block Pointer is the address of the file's header block. The high byte of the field is the sector number and the low byte is the track number. The system "finds" a disk file by searching the directory for the name and then using the header block pointer to seek to the beginning of the file.

Figure 3. Intel Directory Entry Format (Continued)



```
dir: procedure(drv,dens)   public;
  declare  drv             byte,
           dens            byte,
           sector          byte,
           i               byte,
           dir$ptr         byte,
           dir$entry       based rdbptr structure (presence byte,
                           file$name(6) byte,extension(3) byte,
                           attribute byte,eof$count byte,
                           data$blocks address,header$ptr address),
           size (5)        byte,

           invisible$flag  literallv '1',
           system$flag     literally '2',
           protected$flag  literally '4',
           format$flag     literally '80H';

  /* The disk directory starts at cylinder 1, sector 2 */
  call seek(drv,1,0);
  do sector=2 to 26;
    call read(drv,1,0,sector,dens);
    do dir$ptr=0 to 112 by 4;
      if dir$entry.presence=7FH then return;
      if dir$entry.presence=0
        then do;
          do i=0 to 5; call co(dir$entry.file$name(i)); end;
          call co(period);
          do i=0 to 2; call co(dir$entry.extension(i)); end;
          do i=0 to 4; call co(space); end;
          call convert$to$decimal(@size,dir$entry.data$blocks);
          do i=0 to 4; call co(size(i)); end;
          If (dir$entry.attribute and invisible$flag) <> 0 then call co('I');
          If (dir$entry.attribute and system$flag) <> 0 then call co('S');
          If (dir$entry.attribute and protected$flag) <> 0 then call co('W');
          If (dir$entry.attribure and format$flag) <> 0 then call co('F');
        end;
    end;
  end;

end dir;
```

207885-6

Figure 4. Sample PL/M Directory Procedure

7.0 KEY 8272 SOFTWARE INTERFACING CONSIDERATIONS

This section contains a quick review of **Key** 8272 Software design features and issues. (Most items have been mentioned in other sections of this application note.) Before designing 8272 software drivers, it is advisable that the information in this section be thoroughly understood.

1. Non-DMA Transfers

In systems that operate without a DMA controller (in the polled or interrupt driven mode), the system software is responsible for counting data transfers to/from the 8272 and generating a TC signal to the FDC when the transfer is complete.

2. Processor Command/Result Phase Interface

In the command phase, the driver software must write the exact number of parameters in the exact order shown in Table 5. During the result phase, the driver must read the complete result status. For example, the Format Track command requires six command bytes and presents seven result bytes. The 8272 will not accept a new command until all result bytes are read. Note that the number of command and result bytes varies from command-to-command. **Command and result phases cannot be shortened.**

During both the command and result phases, the Main Status Register must be read by the driver before each byte of information is read from, or written to, the FDC Data Register. Before each command byte is written, DIO (bit 6) must be low (indicating a data transfer from the processor) and RQM (bit 7) must be high (indicating that the FDC is ready for data). During the result phase, DIO must be high (indicating a data transfer to the processor).

NOTE:
After the 8272 receives a command byte, the RQM flag may remain set for approximately 16 microseconds (with an 8 MHz clock). The driver should not attempt to read the Main Status Register before this time interval has elapsed; otherwise, the driver may erroneously assume that the FDC is ready to accept the next byte.

3. Sector Sizes

The 8272 does not support 128 byte sectors in the MFM (double-density) mode.

4. Drive Status Changes

The 8272 constantly polls all drives for changes in the drive ready status. This polling begins immediately following RESET. An interrupt is generated every time the FDC senses a change in the drive ready status. After reset, the FDC assumes that all drives are "not ready". If a drive is ready immediately after reset, the 8272 generates a drive status change interrupt.

5. Seek Commands

The 8272 FDC does not perform implied seeks. Before issuing a data read or write command, the read/write head must be positioned over the correct cylinder by means of an explicit seek command. If the head is not positioned correctly, a cylinder address error is generated.

6. Interrupt Processing

When the processor receives an interrupt from the FDC, the FDC may be reporting one of two distinct events:

a) The beginning of the result phase of a previously requested read, write, or scan command.

b) An asynchronous event such as a seek/recalibrate completion, an attention, an abnormal command termination, or an invalid command.

These two cases are distinguished by the FDC BUSY flag (bit 4) in the Main Status Register. If the FDC BUSY flag is high, the interrupt is of type (a). If the FDC BUSY flag is low, the interrupt was caused by an asynchronous event (b).

A single interrupt from the FDC may signal more than one of the above events. After receiving an interrupt, the processor must continue to issue Sense Interrupt Status commands (and service the resulting conditions) until an invalid command code is received. In this manner, all "hidden" interrupts are ferreted out and serviced.

7. Skip Flag (SK)

The skip flag is used during the execution of Read Data, Read Deleted Data, Read Track, and various Scan commands. This flag permits the FDC to skip unwanted sectors on a disk track.

When performing a Read Data, Read Track, or Scan command, a high SK flag indicates that the FDC is to skip over (not transfer) any sector containing a deleted data address mark. A low SK flag indicates that the FDC is to terminate the command (after reading all the data in the sector) when a deleted data address mark is encountered.

When performing a Read Deleted Data command, a high SK flag indicates that sectors containing normal data address marks are to be skipped. Note that this is just the opposite situation from that described in the last paragraph. When a data address mark is encountered during a Read Deleted Data command (and the SK flag is low), the FDC terminates the command after reading all the data in the sector.

8. Bad Track Maintenance

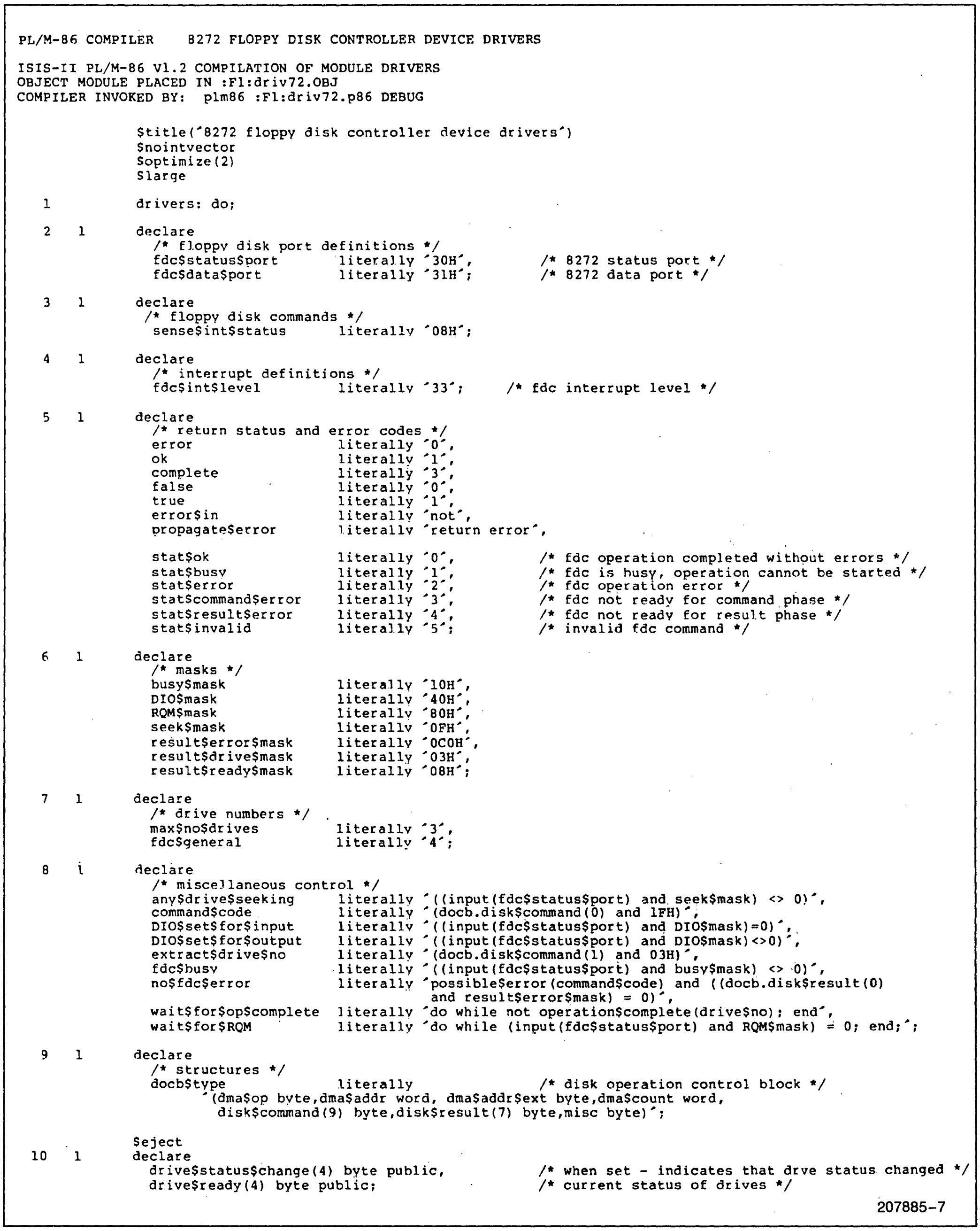
The 8272 does not internally maintain bad track information. The maintenance of this information must be performed by system software. As an example of typical bad track operation, assume that a media test determines that track 31 and track 66 of a given floppy disk are bad. When the disk is formatted for use, the system software formats physical track 0 as logical cylinder 0 (C = 0 in the command phase parameters), physical track 1 as logical track 1 (C = 1), and so on, until physical track 30 is formatted as logical cylinder 30 (C = 30). Physical track 31 is bad and should be formatted as logical cylinder FF (indicating a bad track). Next, physical track 32 is formatted as logical cylinder 31, and so on, until physical track 65 is formatted as logical cylinder 64. Next, bad physical track 66 is formatted as logical cylinder FF (another bad track marker), and physical track 67 is formatted as logical cylinder 65. This formatting continues until the last physical track (77) is formatted as logical cylinder 75. Normally, after this formatting is complete, the bad track information is stored in a prespecified area on the floppy disk (typically in a sector on track 0) so that the system will be able to recreate the bad track information when the disk is removed from the drive and reinserted at some later time.

To illustrate how the system software performs a transfer operation on a disk with bad tracks, assume that the disk drive head is positioned at track 0 and the disk described above is loaded into the drive. If a command to read track 36 is issued by an application program, the system software translates this read command into a **seek** to physical track 37 (since there is one bad track between 0 and 36, namely 31) followed by a **read** of logical cylinder 36. Thus, the cylinder parameter C is set to 37 for the Seek command and 36 for the Read Sector command.

REFERENCES

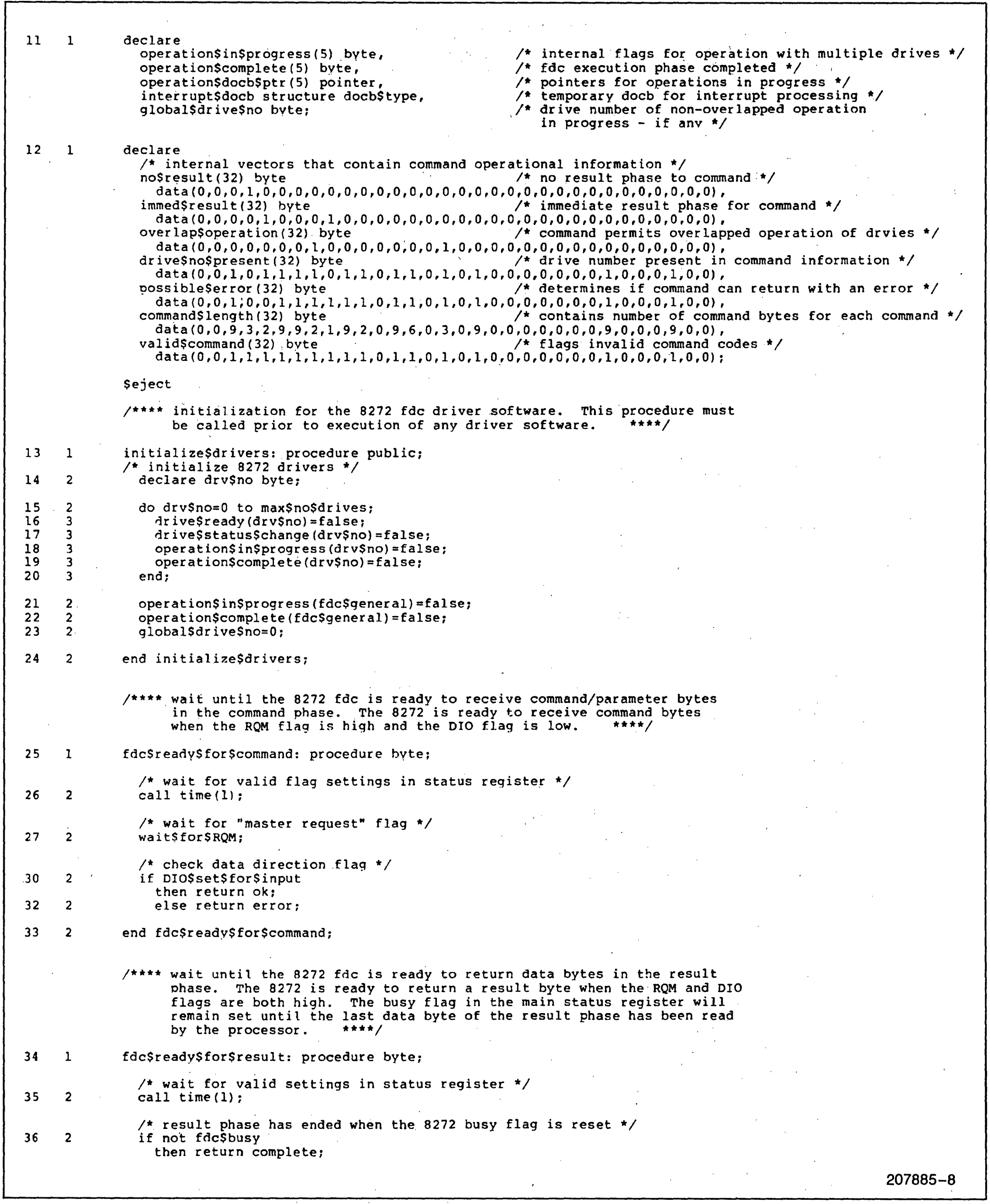
1. Intel, "8272 Single/Double Density Floppy Disk Controller Data Sheet," Intel Corporation, 1980.
2. Intel, "An Intelligent Data Base System Using the 8272," Intel Application Note," AP-116, 1981.
3. Intel, *iSBC 208 Hardware Reference Manual,* Manual Order No. 143078, Intel Corporation, 1980.
4. Intel, *RMX/80 User's Guide,* Manual Order No. 9800522, Intel Corporation, 1978.
5. Brinch Hansen, P., *Operating System Principles,* Prentice-Hall, Inc., New Jersey, 1973.
6. Flores, I., *Computer Software: Programming Systems for Digital Computers,* Prentice-Hall, Inc., New Jersey, 1965.
7. Knuth, D. E., *Fundamental Algorithms,* Addison-Wesley Publishing Company, Massachusetts, 1975.
8. Shaw, A. C., *The Logical Design of Operating Systems,* Prentice-Hall, Inc., New Jersey, 1974.
9. Watson, R. W., *Time Sharing System Design Concepts,* McGraw-Hill, Inc., New York, 1970.
10. Zarrella, J., *Operating Systems: Concepts and Principles,* Microcomputer Applications, California, 1979.

APPENDIX A
8272 FDC DEVICE DRIVER SOFTWARE



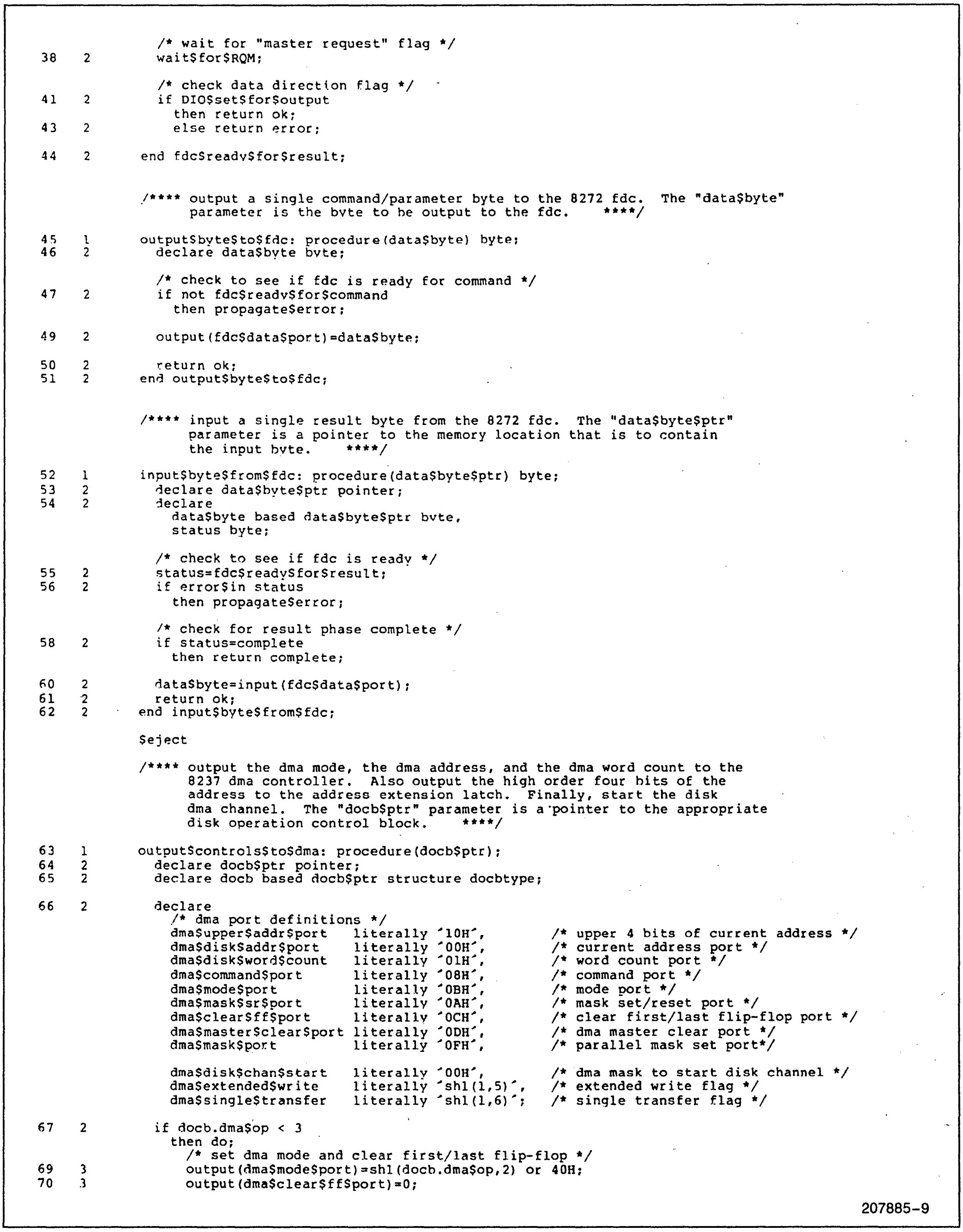
```
PL/M-86 COMPILER    8272 FLOPPY DISK CONTROLLER DEVICE DRIVERS

ISIS-II PL/M-86 V1.2 COMPILATION OF MODULE DRIVERS
OBJECT MODULE PLACED IN :F1:driv72.OBJ
COMPILER INVOKED BY:  plm86 :F1:driv72.p86 DEBUG

              $title('8272 floppy disk controller device drivers')
              $nointvector
              $optimize(2)
              $large

  1           drivers: do;

  2   1       declare
                /* floppy disk port definitions */
                fdc$status$port        literally '30H',        /* 8272 status port */
                fdc$data$port          literally '31H';        /* 8272 data port */

  3   1       declare
               /* floppy disk commands */
                sense$int$status       literally '08H';

  4   1       declare
                /* interrupt definitions */
                fdc$int$level          literally '33';     /* fdc interrupt level */

  5   1       declare
                /* return status and error codes */
                error                  literally '0',
                ok                     literally '1',
                complete               literally '3',
                false                  literally '0',
                true                   literally '1',
                error$in               literally 'not',
                propagate$error        literally 'return error',

                stat$ok                literally '0',          /* fdc operation completed without errors */
                stat$busy              literally '1',          /* fdc is busy, operation cannot be started */
                stat$error             literally '2',          /* fdc operation error */
                stat$command$error     literally '3',          /* fdc not ready for command phase */
                stat$result$error      literally '4',          /* fdc not ready for result phase */
                stat$invalid           literally '5';          /* invalid fdc command */

  6   1       declare
                /* masks */
                busy$mask              literally '10H',
                DIO$mask               literally '40H',
                RQM$mask               literally '80H',
                seek$mask              literally '0FH',
                result$error$mask      literally '0C0H',
                result$drive$mask      literally '03H',
                result$ready$mask      literally '08H';

  7   1       declare
                /* drive numbers */
                max$no$drives          literally '3',
                fdc$general            literally '4';

  8   1       declare
                /* miscellaneous control */
                any$drive$seeking      literally '((input(fdc$status$port) and seek$mask) <> 0)',
                command$code           literally '(docb.disk$command(0) and 1FH)',
                DIO$set$for$input      literally '((input(fdc$status$port) and DIO$mask)=0)',
                DIO$set$for$output     literally '((input(fdc$status$port) and DIO$mask)<>0)',
                extract$drive$no       literally '(docb.disk$command(1) and 03H)',
                fdc$busy               literally '((input(fdc$status$port) and busy$mask) <> 0)',
                no$fdc$error           literally 'possible$error(command$code) and ((docb.disk$result(0)
                                                  and result$error$mask) = 0)',
                wait$for$op$complete   literally 'do while not operation$complete(drive$no); end',
                wait$for$RQM           literally 'do while (input(fdc$status$port) and RQM$mask) = 0; end;';

  9   1       declare
                /* structures */
                docb$type              literally               /* disk operation control block */
                     '(dma$op byte,dma$addr word, dma$addr$ext byte,dma$count word,
                      disk$command(9) byte,disk$result(7) byte,misc byte)';

              $eject
 10   1       declare
                drive$status$change(4) byte public,            /* when set - indicates that drve status changed */
                drive$ready(4) byte public;                    /* current status of drives */
```

207885-7



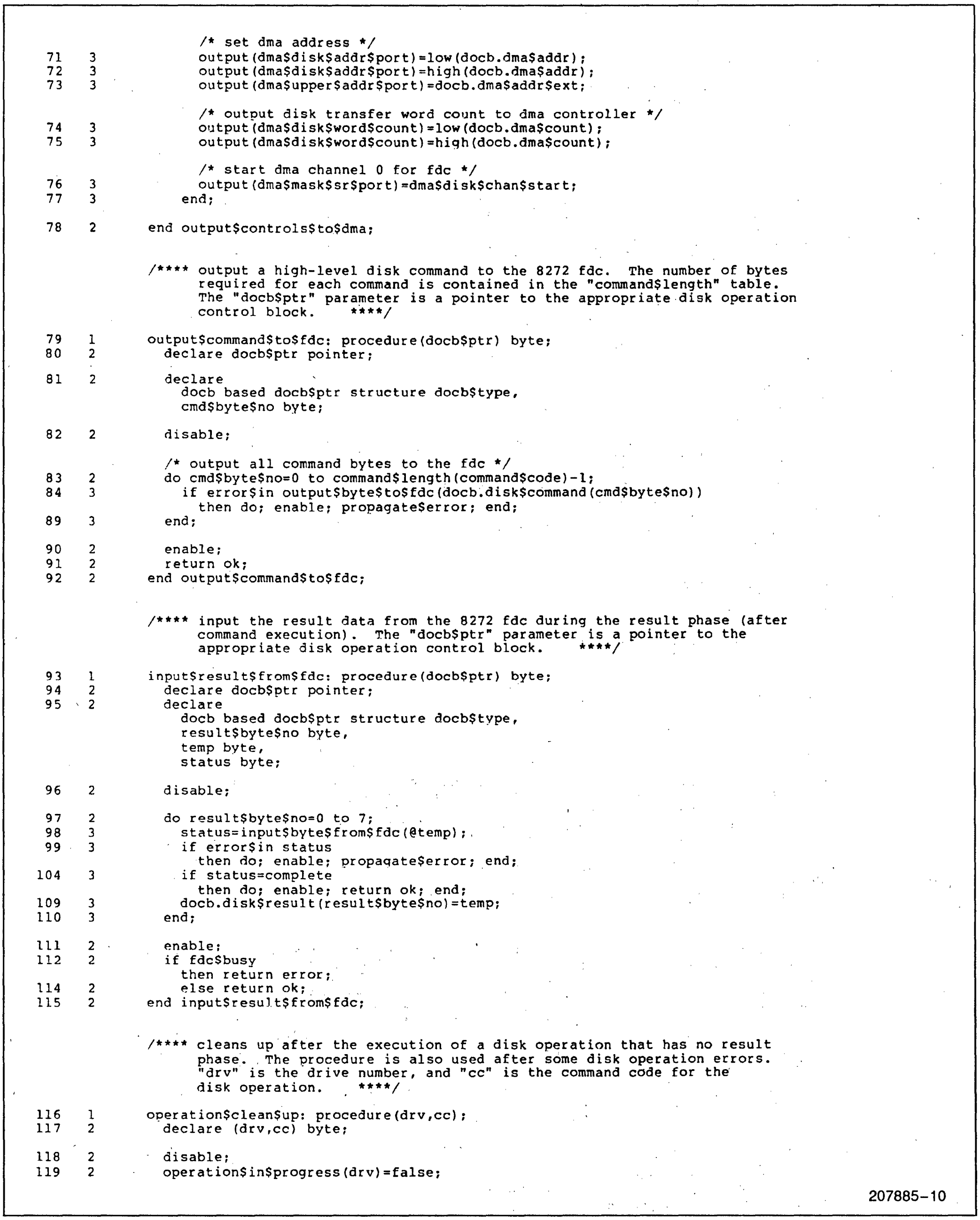
```
11   1      declare
              operation$in$progress(5) byte,             /* internal flags for operation with multiple drives */
              operation$complete(5) byte,                /* fdc execution phase completed */
              operation$docb$ptr(5) pointer,             /* pointers for operations in progress */
              interrupt$docb structure docb$type,        /* temporary docb for interrupt processing */
              global$drive$no byte;                      /* drive number of non-overlapped operation
                                                            in progress - if any */

12   1      declare
              /* internal vectors that contain command operational information */
              no$result(32) byte                         /* no result phase to command */
                data(0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
              immed$result(32) byte                      /* immediate result phase for command */
                data(0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
              overlap$operation(32) byte                 /* command permits overlapped operation of drvies */
                data(0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
              drive$no$present(32) byte                  /* drive number present in command information */
                data(0,0,1,0,1,1,1,1,0,1,1,0,1,1,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,1,0,0),
              possible$error(32) byte                    /* determines if command can return with an error */
                data(0,0,1,0,0,1,1,1,1,1,1,0,1,1,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,1,0,0),
              command$length(32) byte                    /* contains number of command bytes for each command */
                data(0,0,9,3,2,9,9,2,1,9,2,0,9,6,0,3,0,9,0,0,0,0,0,0,0,9,0,0,0,9,0,0),
              valid$command(32) byte                     /* flags invalid command codes */
                data(0,0,1,1,1,1,1,1,1,1,1,0,1,1,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,1,0,0);

            $eject

            /**** initialization for the 8272 fdc driver software.  This procedure must
                  be called prior to execution of any driver software.    ****/

13   1      initialize$drivers: procedure public;
            /* initialize 8272 drivers */
14   2        declare drv$no byte;

15   2        do drv$no=0 to max$no$drives;
16   3          drive$ready(drv$no)=false;
17   3          drive$status$change(drv$no)=false;
18   3          operation$in$progress(drv$no)=false;
19   3          operation$complete(drv$no)=false;
20   3        end;

21   2        operation$in$progress(fdc$general)=false;
22   2        operation$complete(fdc$general)=false;
23   2        global$drive$no=0;

24   2      end initialize$drivers;


            /**** wait until the 8272 fdc is ready to receive command/parameter bytes
                  in the command phase.  The 8272 is ready to receive command bytes
                  when the RQM flag is high and the DIO flag is low.    ****/

25   1      fdc$ready$for$command: procedure byte;

              /* wait for valid flag settings in status register */
26   2        call time(1);

              /* wait for "master request" flag */
27   2        wait$for$RQM;

              /* check data direction flag */
30   2        if DIO$set$for$input
                then return ok;
32   2          else return error;

33   2      end fdc$ready$for$command;


            /**** wait until the 8272 fdc is ready to return data bytes in the result
                  phase.  The 8272 is ready to return a result byte when the RQM and DIO
                  flags are both high.  The busy flag in the main status register will
                  remain set until the last data byte of the result phase has been read
                  by the processor.    ****/

34   1      fdc$ready$for$result: procedure byte;

              /* wait for valid settings in status register */
35   2        call time(1);

              /* result phase has ended when the 8272 busy flag is reset */
36   2        if not fdc$busy
                then return complete;
```

207885-8



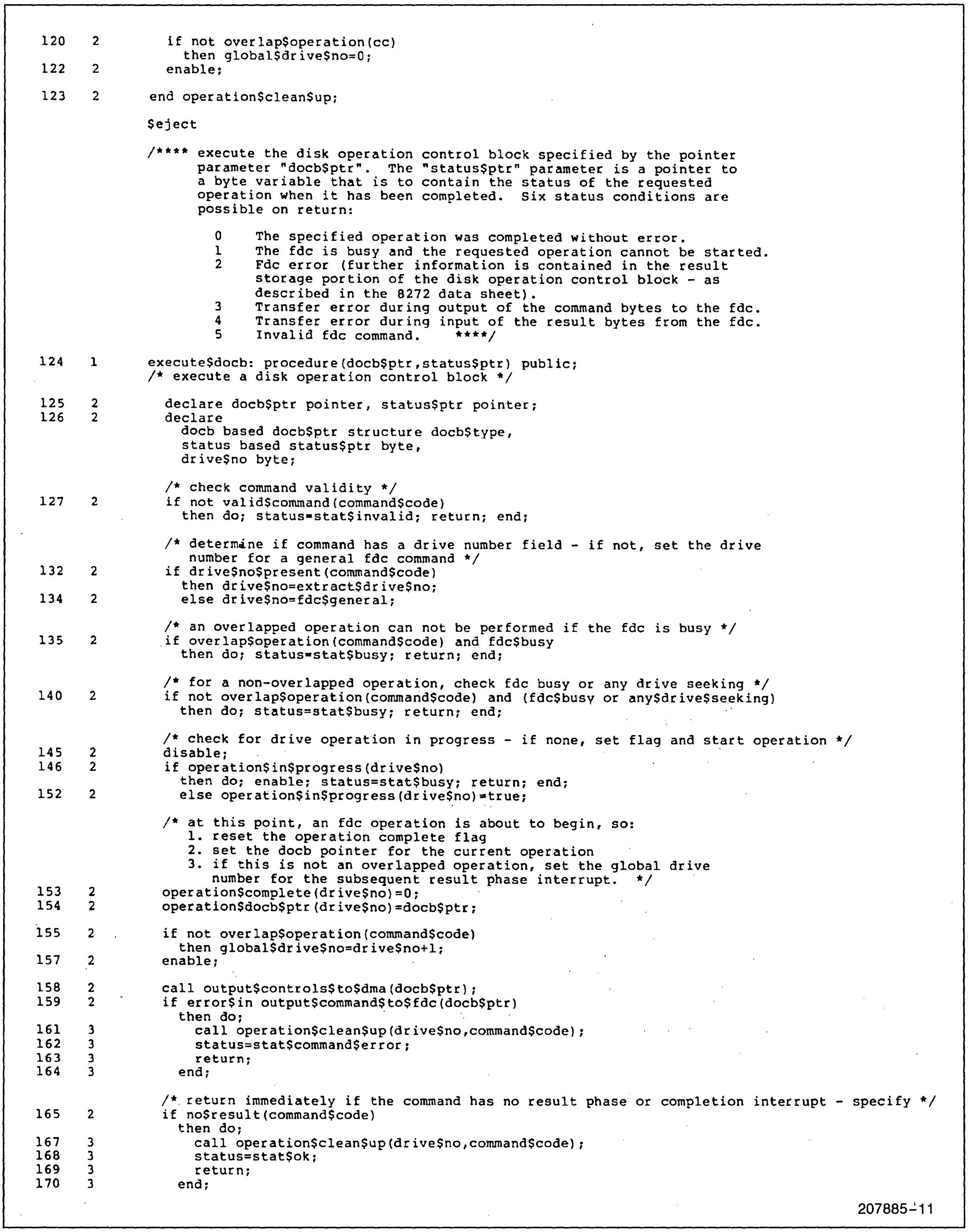
```
                  /* wait for "master request" flag */
 38    2          wait$for$RQM;

                  /* check data direction flag */
 41    2          if DIO$set$for$output
                    then return ok;
 43    2            else return error;

 44    2        end fdc$readv$for$result;


                /**** output a single command/parameter byte to the 8272 fdc.  The "data$byte"
                      parameter is the bvte to he output to the fdc.     ****/

 45    1        output$byte$to$fdc: procedure(data$byte) byte;
 46    2          declare data$byte bvte;

                  /* check to see if fdc is ready for command */
 47    2          if not fdc$readv$for$command
                    then propagate$error;

 49    2          output(fdc$data$port)=data$byte;

 50    2          return ok;
 51    2        end output$byte$to$fdc;


                /**** input a single result byte from the 8272 fdc.  The "data$byte$ptr"
                      parameter is a pointer to the memory location that is to contain
                      the input bvte.     ****/

 52    1        input$byte$from$fdc: procedure(data$byte$ptr) byte;
 53    2          declare data$bvte$ptr pointer;
 54    2          declare
                    data$byte based data$byte$ptr bvte,
                    status byte;

                  /* check to see if fdc is readv */
 55    2          status=fdc$readv$for$result;
 56    2          if error$in status
                    then propagate$error;

                  /* check for result phase complete */
 58    2          if status=complete
                    then return complete;

 60    2          data$byte=input(fdc$data$port);
 61    2          return ok;
 62    2        end input$byte$from$fdc;

                $eject

                /**** output the dma mode, the dma address, and the dma word count to the
                      8237 dma controller.  Also output the high order four bits of the
                      address to the address extension latch.  Finally, start the disk
                      dma channel.  The "docb$ptr" parameter is a pointer to the appropriate
                      disk operation control block.     ****/

 63    1        output$controls$to$dma: procedure(docb$ptr);
 64    2          declare docb$ptr pointer;
 65    2          declare docb based docb$ptr structure docbtype;

 66    2          declare
                    /* dma port definitions */
                    dma$upper$addr$port   literally '10H',        /* upper 4 bits of current address */
                    dma$disk$addr$port    literally '00H',        /* current address port */
                    dma$disk$word$count   literally '01H',        /* word count port */
                    dma$command$port      literally '08H',        /* command port */
                    dma$mode$port         literally '0BH',        /* mode port */
                    dma$mask$sr$port      literally '0AH',        /* mask set/reset port */
                    dma$clear$ff$port     literally '0CH',        /* clear first/last flip-flop port */
                    dma$master$clear$port literally '0DH',        /* dma master clear port */
                    dma$mask$port         literally '0FH',        /* parallel mask set port*/

                    dma$disk$chan$start   literally '00H',        /* dma mask to start disk channel */
                    dma$extended$write    literally 'shl(1,5)',   /* extended write flag */
                    dma$single$transfer   literally 'shl(1,6)';   /* single transfer flag */

 67    2          if docb.dma$op < 3
                    then do;
                      /* set dma mode and clear first/last flip-flop */
 69    3              output(dma$mode$port)=shl(docb.dma$op,2) or 40H;
 70    3              output(dma$clear$ff$port)=0;
```

207885-9



```
                    /* set dma address */
 71   3             output(dma$disk$addr$port)=low(docb.dma$addr);
 72   3             output(dma$disk$addr$port)=high(docb.dma$addr);
 73   3             output(dma$upper$addr$port)=docb.dma$addr$ext;

                    /* output disk transfer word count to dma controller */
 74   3             output(dma$disk$word$count)=low(docb.dma$count);
 75   3             output(dma$disk$word$count)=high(docb.dma$count);

                    /* start dma channel 0 for fdc */
 76   3             output(dma$mask$sr$port)=dma$disk$chan$start;
 77   3           end;

 78   2       end output$controls$to$dma;


              /**** output a high-level disk command to the 8272 fdc.  The number of bytes
                    required for each command is contained in the "command$length" table.
                    The "docb$ptr" parameter is a pointer to the appropriate disk operation
                    control block.    ****/

 79   1       output$command$to$fdc: procedure(docb$ptr) byte;
 80   2         declare docb$ptr pointer;

 81   2         declare
                  docb based docb$ptr structure docb$type,
                  cmd$byte$no byte;

 82   2         disable;

                /* output all command bytes to the fdc */
 83   2         do cmd$byte$no=0 to command$length(command$code)-1;
 84   3           if error$in output$byte$to$fdc(docb.disk$command(cmd$byte$no))
                      then do; enable; propagate$error; end;
 89   3         end;

 90   2         enable;
 91   2         return ok;
 92   2       end output$command$to$fdc;


              /**** input the result data from the 8272 fdc during the result phase (after
                    command execution). The "docb$ptr" parameter is a pointer to the
                    appropriate disk operation control block.    ****/

 93   1       input$result$from$fdc: procedure(docb$ptr) byte;
 94   2         declare docb$ptr pointer;
 95   2         declare
                  docb based docb$ptr structure docb$type,
                  result$byte$no byte,
                  temp byte,
                  status byte;

 96   2         disable;

 97   2         do result$byte$no=0 to 7;
 98   3           status=input$byte$from$fdc(@temp);
 99   3           if error$in status
                      then do; enable; propagate$error; end;
104   3           if status=complete
                      then do; enable; return ok; end;
109   3           docb.disk$result(result$byte$no)=temp;
110   3         end;

111   2         enable;
112   2         if fdc$busy
                   then return error;
114   2           else return ok;
115   2       end input$result$from$fdc;


              /**** cleans up after the execution of a disk operation that has no result
                    phase.  The procedure is also used after some disk operation errors.
                    "drv" is the drive number, and "cc" is the command code for the
                    disk operation.    ****/

116   1       operation$clean$up: procedure(drv,cc);
117   2         declare (drv,cc) byte;

118   2         disable;
119   2         operation$in$progress(drv)=false;
```

207885-10



```
 120   2         if not overlap$operation(cc)
                   then global$drive$no=0;
 122   2         enable;

 123   2       end operation$clean$up;

               $eject

               /**** execute the disk operation control block specified by the pointer
                     parameter "docb$ptr".  The "status$ptr" parameter is a pointer to
                     a byte variable that is to contain the status of the requested
                     operation when it has been completed.  Six status conditions are
                     possible on return:

                       0    The specified operation was completed without error.
                       1    The fdc is busy and the requested operation cannot be started.
                       2    Fdc error (further information is contained in the result
                            storage portion of the disk operation control block - as
                            described in the 8272 data sheet).
                       3    Transfer error during output of the command bytes to the fdc.
                       4    Transfer error during input of the result bytes from the fdc.
                       5    Invalid fdc command.    ****/

 124   1       execute$docb: procedure(docb$ptr,status$ptr) public;
               /* execute a disk operation control block */

 125   2         declare docb$ptr pointer, status$ptr pointer;
 126   2         declare
                   docb based docb$ptr structure docb$type,
                   status based status$ptr byte,
                   drive$no byte;

                 /* check command validity */
 127   2         if not valid$command(command$code)
                   then do; status=stat$invalid; return; end;

                 /* determine if command has a drive number field - if not, set the drive
                    number for a general fdc command */
 132   2         if drive$no$present(command$code)
                   then drive$no=extract$drive$no;
 134   2           else drive$no=fdc$general;

                 /* an overlapped operation can not be performed if the fdc is busy */
 135   2         if overlap$operation(command$code) and fdc$busy
                   then do; status=stat$busy; return; end;

                 /* for a non-overlapped operation, check fdc busy or any drive seeking */
 140   2         if not overlap$operation(command$code) and (fdc$busy or any$drive$seeking)
                   then do; status=stat$busy; return; end;

                 /* check for drive operation in progress - if none, set flag and start operation */
 145   2         disable;
 146   2         if operation$in$progress(drive$no)
                   then do; enable; status=stat$busy; return; end;
 152   2           else operation$in$progress(drive$no)=true;

                 /* at this point, an fdc operation is about to begin, so:
                    1. reset the operation complete flag
                    2. set the docb pointer for the current operation
                    3. if this is not an overlapped operation, set the global drive
                       number for the subsequent result phase interrupt.  */
 153   2         operation$complete(drive$no)=0;
 154   2         operation$docb$ptr(drive$no)=docb$ptr;

 155   2         if not overlap$operation(command$code)
                   then global$drive$no=drive$no+1;
 157   2         enable;

 158   2         call output$controls$to$dma(docb$ptr);
 159   2         if error$in output$command$to$fdc(docb$ptr)
                   then do;
 161   3             call operation$clean$up(drive$no,command$code);
 162   3             status=stat$command$error;
 163   3             return;
 164   3           end;

                 /* return immediately if the command has no result phase or completion interrupt - specify */
 165   2         if no$result(command$code)
                   then do;
 167   3             call operation$clean$up(drive$no,command$code);
 168   3             status=stat$ok;
 169   3             return;
 170   3           end;
```

207885-11

```
 171    2          if immed$result(command$code)
                     then do;
 173    3              if error$in input$result$from$fdc(docb$ptr)
                         then do;
 175    4                  call operation$clean$up(drive$no,command$code);
 176    4                  status=stat$result$error;
 177    4                  return;
 178    4                end;
 179    3          end;
 180    2          else do;
 181    3            wait$for$op$complete;
 183    3            if docb.misc = error
                       then do; status=stat$result$error; return; end;
 188    3          end;

 189    2        if no$fdc$error
                   then status=stat$ok;
 191    2          else status=stat$error;

 192    2      end execute$docb;

               $eject

               /**** copy disk command results from the interrupt control block to the
                     currently active disk operation control block if a disk operation is
                     in progress.    ****/

 193    1      copy$int$result: procedure(drv);
 194    2        declare drv byte;
 195    2        declare
                   i byte,
                   docb$ptr pointer,
                   docb based docb$ptr structure docb$type;

 196    2        if operation$in$progress(drv)
                   then do;
 198    3            docb$ptr=operation$docb$ptr(drv);
 199    3            do i=1 to 6; docb.disk$result(i)=interrupt$docb.disk$result(i); end;
 202    3            docb.misc=ok;
 203    3            operation$in$progress(drv)=false;
 204    3            operation$complete(drv)=true;
 205    3          end;

 206    2      end copy$int$result;

               /**** interrupt processing for 8272 fdc drivers.  Basically, two types of
                     interrupts are generated by the 8272: (a)when the execution phase of
                     an operation has been completed, an interrupt is generated to signal
                     the beginning of the result phase (the fdc busy flag is set
                     when this interrupt is received), and (b) when an overlapped operation
                     is completed or an unexpected interrupt is received (the fdc busy flag
                     is not set when this interrupt is received).

                     When interrupt type (a) is received, the result bytes from the operation
                     are read from the 8272 and the operation complete flag is set.

                     When an interrupt of type (b) is received, the interrupt result code is
                     examined to determine which of the following four actions are indicated:

                       1. An overlapped option (recalibrate or seek) has been completed.  The
                          result data is read from the 8272 and placed in the currently active
                          disk operation control block.
                       2. An abnormal termination of an operation has occurred.  The result
                          data is read and placed in the currently active disk operation
                          control block.
                       3. The execution of an invalid command has been attempted.  This
                          signals the successful completion of all interrupt processing.
                       4. The ready status of a drive has changed.  The "drive$ready" and
                          "drive$ready$status" change tables are updated.  If an operation
                          is currently in progress on the affected drive, the result data
                          is placed in the currently active disk operation control block.

                     After an interrupt is processed, additional sense interrupt status commands
                     must be issued and processed until an invalid command result is returned
                     from the fdc.  This action guarantees that all "hidden" interrupts
                     are serviced.    ****/
```

207885-12

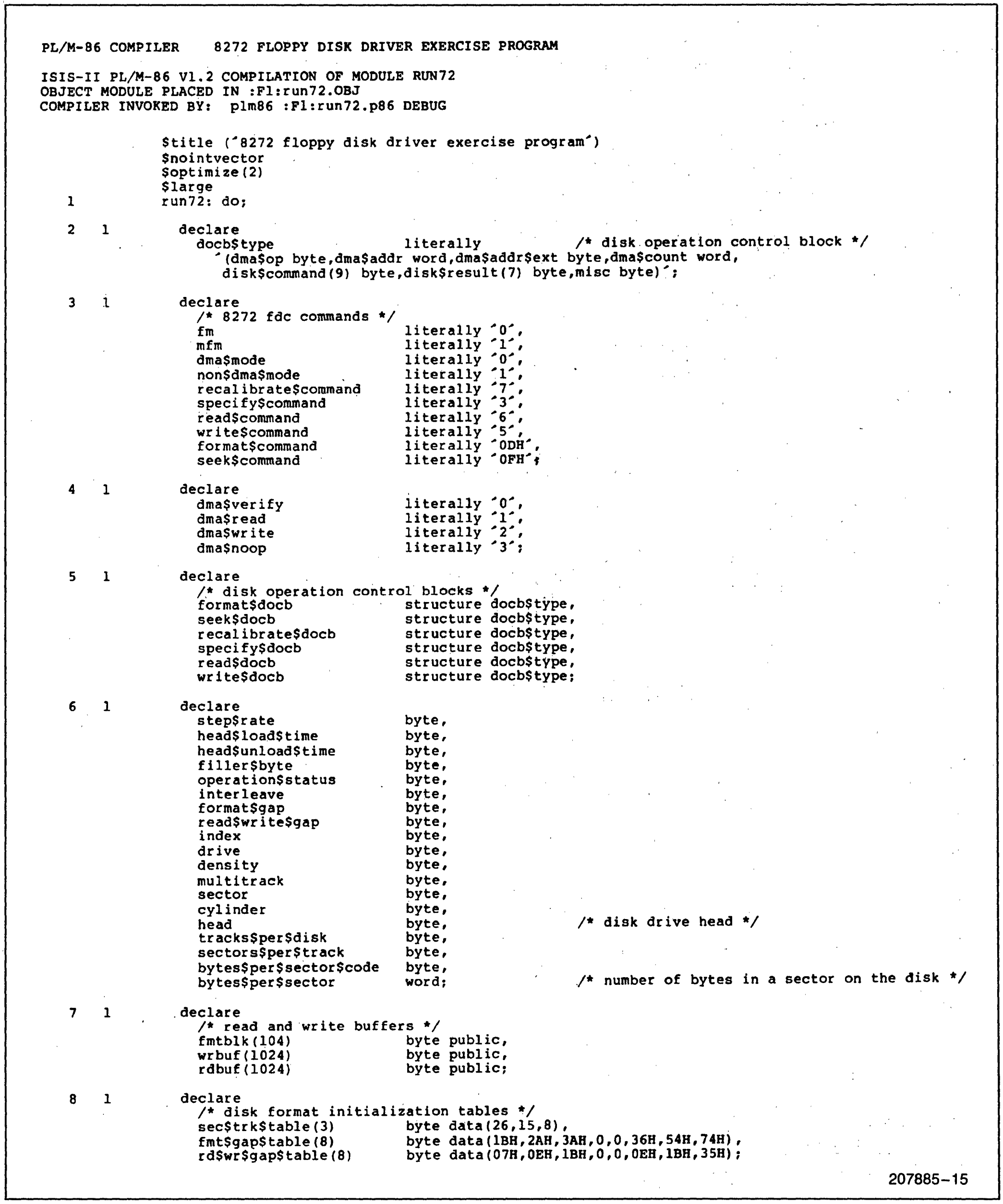
```
207   1      fdcint: procedure public interrupt fdc$int$level;
208   2        declare
                 invalid byte,
                 drive$no byte,
                 docb$ptr pointer,
                 docb based docb$ptr structure docb$type;

209   2        declare
                 /* interrupt port definitions */
                 ocw2                    literally '70H',
                 nseoi                   literally 'shl(1,5)';

210   2        declare
                 /* miscellaneous flags */
                 result$code             literally 'shr(interrupt$docb.disk$result(0) and result$error$mask,6)',
                 result$drive$ready      literally '((interrupt$docb.disk$result(0) and result$ready$mask) = 0)',
                 extract$result$drive$no literally '(interrupt$docb.disk$result(0) and result$drive$mask)',
                 end$of$interrupt        literally 'output(ocw2)=nseoi';

               /* if the fdc is busy when an interrupt is received, then the result
                  phase of the previous non-overlapped operation has begun */
211   2        if fdc$busy
                 then do;
                   /* process interrupt if operation in progress */
213   3            if global$drive$no <> 0
                     then do;
215   4                docb$ptr=operation$docb$ptr(global$drive$no-1);
216   4                if error$in input$result$from$fdc(docb$ptr)
                         then docb.misc=error;
218   4                  else docb.misc=ok;
219   4                operation$in$progress(global$drive$no-1)=false;
220   4                operation$complete(global$drive$no-1)=true;
221   4                global$drive$no=0;
222   4              end;
223   3          end;

               /* if the fdc is not busy, then either an overlapped operation has been
                  completed or an unexpected interrupt has occurred (e.g., drive status
                  change) */
224   2          else do;
225   3            invalid=false;
226   3            do while not invalid;

                     /* perform a sense interrupt status operation - if errors are detected,
                        in the actual fdc interface, interrupt processing is discontinued */
227   4              if error$in output$byte$to$fdc(sense$int$status) then go to ignore;
229   4              if error$in input$result$from$fdc(@interrupt$docb) then go to ignore;

231   4              do case result$code;

                       /* case 0 - operation complete */
232   5                do;
233   6                  drive$no=extract$result$drive$no;
234   6                  call copy$int$result(drive$no);
235   6                end;

                       /* case 1 - abnormal termination */

236   5                do;
237   6                  drive$no=extract$result$drive$no;
238   6                  call copy$int$result(drive$no);
239   6                end;

                       /* case 2 - invalid command */
240   5                invalid=true;

                       /* case 3 - drive ready change */
241   5                do;
242   6                  drive$no=extract$result$drive$no;
243   6                  call copy$int$result(drive$no);
244   6                  drive$status$change(drive$no)=true;
245   6                  if result$drive$ready
                           then drive$ready(drive$no)=true;
247   6                    else drive$ready(drive$no)=false;
248   6                end;
249   5              end;
250   4            end;
251   3          end;

252   2        ignore: end$of$interrupt;
253   2      end fdcint;

254   1      end drivers;
```

207885-13

```
MODULE INFORMATION:

     CODE AREA SIZE     = 0615H   1557D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0050H     80D
     MAXIMUM STACK SIZE = 0032H     50D
     564 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

207885-14

APPENDIX B
8272 FDC EXERCISER PROGRAM



```
PL/M-86 COMPILER    8272 FLOPPY DISK DRIVER EXERCISE PROGRAM

ISIS-II PL/M-86 V1.2 COMPILATION OF MODULE RUN72
OBJECT MODULE PLACED IN :F1:run72.OBJ
COMPILER INVOKED BY:  plm86 :F1:run72.p86 DEBUG

                $title ('8272 floppy disk driver exercise program')
                $nointvector
                $optimize(2)
                $large
   1            run72: do;

   2   1          declare
                    docb$type                 literally          /* disk operation control block */
                      '(dma$op byte,dma$addr word,dma$addr$ext byte,dma$count word,
                      disk$command(9) byte,disk$result(7) byte,misc byte)';

   3   1          declare
                    /* 8272 fdc commands */
                    fm                        literally '0',
                    mfm                       literally '1',
                    dma$mode                  literally '0',
                    non$dma$mode              literally '1',
                    recalibrate$command       literally '7',
                    specify$command           literally '3',
                    read$command              literally '6',
                    write$command             literally '5',
                    format$command            literally '0DH',
                    seek$command              literally '0FH';

   4   1          declare
                    dma$verify                literally '0',
                    dma$read                  literally '1',
                    dma$write                 literally '2',
                    dma$noop                  literally '3';

   5   1          declare
                    /* disk operation control blocks */
                    format$docb               structure docb$type,
                    seek$docb                 structure docb$type,
                    recalibrate$docb          structure docb$type,
                    specify$docb              structure docb$type,
                    read$docb                 structure docb$type,
                    write$docb                structure docb$type;

   6   1          declare
                    step$rate                 byte,
                    head$load$time            byte,
                    head$unload$time          byte,
                    filler$byte               byte,
                    operation$status          byte,
                    interleave                byte,
                    format$gap                byte,
                    read$write$gap            byte,
                    index                     byte,
                    drive                     byte,
                    density                   byte,
                    multitrack                byte,
                    sector                    byte,
                    cylinder                  byte,
                    head                      byte,              /* disk drive head */
                    tracks$per$disk           byte,
                    sectors$per$track         byte,
                    bytes$per$sector$code     byte,
                    bytes$per$sector          word;              /* number of bytes in a sector on the disk */

   7   1          declare
                    /* read and write buffers */
                    fmtblk(104)               byte public,
                    wrbuf(1024)               byte public,
                    rdbuf(1024)               byte public;

   8   1          declare
                    /* disk format initialization tables */
                    sec$trk$table(3)          byte data(26,15,8),
                    fmt$gap$table(8)          byte data(1BH,2AH,3AH,0,0,36H,54H,74H),
                    rd$wr$gap$table(8)        byte data(07H,0EH,1BH,0,0,0EH,1BH,35H);
```

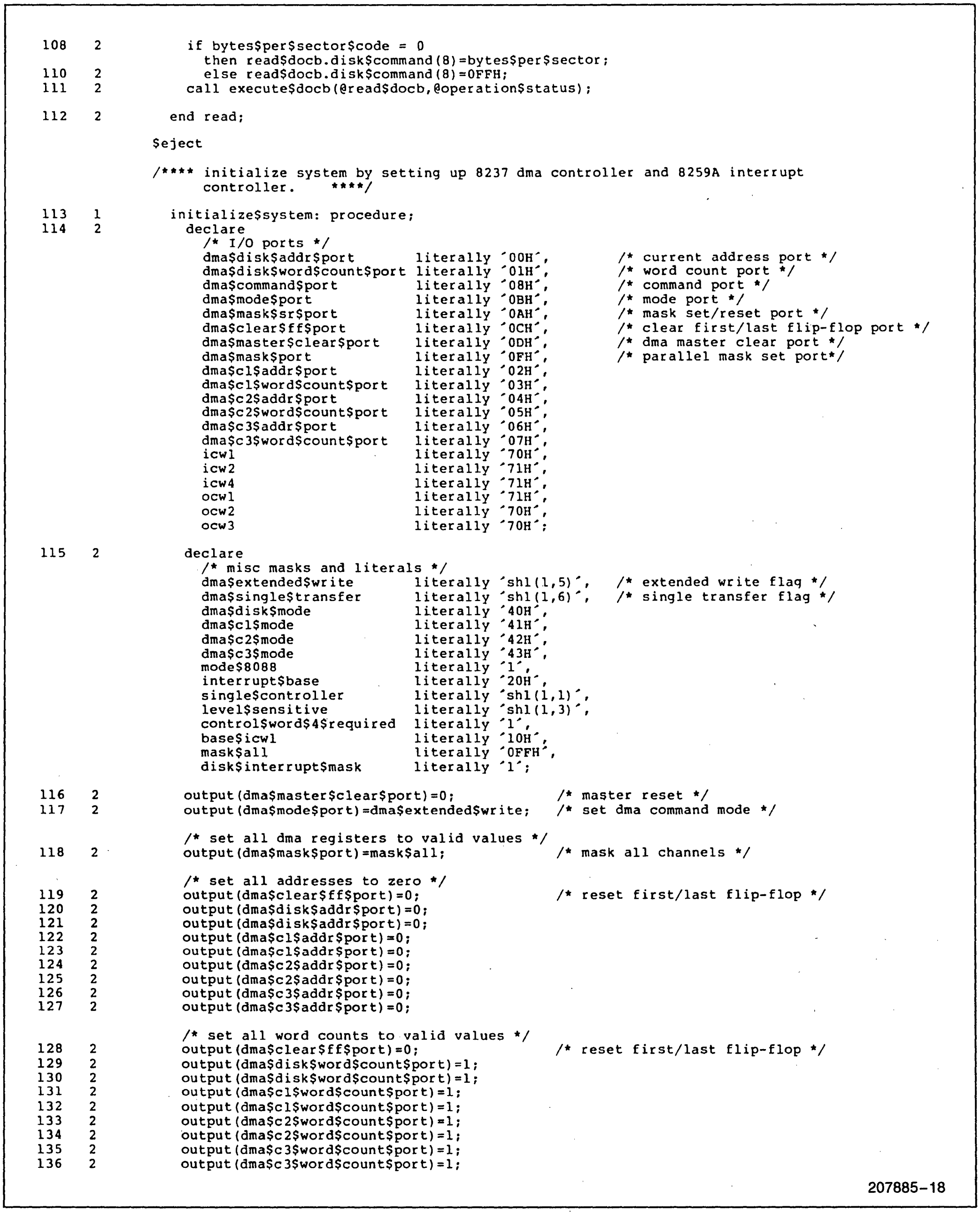
207885-15

```
  9   1         declare
                  /* external pointer tables and interrupt vector */
                  rdbptr(2)           word external,
                  wrbptr(2)           word external,
                  fbptr(2)            word external,
                  intptr(2)           word external,
                  intvec(80H)         word external;


 10   1         execute$docb: procedure(docb$ptr,status$ptr) external;
 11   2           declare docb$ptr pointer, status$ptr pointer;
 12   2         end execute$docb;

 13   1         initialize$drivers: procedure external;
 14   2         end initialize$drivers;

              $eject

              /**** specify step rate ("srt"), head load time ("hlt"), head unload time ("hut"),
                    and dma or non-dma operation ("nd").   ****/

 15   1         specify: procedure(srt,hlt,hut,nd);
 16   2           declare (srt,hlt,hut,nd) byte;

 17   2           specify$docb.dma$op=dma$noop;
 18   2           specify$docb.disk$command(0)=specify$command;
 19   2           specify$docb.disk$command(1)=shl((not srt)+1,4) or shr(hut,4);
 20   2           specify$docb.disk$command(2)=(hlt and 0FEH) or (nd and 1);
 21   2           call execute$docb(@specify$docb,@operation$status);

 22   2         end specify;


              /****  recalibrate disk drive
                     8272 automatically steps out until the track 0 signal is activated
                     by the disk drive.   ****/

 23   1         recalibrate: procedure(drv);
 24   2           declare drv byte;

 25   2           recalibrate$docb.dma$op=dma$noop;
 26   2           recalibrate$docb.disk$command(0)=recalibrate$command;
 27   2           recalibrate$docb.disk$command(1)=drv;
 28   2           call execute$docb(@recalibrate$docb,@operation$status);

 29   2         end recalibrate;


              /**** seek drive "drv", head (side) "hd" to cylinder "cyl".   ****/

 30   1         seek: procedure(drv,cyl,hd);
 31   2           declare (drv,cyl,hd) byte;

 32   2           seek$docb.dma$op=dma$noop;
 33   2           seek$docb.disk$command(0)=seek$command;
 34   2           seek$docb.disk$command(1)=drv or shl(hd,2);
 35   2           seek$docb.disk$command(2)=cyl;
 36   2           call execute$docb(@seek$docb,@operation$status);

 37   2         end seek;


              /**** format a complete side ("head") of a single floppy disk in drive "drv".  The density,
                    (single or double) is specified by flag "dens".   ****/

 38   1         format: procedure(drv,dens,intlve);
                /*  format disk */
 39   2           declare (drv,dens,intlve) byte;
 40   2           declare physical$sector byte;

 41   2           call recalibrate(drv);
 42   2           do cylinder=0 to tracks$per$disk-1;
                     /* set sector numbers in format block to zero before computing interleave */

 43   3             do physical$sector=1 to sectors$per$track; fmtblk((physical$sector-1)*4+2)=0; end;
                     /* physical sector 1 equals logical sector 1 */
 46   3             physical$sector=1;

                     /* assign interleaved sectors */
 47   3             do sector=1 to sectors$per$track;
 48   4               index=(physical$sector-1)*4;
```

207885-16

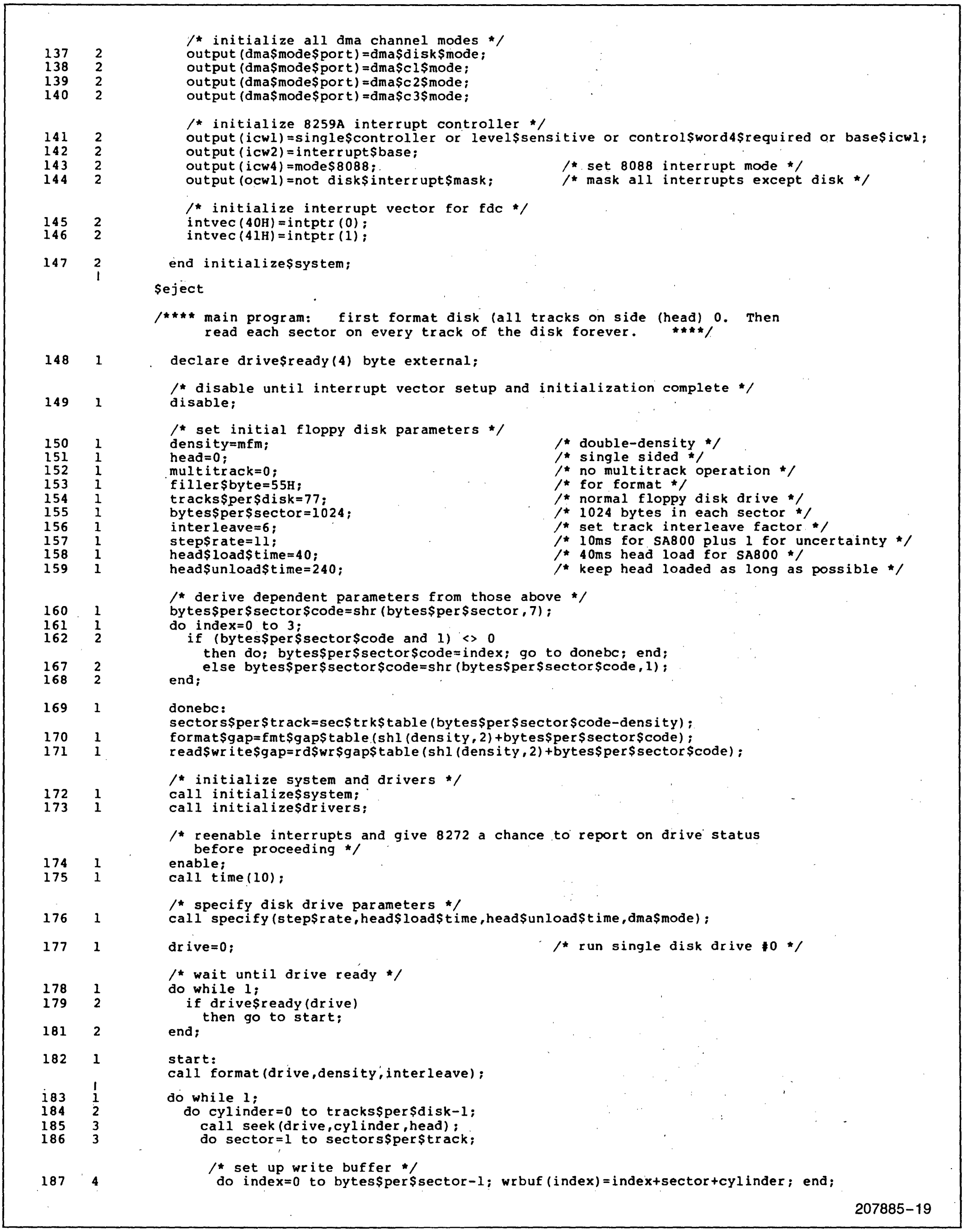
```
                      /* change sector and index if sector has already been assigned */
  49   4              do while fmtblk(index+2) <> 0; index=index+4; physical$sector=physical$sector+1; end;

                      /* set cylinder, head, sector, and size code for current sector into table */
  53   4              fmtblk(index)=cylinder;
  54   4              fmtblk(index+1)=head;
  55   4              fmtblk(index+2)=sector;
  56   4              fmtblk(index+3)=bytes$per$sector$code;

                      /* update physical sector number by interleave */
  57   4              physical$sector=physical$sector+intlve;
  58   4              if physical$sector > sectors$per$track
                        then physical$sector=physical$sector-sectors$per$track;
  60   4            end;

                    /* seek to next cylinder */
  61   3            call seek(drv,cylinder,head);

                    /* set up format control block */
  62   3            format$docb.dma$op=dma$write;
  63   3            format$docb.dma$addr=fbptr(0)+shl(fbptr(1),4);
  64   3            format$docb.dma$addr$ext=0;
  65   3            format$docb.dma$count=sectors$per$track*4-1;
  66   3            format$docb.disk$command(0)=format$command or shl(dens,6);
  67   3            format$docb.disk$command(1)=drv or shl(head,2);
  68   3            format$docb.disk$command(2)=bytes$per$sector$code;
  69   3            format$docb.disk$command(3)=sectors$per$track;
  70   3            format$docb.disk$command(4)=format$gap;
  71   3            format$docb.disk$command(5)=filler$byte;
  72   3            call execute$docb(@format$docb,@operation$status);
  73   3          end;

  74   2        end format;


              /**** write sector "sec" on drive "drv" at head "hd" and cylinder "cyl".  The
                    disk recording density is specified by the "dens" flag.  Data is expected to be
                    in the global write buffer ("wrbuf").    ****/

  75   1        write: procedure(drv,cyl,hd,sec,dens);
  76   2          declare (drv,cyl,hd,sec,dens) byte;

  77   2          write$docb.dma$op=dma$write;
  78   2          write$docb.dma$addr=wrbptr(0)+shl(wrbptr(1),4);
  79   2          write$docb.dma$addr$ext=0;
  80   2          write$docb.dma$count=bytes$per$sector-1;
  81   2          write$docb.disk$command(0)=write$command or shl(dens,6) or shl(multitrack,7);
  82   2          write$docb.disk$command(1)=drv or shl(hd,2);
  83   2          write$docb.disk$command(2)=cyl;
  84   2          write$docb.disk$command(3)=hd;
  85   2          write$docb.disk$command(4)=sec;
  86   2          write$docb.disk$command(5)=bytes$per$sector$code;
  87   2          write$docb.disk$command(6)=sectors$per$track;
  88   2          write$docb.disk$command(7)=read$write$gap;
  89   2          if bytes$per$sector$code = 0
                    then write$docb.disk$command(8)=bytes$per$sector;
  91   2            else write$docb.disk$command(8)=0FFH;
  92   2          call execute$docb(@write$docb,@operation$status);

  93   2        end write;


              /**** read sector "sec" on drive "drv" at head "hd" and cylinder "cyl".  The
                    disk recording density is defined by the "dens" flag.  Data is read into
                    the global read buffer ("rdbuf").    ****/

  94   1        read: procedure(drv,cyl,hd,sec,dens);
  95   2          declare (drv,cyl,hd,sec,dens) byte;

  96   2          read$docb.dma$op=dma$read;
  97   2          read$docb.dma$addr=rdbptr(0)+shl(rdbptr(1),4);
  98   2          read$docb.dma$addr$ext=0;
  99   2          read$docb.dma$count=bytes$per$sector-1;
 100   2          read$docb.disk$command(0)=read$command or shl(dens,6) or shl(multitrack,7);
 101   2          read$docb.disk$command(1)=drv or shl(hd,2);
 102   2          read$docb.disk$command(2)=cyl;
 103   2          read$docb.disk$command(3)=hd;
 104   2          read$docb.disk$command(4)=sec;
 105   2          read$docb.disk$command(5)=bytes$per$sector$code;
 106   2          read$docb.disk$command(6)=sectors$per$track;
 107   2          read$docb.disk$command(7)=read$write$gap;
```

207885-17



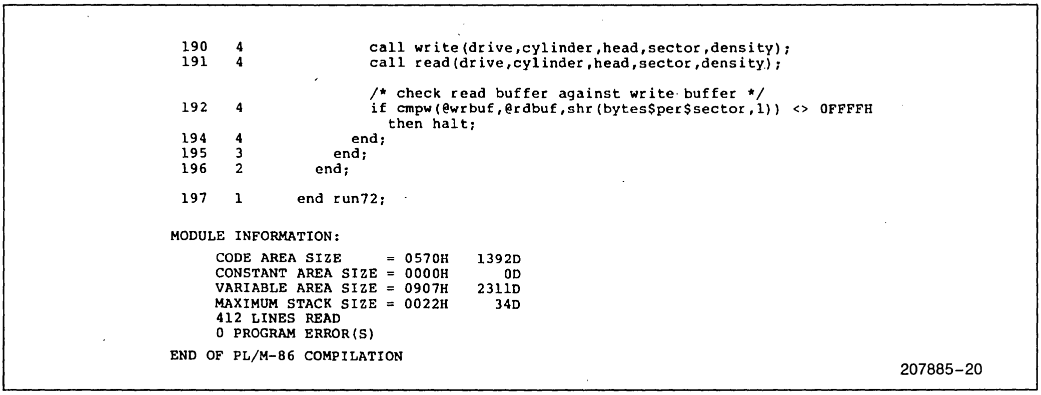
```
108   2         if bytes$per$sector$code = 0
                  then read$docb.disk$command(8)=bytes$per$sector;
110   2           else read$docb.disk$command(8)=0FFH;
111   2         call execute$docb(@read$docb,@operation$status);

112   2       end read;

            $eject

            /**** initialize system by setting up 8237 dma controller and 8259A interrupt
                  controller.    ****/

113   1       initialize$system: procedure;
114   2         declare
                  /* I/O ports */
                  dma$disk$addr$port          literally '00H',       /* current address port */
                  dma$disk$word$count$port    literally '01H',       /* word count port */
                  dma$command$port            literally '08H',       /* command port */
                  dma$mode$port               literally '0BH',       /* mode port */
                  dma$mask$sr$port            literally '0AH',       /* mask set/reset port */
                  dma$clear$ff$port           literally '0CH',       /* clear first/last flip-flop port */
                  dma$master$clear$port       literally '0DH',       /* dma master clear port */
                  dma$mask$port               literally '0FH',       /* parallel mask set port*/
                  dma$cl$addr$port            literally '02H',
                  dma$cl$word$count$port      literally '03H',
                  dma$c2$addr$port            literally '04H',
                  dma$c2$word$count$port      literally '05H',
                  dma$c3$addr$port            literally '06H',
                  dma$c3$word$count$port      literally '07H',
                  icwl                        literally '70H',
                  icw2                        literally '71H',
                  icw4                        literally '71H',
                  ocwl                        literally '71H',
                  ocw2                        literally '70H',
                  ocw3                        literally '70H';

115   2         declare
                  /* misc masks and literals */
                  dma$extended$write          literally 'shl(1,5)',  /* extended write flag */
                  dma$single$transfer         literally 'shl(1,6)',  /* single transfer flag */
                  dma$disk$mode               literally '40H',
                  dma$cl$mode                 literally '41H',
                  dma$c2$mode                 literally '42H',
                  dma$c3$mode                 literally '43H',
                  mode$8088                   literally '1',
                  interrupt$base              literally '20H',
                  single$controller           literally 'shl(1,1)',
                  level$sensitive             literally 'shl(1,3)',
                  control$word$4$required     literally '1',
                  base$icwl                   literally '10H',
                  mask$all                    literally '0FFH',
                  disk$interrupt$mask         literally '1';

116   2         output(dma$master$clear$port)=0;          /* master reset */
117   2         output(dma$mode$port)=dma$extended$write; /* set dma command mode */

                /* set all dma registers to valid values */
118   2         output(dma$mask$port)=mask$all;           /* mask all channels */

                /* set all addresses to zero */
119   2         output(dma$clear$ff$port)=0;              /* reset first/last flip-flop */
120   2         output(dma$disk$addr$port)=0;
121   2         output(dma$disk$addr$port)=0;
122   2         output(dma$cl$addr$port)=0;
123   2         output(dma$cl$addr$port)=0;
124   2         output(dma$c2$addr$port)=0;
125   2         output(dma$c2$addr$port)=0;
126   2         output(dma$c3$addr$port)=0;
127   2         output(dma$c3$addr$port)=0;

                /* set all word counts to valid values */
128   2         output(dma$clear$ff$port)=0;              /* reset first/last flip-flop */
129   2         output(dma$disk$word$count$port)=1;
130   2         output(dma$disk$word$count$port)=1;
131   2         output(dma$cl$word$count$port)=1;
132   2         output(dma$cl$word$count$port)=1;
133   2         output(dma$c2$word$count$port)=1;
134   2         output(dma$c2$word$count$port)=1;
135   2         output(dma$c3$word$count$port)=1;
136   2         output(dma$c3$word$count$port)=1;
```

207885-18



```
                 /* initialize all dma channel modes */
137   2          output(dma$mode$port)=dma$disk$mode;
138   2          output(dma$mode$port)=dma$c1$mode;
139   2          output(dma$mode$port)=dma$c2$mode;
140   2          output(dma$mode$port)=dma$c3$mode;

                 /* initialize 8259A interrupt controller */
141   2          output(icw1)=single$controller or level$sensitive or control$word4$required or base$icw1;
142   2          output(icw2)=interrupt$base;
143   2          output(icw4)=mode$8088;                       /* set 8088 interrupt mode */
144   2          output(ocw1)=not disk$interrupt$mask;         /* mask all interrupts except disk */

                 /* initialize interrupt vector for fdc */
145   2          intvec(40H)=intptr(0);
146   2          intvec(41H)=intptr(1);

147   2        end initialize$system;

             $eject

             /**** main program:   first format disk (all tracks on side (head) 0.  Then
                   read each sector on every track of the disk forever.    ****/

148   1        declare drive$ready(4) byte external;

               /* disable until interrupt vector setup and initialization complete */
149   1        disable;

               /* set initial floppy disk parameters */
150   1        density=mfm;                                   /* double-density */
151   1        head=0;                                        /* single sided */
152   1        multitrack=0;                                  /* no multitrack operation */
153   1        filler$byte=55H;                               /* for format */
154   1        tracks$per$disk=77;                            /* normal floppy disk drive */
155   1        bytes$per$sector=1024;                         /* 1024 bytes in each sector */
156   1        interleave=6;                                  /* set track interleave factor */
157   1        step$rate=11;                                  /* 10ms for SA800 plus 1 for uncertainty */
158   1        head$load$time=40;                             /* 40ms head load for SA800 */
159   1        head$unload$time=240;                          /* keep head loaded as long as possible */

               /* derive dependent parameters from those above */
160   1        bytes$per$sector$code=shr(bytes$per$sector,7);
161   1        do index=0 to 3;
162   2          if (bytes$per$sector$code and 1) <> 0
                   then do; bytes$per$sector$code=index; go to donebc; end;
167   2            else bytes$per$sector$code=shr(bytes$per$sector$code,1);
168   2        end;

169   1        donebc:
               sectors$per$track=sec$trk$table(bytes$per$sector$code-density);
170   1        format$gap=fmt$gap$table(shl(density,2)+bytes$per$sector$code);
171   1        read$write$gap=rd$wr$gap$table(shl(density,2)+bytes$per$sector$code);

               /* initialize system and drivers */
172   1        call initialize$system;
173   1        call initialize$drivers;

               /* reenable interrupts and give 8272 a chance to report on drive status
                  before proceeding */
174   1        enable;
175   1        call time(10);

               /* specify disk drive parameters */
176   1        call specify(step$rate,head$load$time,head$unload$time,dma$mode);

177   1        drive=0;                                       /* run single disk drive #0 */

               /* wait until drive ready */
178   1        do while 1;
179   2          if drive$ready(drive)
                   then go to start;
181   2        end;

182   1        start:
               call format(drive,density,interleave);

183   1        do while 1;
184   2          do cylinder=0 to tracks$per$disk-1;
185   3            call seek(drive,cylinder,head);
186   3            do sector=1 to sectors$per$track;

                     /* set up write buffer */
187   4              do index=0 to bytes$per$sector-1; wrbuf(index)=index+sector+cylinder; end;
```

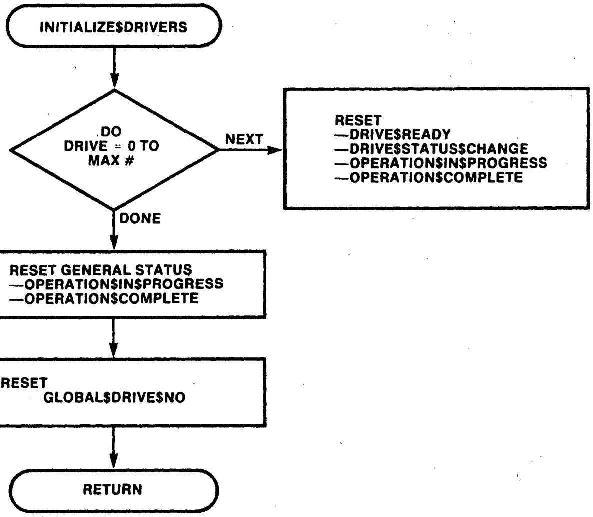
207885-19



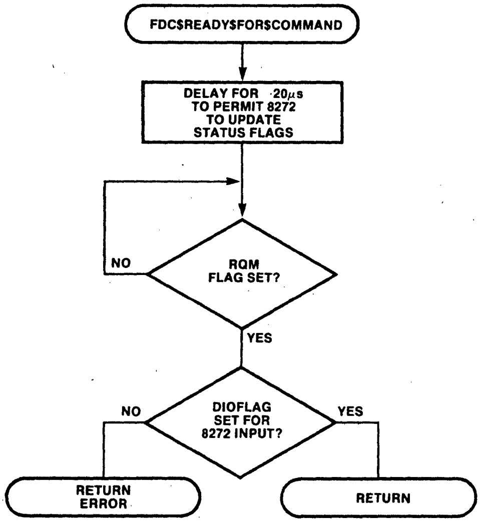
```
 190   4             call write(drive,cylinder,head,sector,density);
 191   4             call read(drive,cylinder,head,sector,density);

                     /* check read buffer against write buffer */
 192   4             if cmpw(@wrbuf,@rdbuf,shr(bytes$per$sector,1)) <> OFFFFH
                       then halt;
 194   4           end;
 195   3         end;
 196   2       end;

 197   1     end run72;

MODULE INFORMATION:

     CODE AREA SIZE     = 0570H   1392D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0907H   2311D
     MAXIMUM STACK SIZE = 0022H     34D
     412 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

207885-20

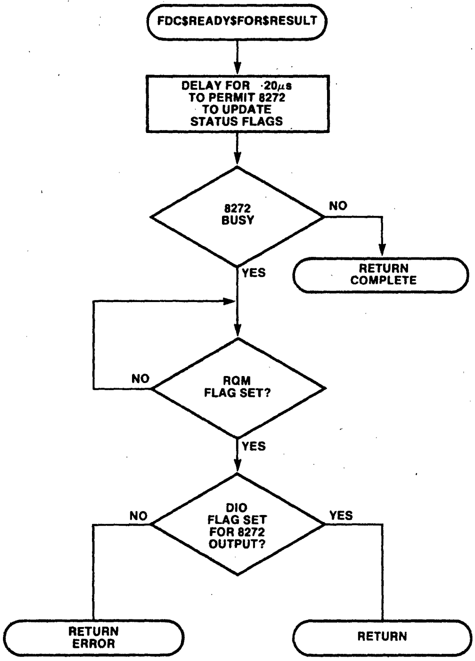
APPENDIX C
8272 DRIVER FLOWCHARTS



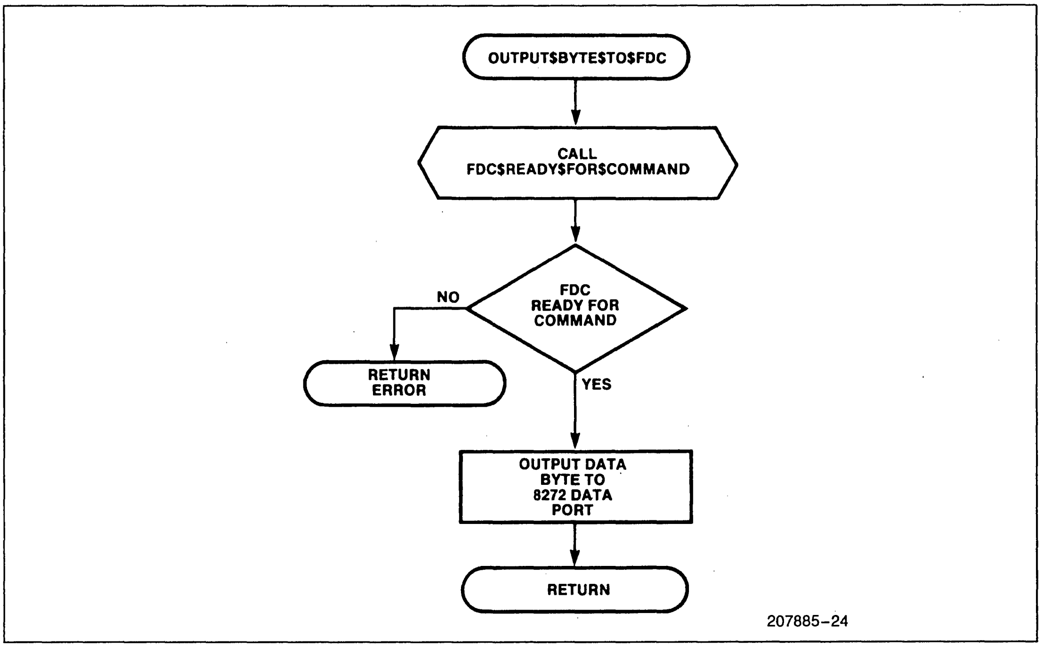
207885-21



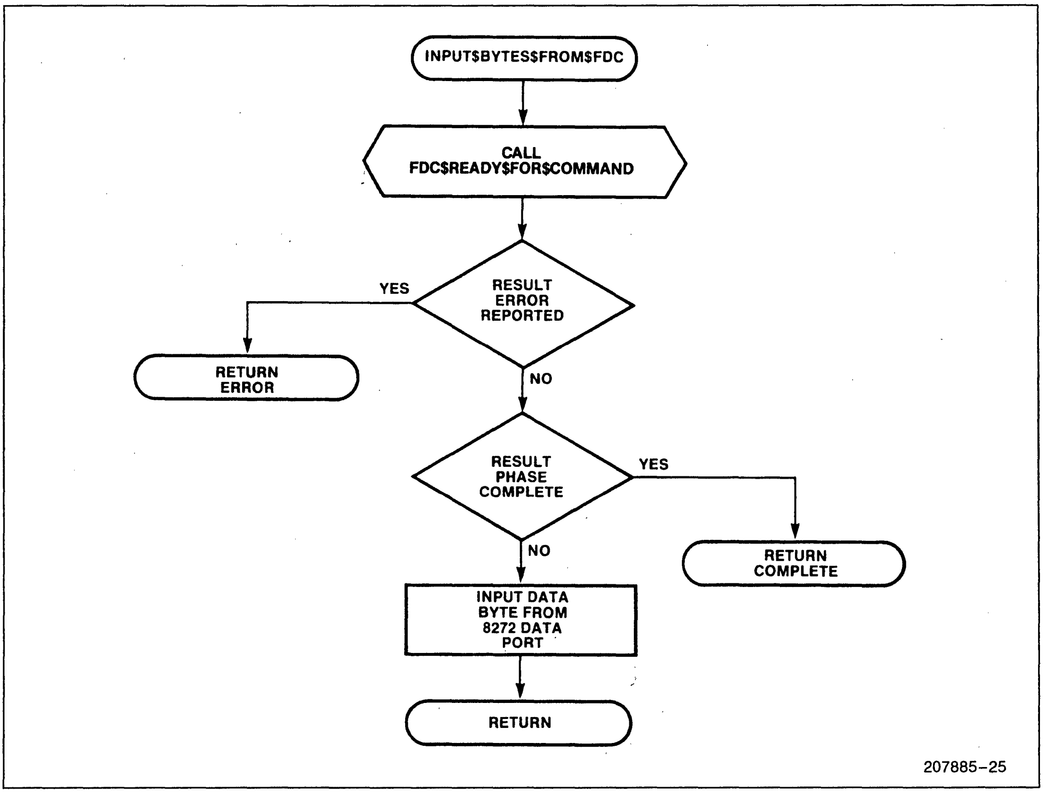
207885-22



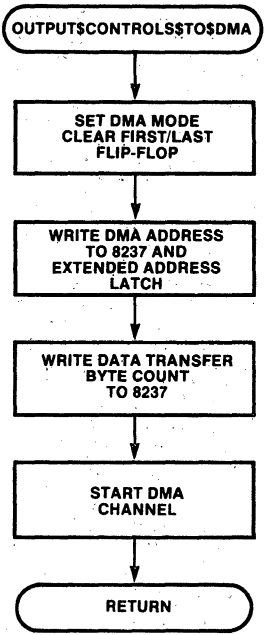
207885-23



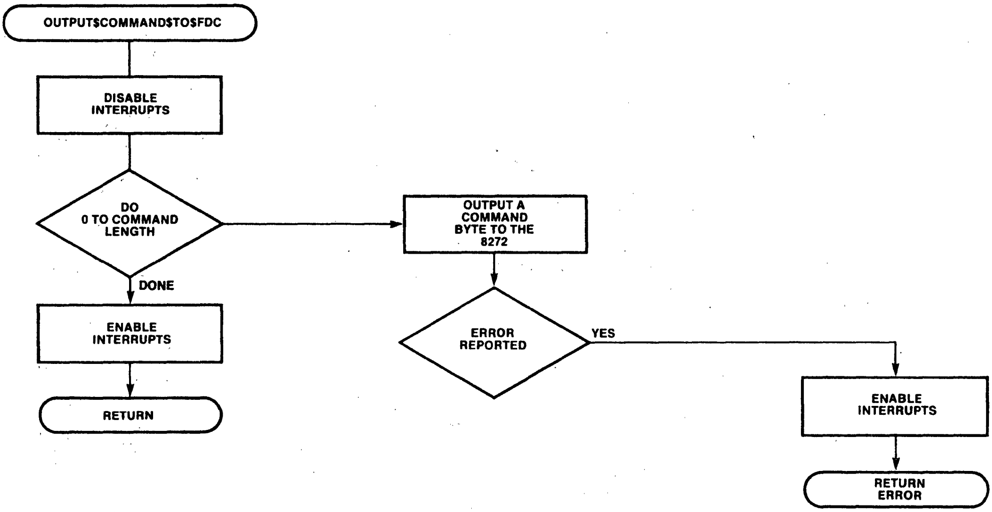
207885-24



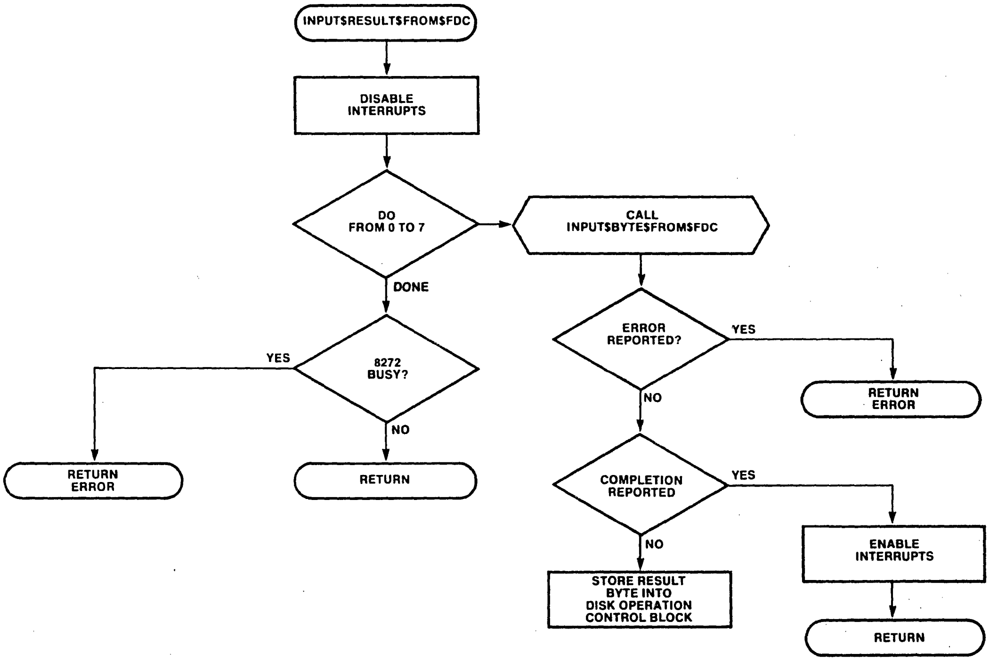
207885-25


OUTPUT$CONTROLS$TO$DMA
SET DMA MODE CLEAR FIRST/LAST FLIP-FLOP
WRITE DMA ADDRESS TO 8237 AND EXTENDED ADDRESS LATCH
WRITE DATA TRANSFER BYTE COUNT TO 8237
START DMA CHANNEL
RETURN

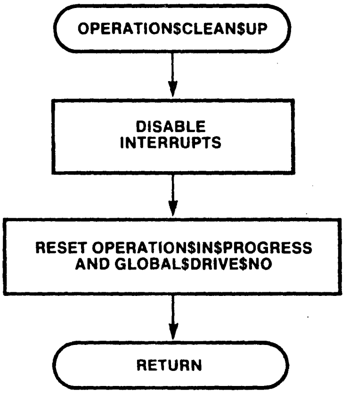
207885-26



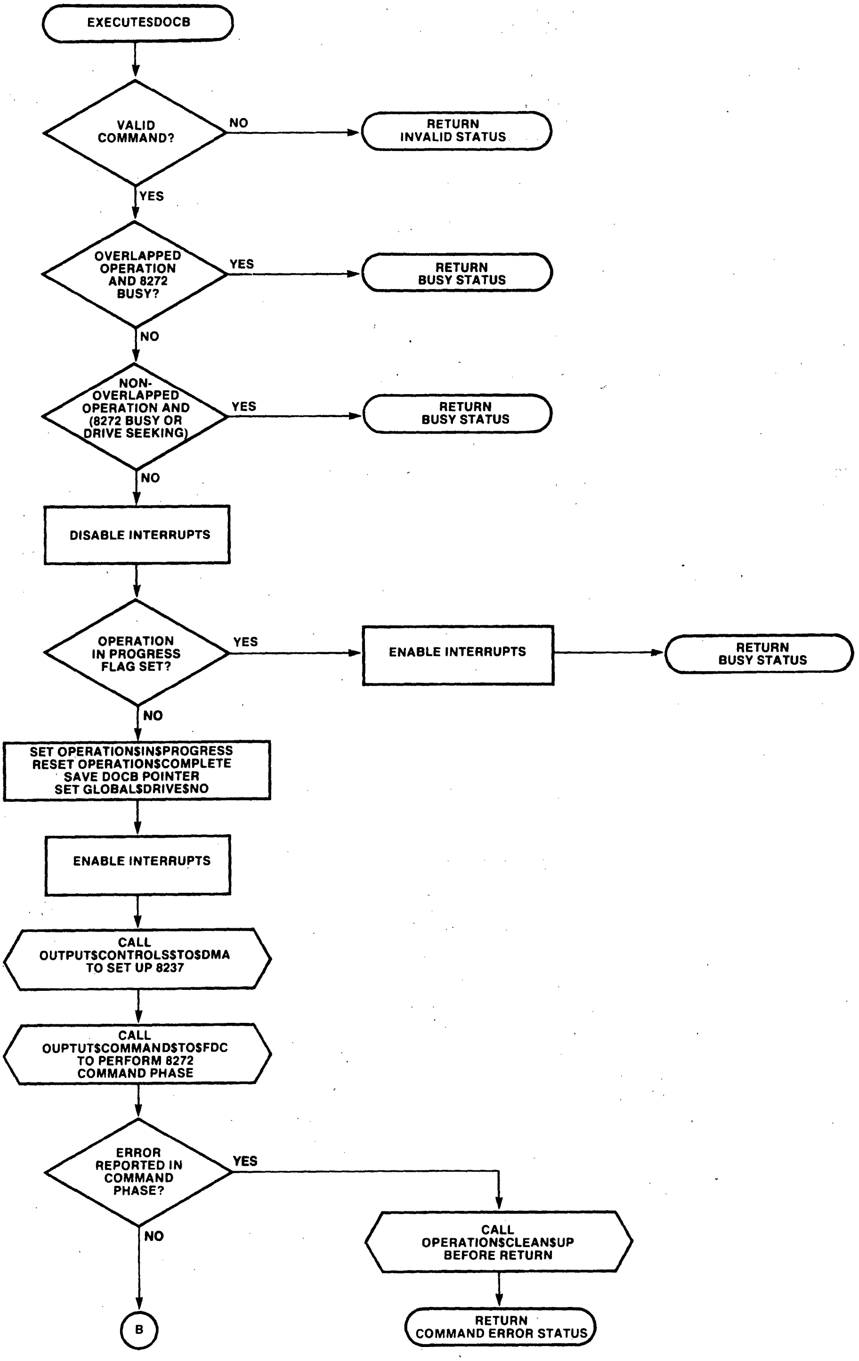
207885-27



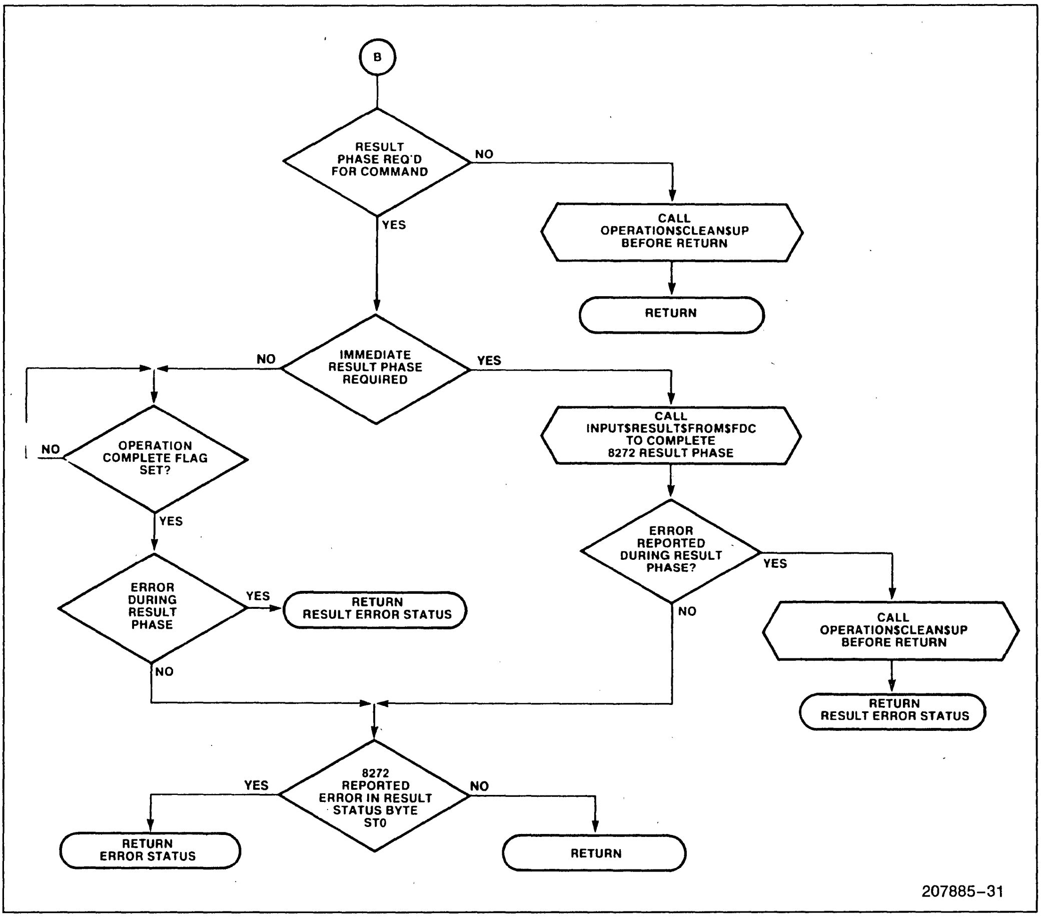
207885–28



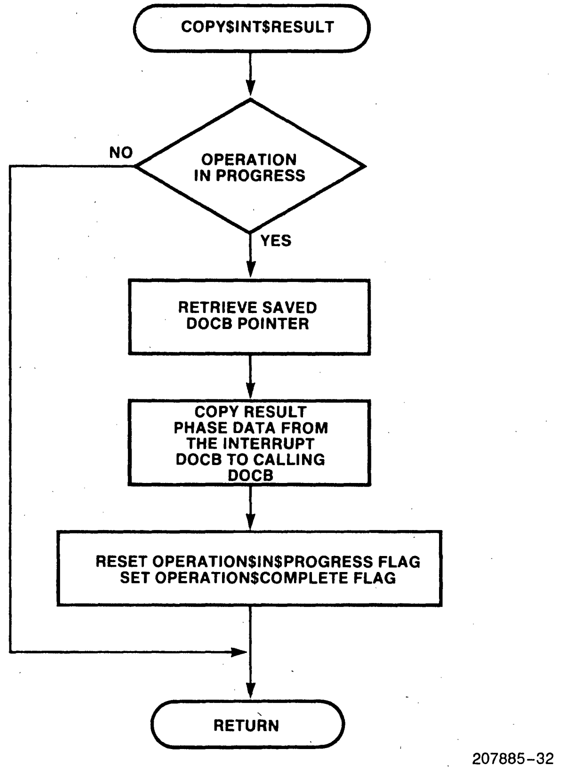
207885–29

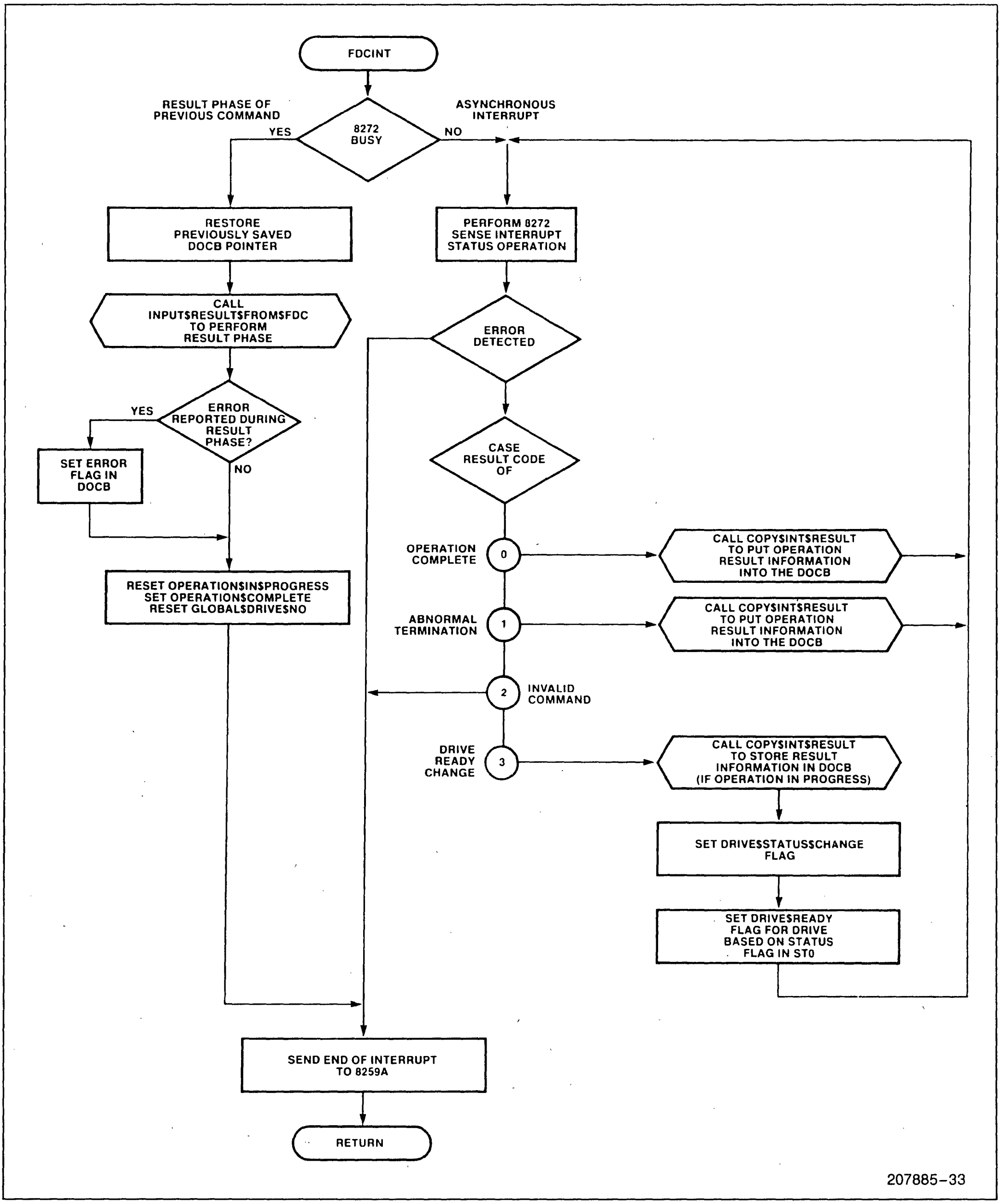

EXECUTE$DOCB
VALID COMMAND?
NO
RETURN INVALID STATUS
YES
OVERLAPPED OPERATION AND 8272 BUSY?
YES
RETURN BUSY STATUS
NO
NON-OVERLAPPED OPERATION AND (8272 BUSY OR DRIVE SEEKING)
YES
RETURN BUSY STATUS
NO
DISABLE INTERRUPTS
OPERATION IN PROGRESS FLAG SET?
YES
ENABLE INTERRUPTS
RETURN BUSY STATUS
NO
SET OPERATION$IN$PROGRESS
RESET OPERATION$COMPLETE
SAVE DOCB POINTER
SET GLOBAL$DRIVE$NO
ENABLE INTERRUPTS
CALL OUTPUT$CONTROLS$TO$DMA TO SET UP 8237
CALL OUPTUT$COMMAND$TO$FDC TO PERFORM 8272 COMMAND PHASE
ERROR REPORTED IN COMMAND PHASE?
YES
CALL OPERATION$CLEAN$UP BEFORE RETURN
RETURN COMMAND ERROR STATUS
NO
B

207885–30


B
RESULT PHASE REQ'D FOR COMMAND
NO
YES
CALL OPERATION$CLEAN$UP BEFORE RETURN
RETURN
IMMEDIATE RESULT PHASE REQUIRED
NO
YES
CALL INPUT$RESULT$FROM$FDC TO COMPLETE 8272 RESULT PHASE
NO
OPERATION COMPLETE FLAG SET?
YES
ERROR REPORTED DURING RESULT PHASE?
YES
NO
CALL OPERATION$CLEAN$UP BEFORE RETURN
RETURN RESULT ERROR STATUS
ERROR DURING RESULT PHASE
YES
RETURN RESULT ERROR STATUS
NO
8272 REPORTED ERROR IN RESULT STATUS BYTE ST0
YES
NO
RETURN ERROR STATUS
RETURN

207885–31


COPY$INT$RESULT
OPERATION IN PROGRESS
NO
YES
RETRIEVE SAVED DOCB POINTER
COPY RESULT PHASE DATA FROM THE INTERRUPT DOCB TO CALLING DOCB
RESET OPERATION$IN$PROGRESS FLAG
SET OPERATION$COMPLETE FLAG
RETURN
207885-32


FDCINT
RESULT PHASE OF PREVIOUS COMMAND
ASYNCHRONOUS INTERRUPT
YES
8272 BUSY
NO
RESTORE PREVIOUSLY SAVED DOCB POINTER
PERFORM 8272 SENSE INTERRUPT STATUS OPERATION
CALL INPUT$RESULT$FROM$FDC TO PERFORM RESULT PHASE
ERROR DETECTED
YES
ERROR REPORTED DURING RESULT PHASE?
NO
SET ERROR FLAG IN DOCB
CASE RESULT CODE OF
OPERATION COMPLETE
0
CALL COPY$INT$RESULT TO PUT OPERATION RESULT INFORMATION INTO THE DOCB
RESET OPERATION$IN$PROGRESS SET OPERATION$COMPLETE RESET GLOBAL$DRIVE$NO
ABNORMAL TERMINATION
1
CALL COPY$INT$RESULT TO PUT OPERATION RESULT INFORMATION INTO THE DOCB
2
INVALID COMMAND
DRIVE READY CHANGE
3
CALL COPY$INT$RESULT TO STORE RESULT INFORMATION IN DOCB (IF OPERATION IN PROGRESS)
SET DRIVE$STATUS$CHANGE FLAG
SET DRIVE$READY FLAG FOR DRIVE BASED ON STATUS FLAG IN ST0
SEND END OF INTERRUPT TO 8259A
RETURN
207885-33

Hard Disk Controllers 8

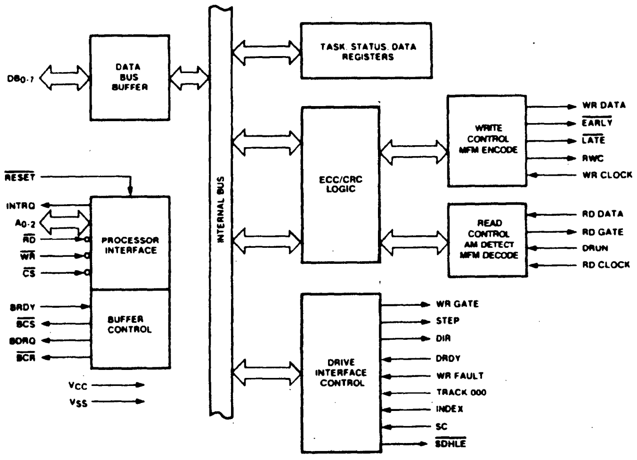
PRELIMINARY

82064
CHMOS WINCHESTER DISK CONTROLLER WITH ON-CHIP ERROR DETECTION AND CORRECTION

- Controls ST506/ST412 Interface Winchester Disk Drives
- 5 Mbit/sec Data Transfer Rate
- Compatible with All Intel and Most Other Microprocessors
- High Speed Operation
  - "Zero Wait State" Operation with 8 MHz 80286 and 10 MHz 80186/188
  - "One Wait State" Operation with 10 MHz 80286
- Eight High-Level Commands: Restore, Seek, Read Sector, Write Sector, Scan ID, Write Format, Compute Correction, Set Parameter
- Low Power CHMOS III
- On-Chip ECC Unit Automatically Corrects Errors
- 5 or 11-Bit Correction—Span Software Selectable
- Implied Seeks with Read/Write Commands
- Multiple Sector Transfer Capability
- 128, 256, 512 and 1024 Byte Sector Lengths
- Available in 40-Lead Ceramic Dual In-Line, 40-Lead Plastic Dual In-Line, and 44-Lead Plastic Chip Carrier Packages
  (See Packaging Spec., Order #231369)

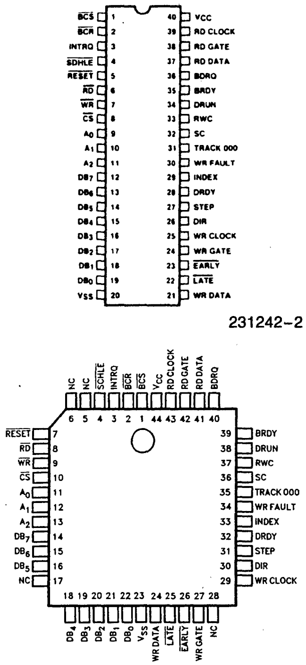
The 82064 Winchester Disk Controller (WDC) with on-chip error detection and correction circuitry interfaces microprocessor systems to 5¼" Winchester disk drives. The 82064 is a CHMOS version of the Western Digital WD2010. It is an upgrade to the Western Digital WD1010A-05 Winchester Disk Controller, and includes on-chip ECC, support for drives with up to 2k tracks, and has an additional control signal which eliminates an external decoder.

The 82064 is fabricated on Intel's advanced CHMOS III technology and is available in 40-lead CERDIP, plastic DIP, and 44-lead plastic leaded chip carrier packages.



231242-1

Figure 1. 82064 Block Diagram



231242-2

231242-29

Figure 2. 82064 Pinouts

Table 1. Pin Description

Symbol	Pin No.		Type	Name and Function
	DIP	PLCC		
$\overline{\text{BCS}}$	1	1	O	**BUFFER CHIP SELECT:** Output used to enable reading or writing of the external sector buffer by the 82064. When low, the host should not be able to drive the 82064 data bus, $\overline{\text{RD}}$, or $\overline{\text{WR}}$ lines.
$\overline{\text{BCR}}$	2	2	O	**BUFFER COUNTER RESET:** Output that is asserted by the 82064 prior to read/write operation. This pin is asserted whenever $\overline{\text{BCS}}$ changes state. Used to reset the address counter of the buffer memory.
INTRQ	3	3	O	**INTERRUPT REQUEST:** Interrupt generated by the 82064 upon command termination. It is reset when the STATUS register is read, or a new command is written to the COMMAND register. Optionally signifies when a data transfer is required on Read Sector commands.
$\overline{\text{SDHLE}}$	4	4	O	$\overline{\text{SDHLE}}$ is asserted when the SDH register is written by the host.
$\overline{\text{RESET}}$	5	7	I	**RESET:** Initializes the controller and clears all status flags. Does not clear the Task Register File.
$\overline{\text{RD}}$	6	8	I/O	**READ:** Tri-state, bi-directional signal. As an input, $\overline{\text{RD}}$ controls the transfer of information from the 82064 registers to the host. $\overline{\text{RD}}$ is an output when the 82064 is reading data from the sector buffer ($\overline{\text{BCS}}$ low).
$\overline{\text{WR}}$	7	9	I/O	**WRITE:** Tri-state, bi-directional signal. As an input, $\overline{\text{WR}}$ controls the transfer of command or task information into the 82064 registers. $\overline{\text{WR}}$ is an output when the 82064 is writing data to the sector buffer ($\overline{\text{BCS}}$ low).
$\overline{\text{CS}}$	8	10	I	**CHIP SELECT:** Enables $\overline{\text{RD}}$ and $\overline{\text{WR}}$ as inputs for access to the Task Registers. It has no effect once a disk command starts.
A_{0-2}	9–11	11–13	I	**ADDRESS:** Used to select a register from the task register file.
DB_{0-7}	12–19	14–16 18–22	I/O	**DATA BUS:** Tri-state, bi-directional 8-bit Data Bus with control determined by BCS. When BCS is high the microprocessor has full control of the data bus for reading and writing the Task Register File. When BCS is low the 82064 controls the data bus to transfer to or from the buffer.
V_{SS}	20	23		Ground
WR DATA	21	24	O	**WRITE DATA:** Output that shifts out MFM data at a rate determined by Write Clock. Requires an external D flip-flop clocked at 10 MHz. The output has an active pullup and pulldown that can sink 4.8 mA.
$\overline{\text{LATE}}$	22	25	O	**LATE:** Output used to derive a delay value for write precompensation. Valid when WR GATE is high. Active on all cylinders.
$\overline{\text{EARLY}}$	23	26	O	**EARLY:** Output used to derive a delay value for write precompensation. Valid when WR GATE is high. Active on all cylinders.

Table 1. Pin Description (Continued)

Symbol	Pin No.		Type	Name and Function
	DIP	PLCC		
WR GATE	24	27	O	**WRITE GATE:** High when write data is valid. WR GATE goes low if the WR FAULT input is active. This output is used by the drive to enable head write current.
WR CLOCK	25	29	I	**WRITE CLOCK:** Clock input used to derive the write data rate. Frequency = 5 MHz for the ST506 interface.
DIR	26	30	O	**DIRECTION:** High level on this output tells the drive to move the head inward (increasing cylinder number). The state of this signal is determined by the 82064's internal comparison of actual cylinder location vs. desired cylinder.
STEP	27	31	O	**STEP:** This signal is used to move the drive head to another cylinder at a programmable frequency. Pulse width = 1.6 μs for a step rate of 3.2 μs/step, and 8.4 μs for all other step rates.
DRDY	28	32	I	**DRIVE READY:** If DRDY from the drive goes low, the command will be terminated.
INDEX	29	33	I	**INDEX:** Signal from the drive indicating the beginning of a track. It is used by the 82064 during formatting, and for counting retries. Index is edge triggered. Only the rising edge is valid.
WR FAULT	30	34	I	**WRITE FAULT:** An error input to the 82064 which indicates a fault condition at the drive. If WR FAULT from the drive goes high, the command will be terminated.
TRACK 000	31	35	I	**TRACK ZERO:** Signal from the drive which indicates that the head is at the outermost cylinder. Used to verify proper completion of a RESTORE command.
SC	32	36	I	**SEEK COMPLETE:** Signal from the drive indicating to the 82064 that the drive head has settled and that reads or writes can be made. SC is edge triggered. Only the rising edge is valid.
RWC	33	37	O	**REDUCED WRITE CURRENT:** Signal goes high for all cylinder numbers above the value programmed in the Write Precomp Cylinder register. It is used by the precompensation logic and by the drive to reduce the effects of bit shifting.
DRUN	34	38	I	**DATA RUN:** This signal informs the 82064 when a field of all ones or all zeroes has been detected in the read data stream by an external one-shot. This indicates the beginning of an ID field. RD GATE is brought high when DRUN is sampled high for 16 clock periods.
BRDY	35	39	I	**BUFFER READY:** Input used to signal the controller that the buffer is ready for reading (full), or writing (empty), by the host μP. Only the rising edge indicates the condition.

Table 1. Pin Description (Continued)

Symbol	Pin No.		Type	Name and Function
	DIP	PLCC		
BDRQ	36	40	O	**BUFFER DATA REQUEST:** Activated during Read or Write commands when a data transfer between the host and the 82064's sector buffer is required. Typically used as a DMA request line.
RD DATA	37	41	I	**READ DATA:** Single ended input that accepts MFM data from the drive.
RD GATE	38	42	O	**READ GATE:** Output that is asserted when a search for an address mark is initiated. It remains asserted until the end of the ID or data field.
RD CLOCK	39	43	I	**READ CLOCK:** Clock input derived from the external data recovery circuits.
V_{CC}	40	44	I	**D.C. POWER:** +5V.
NC	—	5, 6 17, 28		**No Connects**

FUNCTIONAL DESCRIPTION

The Intel 82064 CHMOS Winchester Disk Controller (WDC) interfaces microprocessor systems to Winchester disk drives that use the Seagate Technology ST506/ST412 interface. The device translates parallel data from the microprocessor to a 5 Mbit/sec, MFM-encoded serial bit stream. It provides all of the drive control logic and control signals which simplify the design of external data separation and write precompensation circuitry. The 82064 is designed to interface to the host processor through an external sector buffer.

On-chip error detection algorithms include the CRC/CCITT and a 32-bit computer generated ECC polynomial. If the ECC code is selected, the 82064 provides three possible error handling techniques if an error is detected during a read operation:

1. Automatically correct the data in the sector buffer, providing the host with good information.
2. Provide the host with the error location and pattern, allowing the host to correct the error.
3. Take no action other than setting the error flag.

The Intel 82064 is an enhanced version of the Western Digital WD2010 Winchester Disk Controller. The 82064 has been completely redesigned for Intel's advanced CHMOS III fabrication process, allowing Intel to offer a high quality, low power device while at the same time maintaining complete compatibility with the WD2010.

Enhancements to the basic design include:

Conversion to a CHMOS III fabrication process for low power consumption.

Improvements to the processor interface to provide high-speed "zero wait state" operation with 10 MHz 80186/188 and 8 MHz 80286. High-speed "one wait state" operation with 10 MHz 80286.

The 82064 is completely socket and software compatible with the WD2010 Winchester Disk Controller. As with the WD2010, the 82064 is also socket and software compatible with existing WD1010A-05 designs that do not include external ECC.

INTERNAL ARCHITECTURE

The internal architecture of the 82064 is shown in more detail in Figure 3. It is made up of seven major blocks as described below.

PLA Controller

The PLA interprets commands and provides all control functions. It is synchronized with WR CLOCK.

Magnitude Comparator

An 11-bit magnitude comparator is used to calculate the direction and number of steps needed to move the heads from the present to the desired cylinder position. It compares the cylinder number in the task file to the internal "present position" cylinder number.

A separate high-speed equivelance comparator is used to compare ID field bytes when searching for a sector ID field.

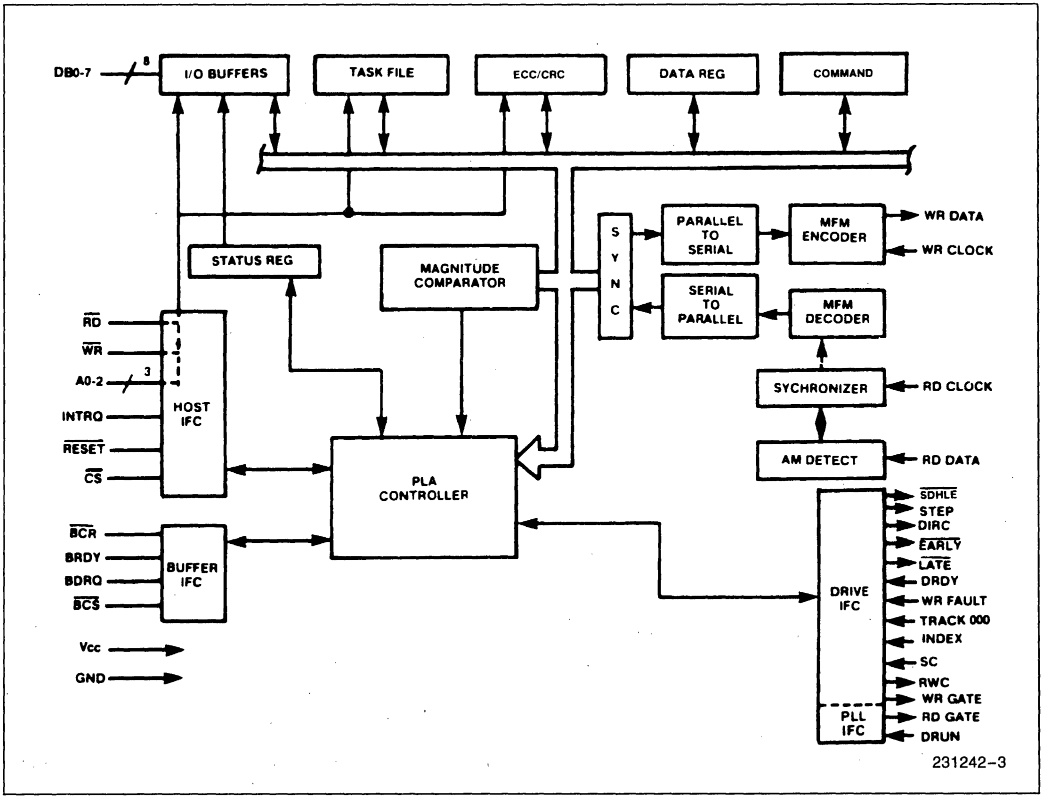


Figure 3. 82064 Detailed Block Diagram

CRC and ECC Generator and Checker

The 82064 provides two options for protecting the integrity of the data field. The data field may have either a CRC (SDH register, bit 7=0), or a 32-bit ECC (SDH register, bit 7=1) appended to it. The ID field is always protected by a CRC.

The CRC mode provides a means of verifying the accuracy of the data read from the disk, but does not attempt to correct it. The CRC generator computes and checks cyclic redundancy check characters that are written and read from the disk after ID and data fields. The polynomial used is:

$X^{16}+X^{12}+X^{5}+1$

The CRC register is preset to all one's before computation starts.

If the CRC character generated while reading the data does not equal the one previously written an error exists. If an ID field CRC error occurs the "ID not found" bit in the error register will be set. If a data field CRC error occurs the "ECC/CRC" bit in the error register will be set.

The ECC mode is only applicable to the data field. It provides the user with the ability to detect and correct errors in the data field automatically. The commands and registers which must be considered when ECC is used are:

1. SDH Register, bit 7 (CRC/ECC)
2. READ SECTOR Command, bit 0 (T)
3. READ SECTOR and WRITE SECTOR Commands, bit 1 (L)
4. COMPUTE CORRECTION Command
5. SET PARAMETER Command
6. STATUS Register, bit 2 - error correction successful
7. STATUS Register, bit 0 - error occurred
8. ERROR Register, bit 6 - uncorrectable error

To enable the ECC mode, bit 7 of the SDH register must be set to one.

Bit 0 (T) of the READ Command controls whether or not error correction is attempted. When T = 0 and an error is detected, the 82064 tries up to 10 times to correct the error. If the error is successfully corrected, bit 2 of the STATUS Register is set. The host can interrogate the status register and detect that an error occurred and was corrected. If the error was not correctable, bit 6 of the ERROR Register is set. If the correction span was set to 5 bits, the host may now execute the SET PARAMETER Command to change the correction span to 11 bits, and attempt the read again. If the error persists, the host can read the data, but it will contain errors.

When T = 1 and an error is detected, no attempt is made to correct it. Bit 0 of the STATUS Register and bit 6 of the ERROR Register are set. The user now has two choices:

1. Ignore the error and make no attempt to correct it.
2. Use the COMPUTE CORRECTION Command to determine the location and pattern of the error, and correct it within the user's program.

When the COMPUTE CORRECTION Command is implemented, it must be done before executing any command which can alter the contents of the ECC Register. The READ SECTOR, WRITE SECTOR, SCAN ID, and FORMAT Commands will alter this register and correction will be impossible. The COMPUTE CORRECTION Command may determine that the error is uncorrectable, at which point the error bits in the STATUS and ERROR Registers are set.

Although ECC generation starts with the first bit of the F8H byte in the data ID field, the actual ECC bytes written will be the same as if the A1H byte was included. The ECC polynomial used is:

$$X^{32}+X^{28}+X^{26}+X^{19}+X^{17}+X^{10}+X^{6}+X^{2}+1$$

For automatic error correction, the external sector buffer must be implemented with a static RAM and counter, not with a FIFO.

The SET PARAMETER Command is used to select a 5-bit or 11-bit correction span.

When the L Bit (bit 1) of the READ SECTOR and WRITE SECTOR commands is set to one, they are referred to as READ LONG and WRITE LONG commands. For these commands, no CRC or ECC characters are generated or checked by the 82064. In effect, the data field is extended by 4 bytes which are passed to/from the sector buffer.

With proper use of the WRITE SECTOR, READ LONG, WRITE LONG, and READ SECTOR Commands, a diagnostic routine may be developed to test the accuracy of the error correction process.

MFM ENCODER/DECODER

Encodes and decodes MFM data to be written/read from the drive. The MFM encoder operates from WR CLOCK, a clock having a frequency equal to the bit rate. The MFM decoder operates from RD CLOCK, a bit rate clock generated by the external data separator. RD CLOCK and WR CLOCK need not be synchronous.

The MFM encoder also generates the write precompensation control signals. Depending on the bit pattern of the data, $\overline{\text{EARLY}}$ or $\overline{\text{LATE}}$ may be asserted. External circuitry uses these signals to compensate for drift caused by the influence one bit has over another. More information on the use of the $\overline{\text{EARLY}}$ and $\overline{\text{LATE}}$ control signals can be found in the section which describes the drive interface.

Address Mark (AM) Detection

An address mark is comprised of two unique bytes preceeding both the ID field and the data field. The first byte is used for resynchronization. The second byte indicates whether it is an ID field or a data field.

The first byte, A1H, normally has a clock pattern of 0EH; however, one clock pulse has been suppressed, making it 0AH. With this pattern, the AM detector knows it is looking at an address mark. It now examines the next byte to determine if it is an ID or data field. If this byte is 111101XX or 111111XX it is an ID field. Bits 3, 1, and 0 are the high order cylinder number bits. If the second byte is F8H, it is a data field.

Host/Buffer Interface Control

The primary interface between the host processor and the 82064 is an 8-bit bi-directional bus. This bus is used to transmit and receive data for both the 82064 and the sector buffer. The sector buffer consists of a static RAM and counter. Since the 82064 makes the bus active when accessing the sector buffer, a transceiver must be used to isolate the host during this time. Figure 4 illustrates a typical interface with a sector buffer. Whenever the 82064 is not using the sector buffer, the BUFFER CHIP SELECT ($\overline{\text{BCS}}$) is high (disabled). This allows the host access to the 82064's Task Register File and to the sector buffer. A decoder is used to generate $\overline{\text{BCS}}$ when A_{0-2} is '000', an unused address in the 82064. A binary counter is enabled whenever $\overline{\text{RD}}$ or $\overline{\text{WR}}$ go active. The location within the sector buffer which is addressed by the counter will be accessed. The counter will be incremented by the trailing edge of the $\overline{\text{RD}}$ or $\overline{\text{WR}}$. This allows the host to access se-

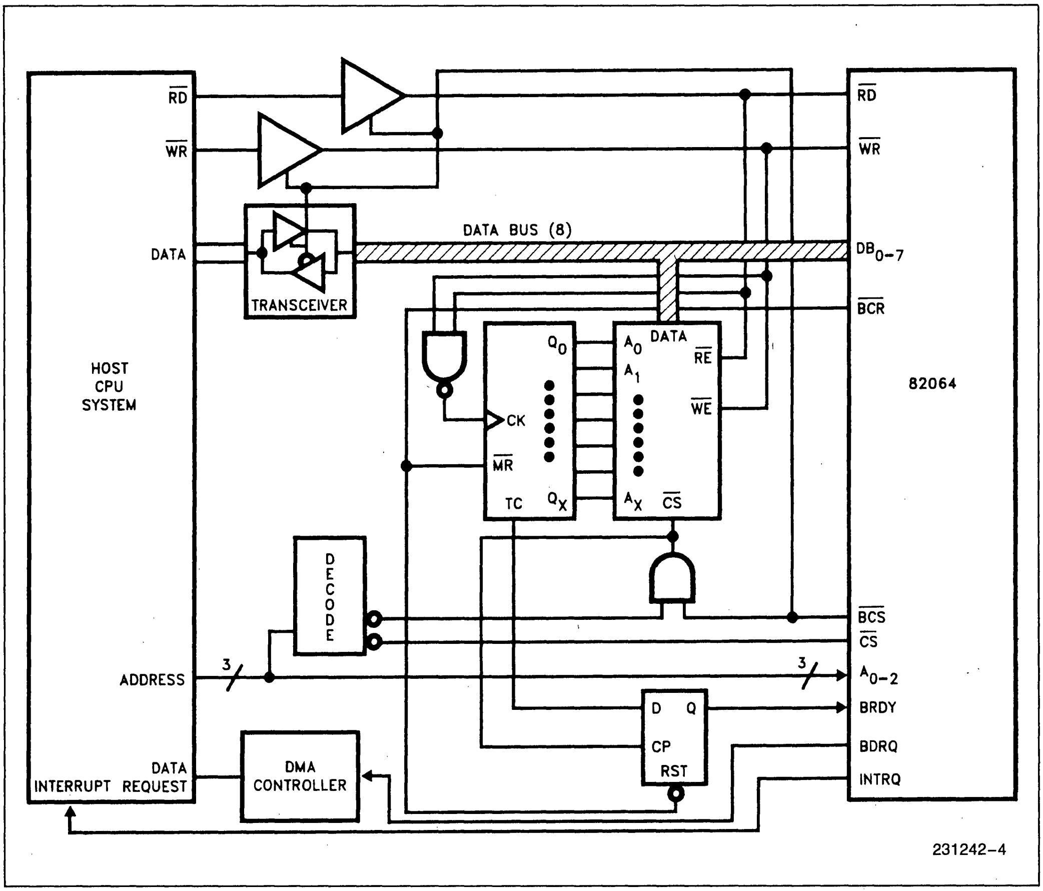


Figure 4. Host Interface Block Diagram

quential bytes within the sector buffer. The decoder also generates a $\overline{CS}$ for the 82064 whenever A_{0-2} does not equal '000', allowing access to the 82064's internal Task Register File while keeping the sector buffer tri-stated.

During a WRITE SECTOR Command, the host processor sets up data in the Task Register File and then issues the command. The 82064 asserts BUFFER COUNTER RESET ($\overline{BCR}$) to reset the counter. It then generates a status to inform the host that it can load the sector buffer with data to be written. When the counter reaches its maximum count, the BUFFER READY (BRDY) signal is asserted by the carry out of the counter, informing the 82064 that the sector buffer is full. (BRDY is a rising edge triggered signal which will be ignored if asserted before the 82064 asserts $\overline{BCR}$.) $\overline{BCS}$ is then asserted, disconnecting the host through the transceivers, and the $\overline{RD}$ and $\overline{WR}$ lines become outputs from the 82064 to allow access to the sector buffer. When the 82064 is done using the buffer, it deasserts $\overline{BCS}$ which again allows the host to access the local bus. The READ SECTOR command operates in a similar manner, except the buffer is loaded by the 82064 instead of the host.

Another control signal, BUFFER DATA REQUEST (BDRQ), can be used with a DMA controller to indicate that the 82064 is ready to send or receive data. When data transfer is via a programmed I/O environment, it is the responsibility of the host to interrogate the DRQ status bit to determine if the 82064 is ready (bit 3 of the status register). For further explanation, refer to the individual command descriptions and the A.C. Characteristics.

When INTRQ is asserted, the host is signaled that execution of a command has terminated (either a normal termination or an aborted command). For the READ SECTOR command, interrupts may be programmed to be asserted either at the termination of the command, or when BDRQ is asserted. INTRQ will remain active until the host reads the STATUS register to determine the cause of the termination, or writes a new command into the COMMAND register.

The 82064 asserts $\overline{\text{SDHLE}}$ whenever the SDH register is being written. This signal can be used to latch the drive and head select information in an external register for decoding. Figure 5 illustrates one method.

Drive Interface

The drive side of the 82064 WDC requires three sections of external logic. These are the control line buffer/receivers, data separator, and write precompensation. Figure 5 illustrates a drive interface.

The buffer/receivers condition the control lines to be driven down the cable to the drive. The control lines are typically single-ended, resistor terminated, TTL levels. The data lines to and from the drive also require buffering. This is typically done with differential RS-422 drivers. The interface specification for the drive will be found in the drive manufacturer's OEM

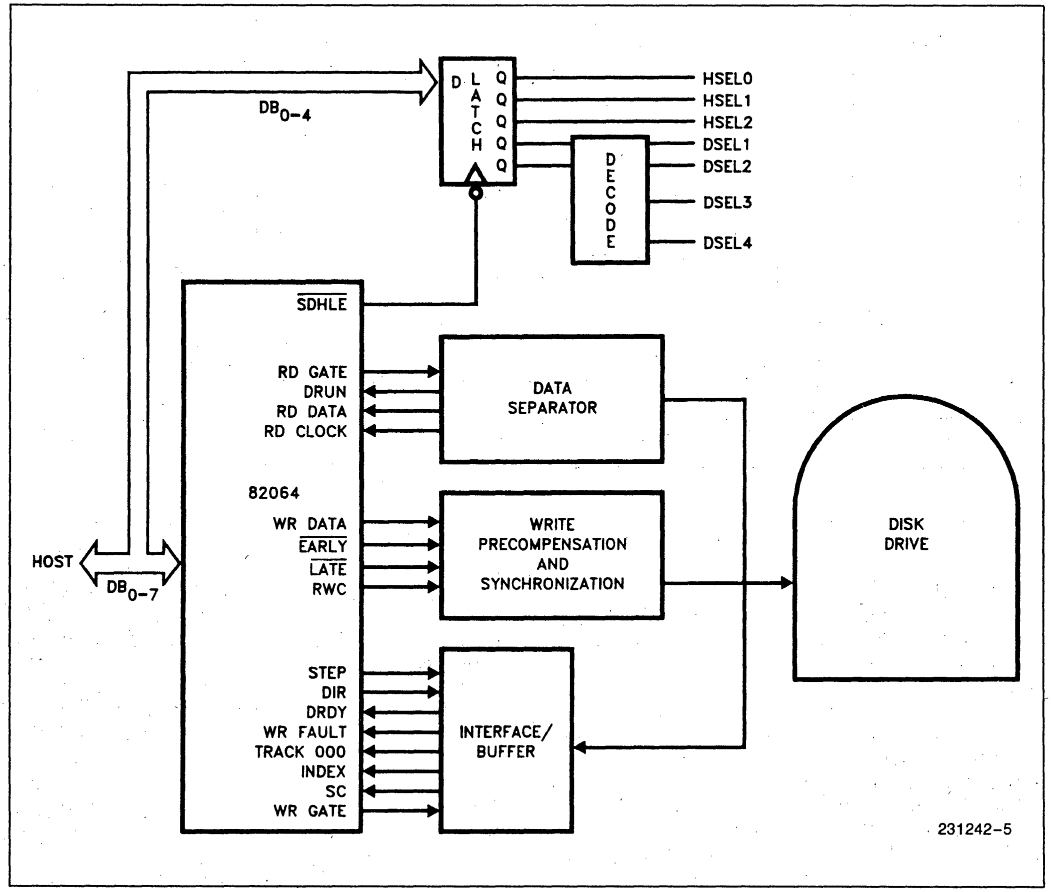


Figure 5. Drive Interface Block Diagram

manual. The 82064 supplies TTL compatible signals, and will interface to most buffer/driver devices.

The data recovery circuits consist of a phase locked loop, data separator, and associated components. The 82064 interacts with the data separator through the DATA RUN (DRUN) and RD GATE signals. A block diagram of a typical data separator circuit is shown in Figure 6. Read data from the drive is presented to the RD DATA input of the 82064, the reference multiplexor, and a retriggerable one shot. The RD GATE output will be deasserted when the 82064 is not inspecting data. The PLL should remain locked to the reference clock.

When any READ or WRITE command is initiated and a search for an address mark begins, the DRUN input is examined. The DRUN one-shot is set for slightly longer than one bit time, allowing it to retrigger constantly on a field of all ones or all zeroes. An internal counter times out to see that DRUN is asserted for two byte times. RD GATE is asserted by the 82064, switching the data separator to lock on to the incoming data stream. If DRUN is deasserted prior to an additional seven byte times, RD GATE is deasserted and the process is repeated. RD GATE will remain asserted until a non-zero, non-address mark byte is detected. The 82064 will then deassert RD GATE for two byte times to allow the PLL to lock back on the reference clock, and start the DRUN search again. If an address mark is detected, RD GATE remains asserted and the command will continue searching for the proper ID field. This sequence is shown in the flow chart in Figure 7.

The write precompensation circuitry is designed to reduce the drift in the data caused by interaction between bits. It is divided into two parts, REDUCED WRITE CURRENT (RWC) and EARLY/LATE writing of bits. A block diagram of a typical write precompensation circuit is shown in Figure 8.

The cylinder in which the RWC line becomes active is controlled by the REDUCE WRITE CURRENT register in the Task Register File. When a cylinder is written which has a cylinder number greater than or equal to the contents of this register, the write current will be reduced. This will decrease the interaction between the bits.

Drift may also be caused by the bit pattern. With certain combinations of ones and zeroes some of the bits can drift far enough apart to be difficult to read without error. This phenomenon can be minimized by using EARLY and LATE as described below. The 82064 examines three bits, the last one written, the one being written, and the next one to be written. From this, it determines whether to assert EARLY or LATE. Since the bit leaving the 82064 has already been written, it is too late to make it early. Therefore, the external delay circuit must be as follows:

EARLY asserted and LATE deasserted = no delay

EARLY deasserted and LATE deasserted = one unit delay (typically 12–15 ns)

EARLY deasserted and LATE asserted = two units delay (typically 24–30 ns)

EARLY and LATE are always active, and should be gated externally by the RWC signal. Figure 8 illustrates one method of using these signals.

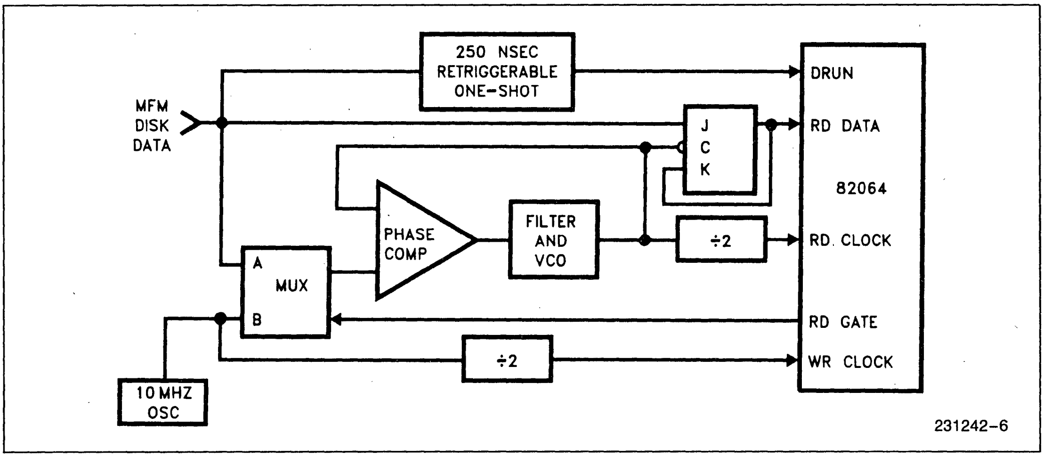


Figure 6. Data Separator Circuit

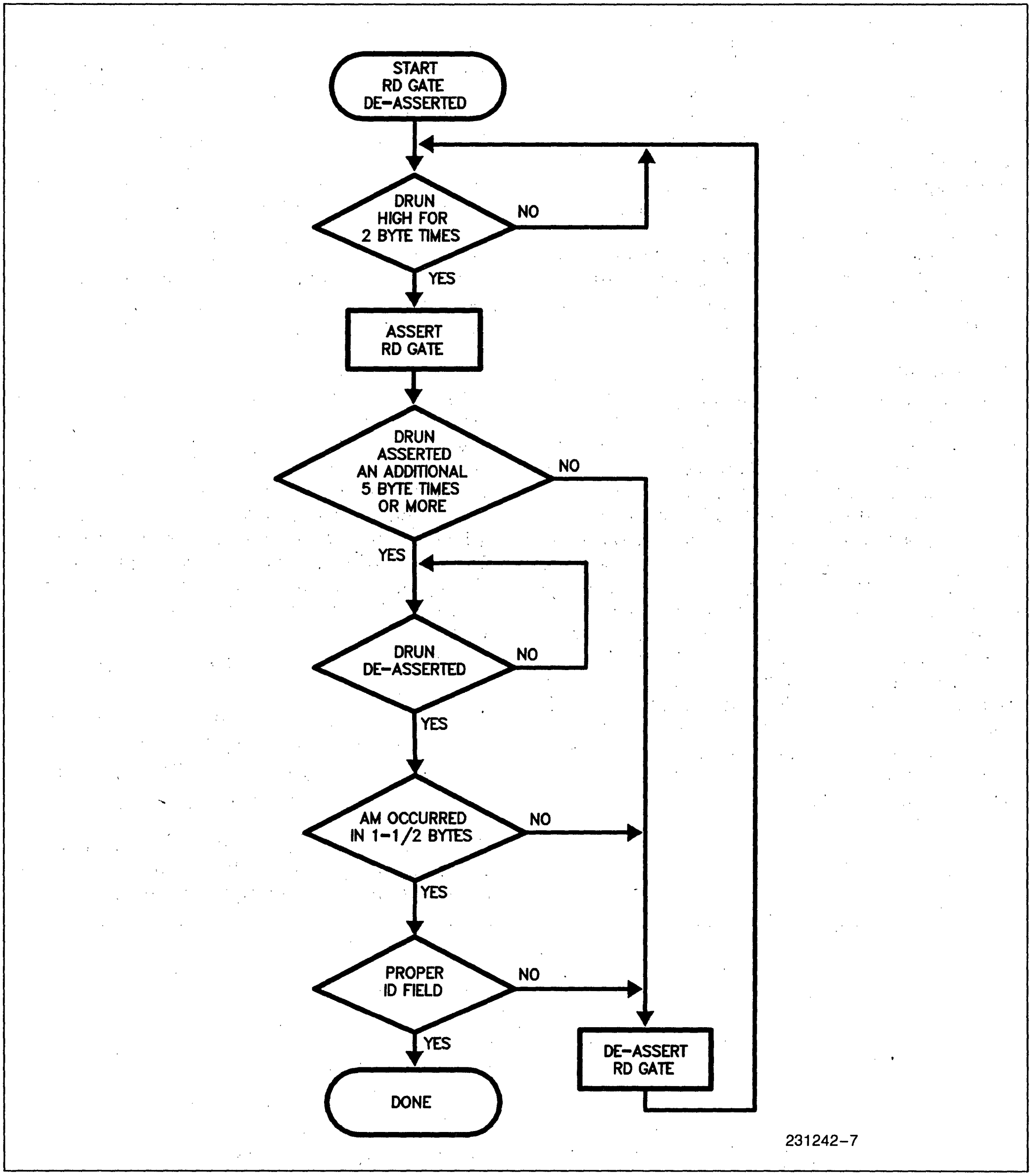


Figure 7. PLL Control Sequence

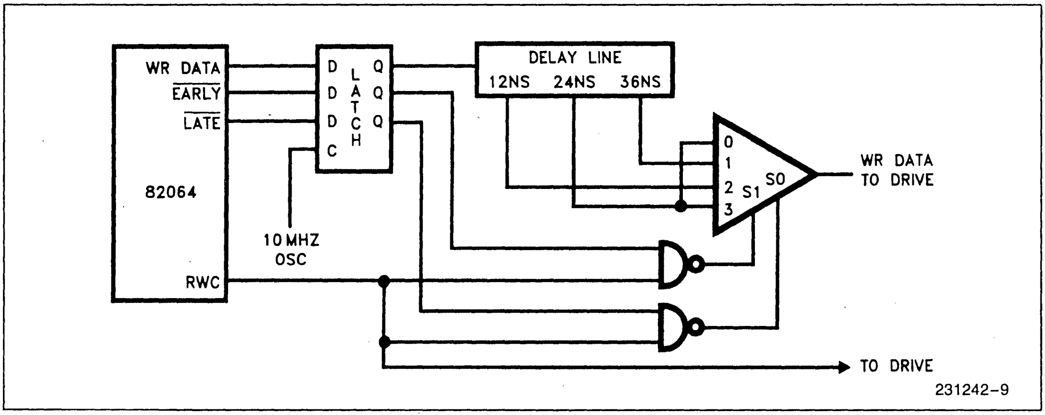


Figure 8. Write Precompensation Circuit

TASK REGISTER FILE

The Task Register File is a bank of nine registers used to hold parameter information pertaining to each command, status information, and the command itself. These registers and their addresses are:

A2	A1	A0	READ	WRITE
0	0	0	BUS TRI-STATED	BUS TRI-STATED
0	0	1	ERROR REGISTER	REDUCE WRITE CURRENT
0	1	0	SECTOR COUNT	SECTOR COUNT
0	1	1	SECTOR NUMBER	SECTOR NUMBER
1	0	0	CYLINDER LOW	CYLINDER LOW
1	0	1	CYLINDER HIGH	CYLINDER HIGH
1	1	0	SDH	SDH
1	1	1	STATUS	COMMAND

NOTE:
These registers are not cleared by $\overline{\text{RESET}}$ being asserted.

ERROR REGISTER

This read only register contains specific error information after the termination of a command. The bits are defined as follows:

7	6	5	4	3	2	1	0
BB	CRC/ECC	0	ID	0	AC	TK000	DAM

Bit 7 - Bad Block Detect (BB)

This bit is set when an ID field has been encountered that contains a bad block mark. It is used for bad sector mapping.

Bit 6 - CRC/ECC Data Field Error (CRC/ECC)

When in the CRC mode (SDH register, bit 7 = 0), this bit is set when a CRC error occurs in the data field. When retries are enabled, ten more attempts are made to read the sector correctly. If none of these attempts are successful bit 0 in the STATUS register is also set. If one of the attempts is successful, the CRC/ECC error bit remains set to inform the host that a marginal condition exists; however, bit 0 in the STATUS register is not set.

When in the ECC mode (SDH register, bit 7 = 1), this bit is set when the first non-zero syndrome is detected. When retries are enabled, up to ten attempts are made to correct the error. If the error is successfully corrected, this bit remains set; however, bit 2 of the STATUS register is also set to inform the host that the error has been corrected. If the error is not correctable, the CRC/ECC error bit remains set and bit 0 of the STATUS register is also set.

The data may be read even if uncorrectable errors exist.

NOTE: If the long mode (L) bit is set in the READ or WRITE command, no error checking is performed.

Bit 5 - Reserved

Not used. Forced to zero.

Bit 4 - ID Not Found (ID)

This bit is set to indicate that the correct cylinder, head, sector, or size parameter could not be found, or that a CRC error occurred in the ID field. This bit is set on the first failure and remains set even if the error is recovered on a retry. When recovery is unsuccessful, the Error bit (bit 0) of the STATUS register is also set.

For a SCAN ID command with retries enabled (T = 0), the Error bit in the STATUS register is set after ten unsuccessful attempts have been made to find the correct ID. With retries disabled (T = 1), only two attempts are made before setting the Error bit.

For a READ or WRITE command with retries enabled (T = 0), ten attempts are made to find the correct ID field. If there is still an error on the tenth try, an auto-scan and auto-seek are performed. Then ten more retries are made before setting the Error bit. When retries are disabled (T = 1), only two tries are made. No auto-scan or auto-seek operations are performed.

Bit 3 - Reserved

Not used. Forced to zero.

Bit 2 - Aborted Command (AC)

Command execution is aborted and this bit is set if a command was issued while DRDY is deasserted or WR FAULT is asserted. This bit will also be set if an undefined command is written to the COMMAND register; however, an implied seek will be executed.

Bit 1 - Track 000 Error (TK000)

This bit is set during the execution of a RESTORE command if the TRACK 000 pin has not gone active after the issuance of 2047 step pulses.

Bit 0 - Data Address Mark (DAM) Not Found

This bit is set during the execution of a READ SECTOR command if the DAM is not found following the proper sector ID.

REDUCE WRITE CURRENT REGISTER

This register is used to define the cylinder number where the RWC output (Pin 33) is asserted.

7	6	5	4	3	2	1	0
CYLINDER NUMBER ÷ 4							

The value (00-FFH) loaded into this cylinder is internally multiplied by four to specify the actual cylinder where RWC is asserted. Thus a value of 01H will cause RWC to be asserted on cylinder 04H, 02H on cylinder 08H, . . . , 9CH on cylinder 270H, 9DH on cylinder 274H, and so on. RWC will be asserted when the present cylinder is greater than or equal to four times the value of this register. For example, the ST506 interface requires precomp on cylinder 80H and above. Therefore, the REDUCE WRITE CURRENT register should be loaded with 20H.

A value of FFH causes RWC to remain deasserted, regardless of the actual cylinder number.

SECTOR COUNT REGISTER

This register is used to define the number of sectors that need to be transferred to the buffer during a READ MULTIPLE SECTOR or WRITE MULTIPLE SECTOR command.

7	6	5	4	3	2	1	0
NUMBER OF SECTORS							

The value contained in the register is decremented after each sector is transferred to/from the sector buffer. A zero represents a 256 sector transfer, a one a one sector transfer, etc. This register is a "don't care" when single sector commands are specified.

SECTOR NUMBER REGISTER

This register holds the sector number of the desired sector.

7	6	5	4	3	2	1	0
SECTOR NUMBER							

For a multiple sector command, it specifies the first sector to be transferred. It is incremented after each sector is transferred to/from the sector buffer. The SECTOR NUMBER register may contain any value from 0 to 255.

The SECTOR NUMBER register is also used to program the Gap 1 and Gap 3 lengths to be used when formating a disk. See the WRITE FORMAT command description for further explanation.

CYLINDER NUMBER LOW REGISTER

This register holds the lower byte of the desired cylinder number.

7	6	5	4	3	2	1	0
LS BYTE OF CYL. NUMBER							

It is used with the CYLINDER NUMBER HIGH register to specify the desired cylinder number over a range of 0 to 2047.

CYLINDER NUMBER HIGH REGISTER

This register holds the three most significant bits of the desired cylinder number.

7	6	5	4	3	2	1	0
x	x	x	x	x	#	#	#

The CYLINDER NUMBER LOW/HIGH register pair determine where the R/W heads are to be positioned. The host writes the desired cylinder number into these registers. Internal to the 82064 is another pair of registers that hold the present head location. When any command other than a RESTORE is executed, the internal head location registers are compared to the CYLINDER NUMBER registers to determine how many cylinders to move the heads and in what direction.

The internal head location registers are updated to equal the CYLINDER NUMBER registers after the completion of the seek.

When a RESTORE command is executed, the internal head location registers are reset to zero while DIR and STEP move the heads to track zero.

SECTOR/DRIVE/HEAD (SDH) REGISTER

The SDH register contains the desired sector size, drive number, and head parameters. The format is shown in Figure 9. The EXT bit (bit 7) is used to select between the CRC or ECC mode. When bit 7 = 1 the ECC mode is selected for the data field. When bit 7 = 0 the CRC mode is selected.

The SDH byte written in the ID field of the disk by the FORMAT command is different than the SDH register contents. The recorded SDH byte does not have the drive number recorded, but does have the bad block mark written. The format of the SDH byte written on the disk is:

7	6	5	4	3	2	1	0
BAD B.	SIZE		0	0	HEAD		

STATUS REGISTER

The status register is used to inform the host of certain events performed by the 82064, as well as reporting status from the drive control lines. Reading the STATUS register deasserts INTRQ. The format is:

7	6	5	4	3	2	1	0
BUSY	READY	WF	SC	DRQ	DWC	CIP	ERR

Bit 7 - Busy

This bit is asserted when a command is written into the COMMAND register and, except for the READ command, is deasserted at the end of the command. When executing a READ command, Busy will be deasserted when the sector buffer is full. Commands should not be loaded into the COMMAND register when Busy is set. When the Busy bit is set, no other bits in the STATUS or ERROR registers are valid.

Bit 6 - Ready

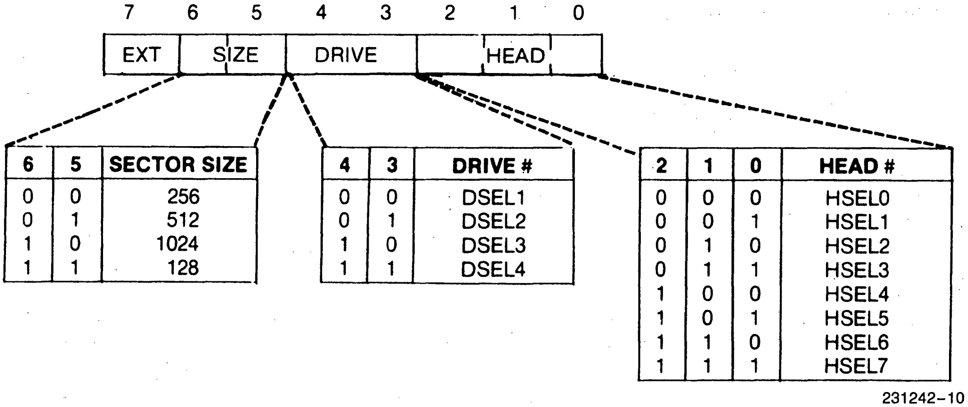
This bit reflects the status of DRDY (pin 28). When this bit equals zero, the command is aborted and the status of this bit is latched.

Bit 5 - Write Fault (WF)

This bit reflects the status of WR FAULT (pin 30). When this bit equals one the command is aborted, INTRQ is asserted, and the status of this bit is latched.

Bit 4 - Seek Complete (SC)

This bit reflects the status of SC (pin 32). When a seek or implied seek has been initiated by a command, execution of the command pauses until the seek is complete. This bit is latched after an aborted command error.



6	5	SECTOR SIZE
0	0	256
0	1	512
1	0	1024
1	1	128

4	3	DRIVE #
0	0	DSEL1
0	1	DSEL2
1	0	DSEL3
1	1	DSEL4

2	1	0	HEAD #
0	0	0	HSEL0
0	0	1	HSEL1
0	1	0	HSEL2
0	1	1	HSEL3
1	0	0	HSEL4
1	0	1	HSEL5
1	1	0	HSEL6
1	1	1	HSEL7

NOTE:
Drive select and head select lines must be generated externally. Figure 3 represents one method of achieving this.

Figure 9. SDH Register Format

Bit 3 - Data Request (DRQ)

The DRQ bit reflects the status of BDRQ (pin 36). It is asserted when the sector buffer must be written into or read from. DRQ and BDRQ remain asserted until BRDY indicates that the sector buffer has been filled or emptied, depending upon the command. BDRQ can be used for DMA interfacing, while DRQ is used in a programmed I/O environment.

Bit 2 - Data Was Corrected (DWC)

When set, this bit indicates that an ECC error has been detected during a read operation, and that the data in the sector buffer has been corrected. This provides the user with an indication that there may be a marginal condition within the drive before the errors become uncorrectable. This bit is forced to zero when not in the ECC mode.

Bit 1 - Command In Progress (CIP)

When this bit is set a command is being executed and a new command should not be loaded. Although a command is being executed, the sector buffer is still available for access by the host. When the 82064 is no longer Busy (bit 7 = 0) the STATUS register can be read. If other registers are read while CIP is set the contents of the STATUS register will be returned.

Bit 0 - Error

This bit is set whenever any bits in the ERROR register are set. It is the logical 'or' of the bits in the ERROR register and may be used by the host processor to quickly check for nonrecoverable errors. The host must read the ERROR register to determine what type of error occurred. This bit is reset when a new command is written into the COMMAND register.

COMMAND REGISTER

The command to be executed is written into this write-only register:

7	6	5	4	3	2	1	0
COMMAND							

The command sets Busy and CIP, and begins to execute as soon as it is written into this register. Therefore, all necessary information should be loaded into the Task Register File prior to entering the command. Any attempt to write a register will be ignored until command execution has terminated, as indicated by the CIP bit being cleared. INTRQ is deasserted when the COMMAND register is written.

COMMAND	7	6	5	4	3	2	1	0
RESTORE	0	0	0	1	R3	R2	R1	R0
SEEK	0	1	1	1	R3	R2	R1	R0
READ SECTOR	0	0	1	0	I	M	L	T
WRITE SECTOR	0	0	1	1	0	M	L	T
SCAN ID	0	1	0	0	0	0	0	T
WRITE FORMAT	0	1	0	1	0	G	0	0
COMPUTE CORRECTION	0	0	0	0	1	0	0	0
SET PARAMETER	0	0	0	0	0	0	0	S

R_{3-0} = Stepping Rate Field

For 5 MHz WR CLOCK:

R_{3-0} =	0000	35 µs
	0001	0.5 ms
	0010	1.0 ms
	0011	1.5 ms
	0100	2.0 ms
	0101	2.5 ms
	0110	3.0 ms
	0111	3.5 ms
	1000	4.0 ms
	1001	4.5 ms
	1010	5.0 ms
	1011	5.5 ms
	1100	6.0 ms
	1101	6.5 ms
	1110	3.2 µs
	1111	16 µs

I = Interrupt Control

I = 0 INTRQ occurs with BDRQ/DRQ indicating the sector buffer is full. Valid only when M = 0.

I = 1 INTRQ occurs when the command is completed and the host has read the sector buffer.

M = Multiple Sector Flag

M = 0 Transfer one sector. Ignore the SECTOR COUNT register.

M = 1 Transfer multiple sectors.

L = Long Mode

L = 0 Normal mode. Normal CRC or ECC functions are performed.

L = 1 Long mode. No CRC or ECC bytes are developed or error checking performed on the data field. The 82064 appends the four additional bytes supplied by the host or disk to the data field.

T = Retry Enable

T = 0 Enable retries.

T = 1 Disable retries.

G = Gap Filler Byte.

G = 0. Gaps 1, 3 and pad bytes "4E".

G = 1. Gaps 1, 3 and pad bytes "AA".

S = Error Correction Span

S = 0 5-bit span.

S = 1 11-bit span.

RESTORE COMMAND

The RESTORE command is used to position the R/W heads over track zero. It is usually issued by the host when a drive has just been turned on. The 82064 forces an auto-restore when a FORMAT command has been issued following a drive number change.

The actual step rate used for the RESTORE command is determined by the seek complete time. A step pulse is issued and the 82064 waits for a rising edge on the SC line before issuing the next pulse. If the rising edge of SC has not occurred within ten revolutions (INDEX pulses) the 82064 switches to sensing the level of SC. If after 2047 step pulses the TRACK 000 line does not go active the 82064 will set the TRACK 000 bit in the ERROR register, assert INTRQ, and terminate execution of the command. An interrupt will also occur if WR FAULT is asserted on DRDY is deasserted at any time during execution.

The rate field specified (R_{3-0}) is stored in an internal register for future use in commands with implied seeks.

A flowchart of the RESTORE command is shown in Figure 10.

SEEK COMMAND

The SEEK command can be used for overlapping seeks on multiple drives. The step rate used is taken from the Rate Field of the command, and is stored in an internal register for future use by those commands with implied seek capability.

The direction and number of step pulses needed are calculated by comparing the contents of the CYLINDER NUMBER registers in the Task Register File to the present cylinder position stored internally. After all the step pulses have been issued the present cylinder position is updated, INTRQ is asserted, and the command terminated.

If DRDY is deasserted or WR FAULT is asserted during the execution of the command, INTRQ is asserted and the command aborts setting the AC bit in the ERROR register.

If an implied seek is performed, the step rate indicated by the rate field is used for all but the last step pulse. On the last pulse, the command execution continues until the rising edge of SC is detected. If 10 INDEX pulses are received without a rising edge of SC, the 82064 will switch to sensing the level of SC.

A flowchart of the SEEK command flow is shown in Figure 11.

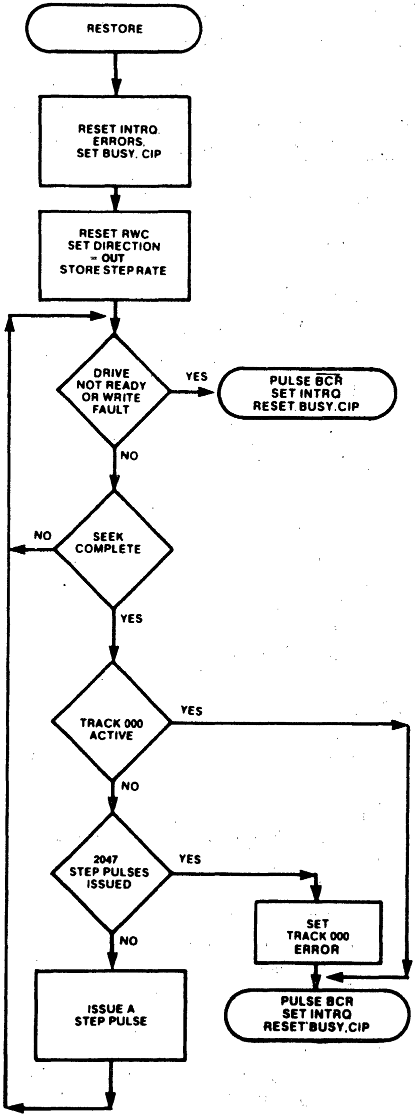
READ SECTOR

The READ SECTOR command is used to transfer one or more sectors of data from the disk to the sector buffer. Upon receipt of the command, the 82064 checks the CYLINDER NUMBER LOW/HIGH register pair against the internal cylinder position register to see if they are equal. If not, the direction and number of steps calculation takes place, and a seek is initiated. As stated in the description of the SEEK command, if an implied seek occurs, the step rate specified by the rate field is used for all but the last step pulse. On the last step pulse the seek continues until the rising edge of SC is detected.

If the 82064 detects a change in the drive number since the last command, an auto-scan ID is performed. This updates the internal cylinder position register to reflect the current drive before the seek begins.

After the 82064 senses SC (with or without an implied seek) it must find an ID field with the correct cylinder number, head, sector size, and CRC. If retries are enabled (T = 0), ten attempts are made to find the correct ID field. If there is still an error on the tenth try, an auto-scan ID and auto-seek are performed. Then ten more retries are attempted before setting the ID Not Found error bit. When retries are disabled (T = 1) only two tries are made. No auto-scan or auto-seek operations are performed.

When the data address mark (DAM) is found, the 82064 is ready to transfer data into the sector buffer. When the disk has filled the sector buffer, the 82064 asserts BDRQ and DRQ and then checks the I flag. If I = 0, INTRQ is asserted, signaling the host to read the contents of the sector buffer. If I = 1, INTRQ occurs after the host has read the sector buffer and the command has terminated. If after successfully reading the ID field, the DAM is not found the DAM Not Found bit in the ERROR register is set.



231242-11

Figure 10. Restore Command Flow

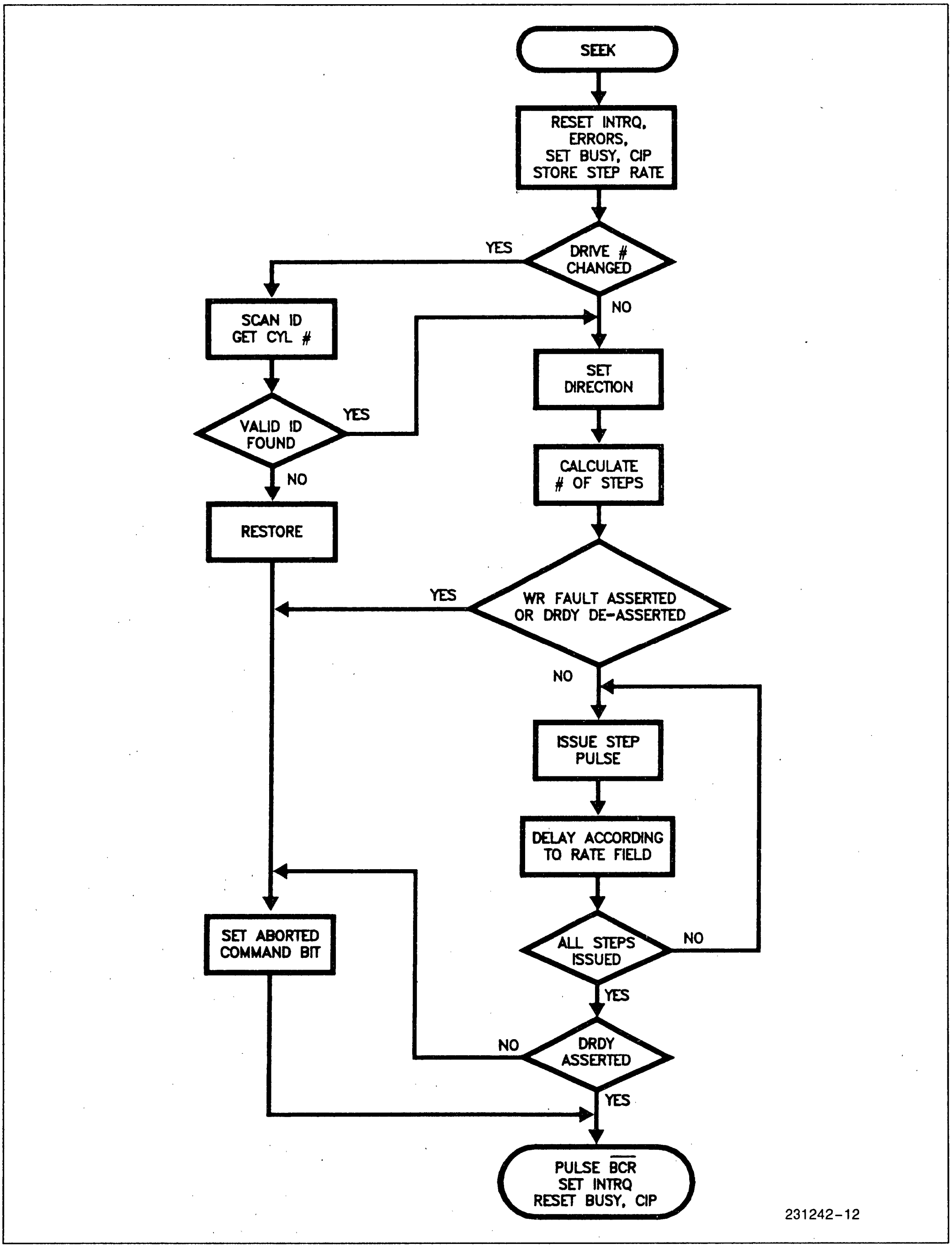


Figure 11. Seek Command Flow

An optional M flag can be set for multiple sector transfers. When M = 0, one sector is transferred and the SECTOR COUNT register is ignored. When M = 1, multiple sectors are transferred. After each sector is transferred, the 82064 decrements the SECTOR COUNT register and increments the SECTOR NUMBER register. The next logical sector is transferred regardless of any interleave. Sectors are numbered during the FORMAT command by a byte in the ID field.

For the 82064 to make multiple sector transfers to the sector buffer, the BRDY signal must be toggled from low to high for each sector. The transfers continue until the SECTOR COUNT register equal zero. If the SECTOR COUNT is not zero (indicating more sectors remain to be read), and the sector buffer is full, BDRQ will be asserted and the host must unload the sector buffer. Once this occurs, the sector buffer is free to accept the next sector.

WR FAULT and DRDY are monitored throughout the command execution. If WR FAULT is asserted or DRDY is deasserted, the command will terminate and the Aborted Command bit in the ERROR register will be set. For a description of the error checking procedure on the data field see the explanation in the section entitled "CRC and ECC Generator and Checker."

Both the READ and WRITE commands feature a "simulated completion" to ease programming. BDRQ, DRQ, and INTRQ are generated in a normal manner upon detection of an error condition. This allows the same program flow for successful or unsuccessful completion of a command.

In summary then, the READ SECTOR operation is as follows:

When M = 0 (Single Sector Read)

1. HOST: Sets up parameters. Issues READ SECTOR command.
2. 82064: Asserts $\overline{\text{BCR}}$.
3. 82064: Finds sector specified. Asserts $\overline{\text{BCR}}$ and $\overline{\text{BCS}}$. Transfers data to sector buffer.
4. 82064: Asserts $\overline{\text{BCR}}$. Deasserts $\overline{\text{BCS}}$.
5. 82064: Asserts BDRQ and DRQ.
6. 82064: If I = 1 then go to 9.
7. HOST: Read contents of sector buffer.
8. 82064: Wait for BRDY, then assert INTRQ. End.
9. 82064: Assert INTRQ.
10. HOST: Read contents of sector buffer. End.

When M = 1 (Multiple Sector Read)

1. HOST: Sets up parameters. Issues READ SECTOR command.
2. 82064: Asserts $\overline{\text{BCR}}$.
3. 82064: Finds sector specified. Asserts $\overline{\text{BCR}}$ and $\overline{\text{BCS}}$. Transfers data to sector buffer.
4. 82064: Asserts $\overline{\text{BCR}}$. Deasserts $\overline{\text{BCS}}$.
5. 82064: Asserts BDRQ and DRQ.
6. HOST: Reads contents of sector buffer.
7. SECTOR BUFFER: Indicates data has been transferred by asserting BRDY.
8. 82064: When BRDY is asserted, decrement SECTOR COUNT, increment SECTOR NUMBER. If SECTOR COUNT = 0, go to 10.
9. 82064: Go to 2.
10. 82064: Assert INTRQ.

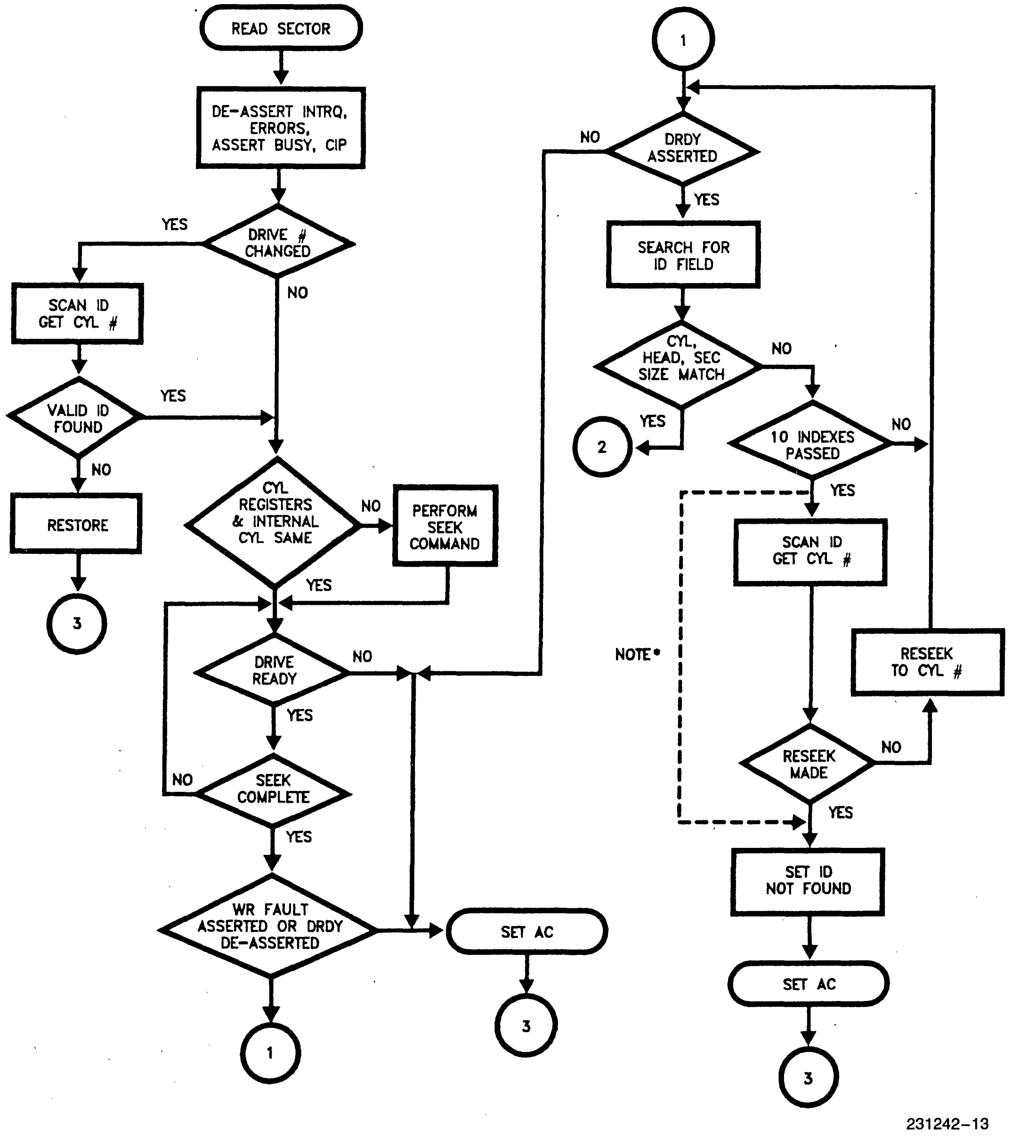
A flowchart of the READ SECTOR command is shown in Figure 12.

WRITE SECTOR

The WRITE SECTOR command is used to write one or more sectors of data from the sector buffer to the disk. Upon receipt of the command, the 82064 checks the CYLINDER NUMBER LOW/HIGH register pair against the internal cylinder position register to see if they are equal. If not, the direction and number of steps calculation takes place, and a seek is initiated. As stated in the description of the SEEK command, if an implied seek occurs, the step rate specified by the rate field is used for all but the last step pulse. On the last step pulse the seek continues until the rising edge of SC is detected.

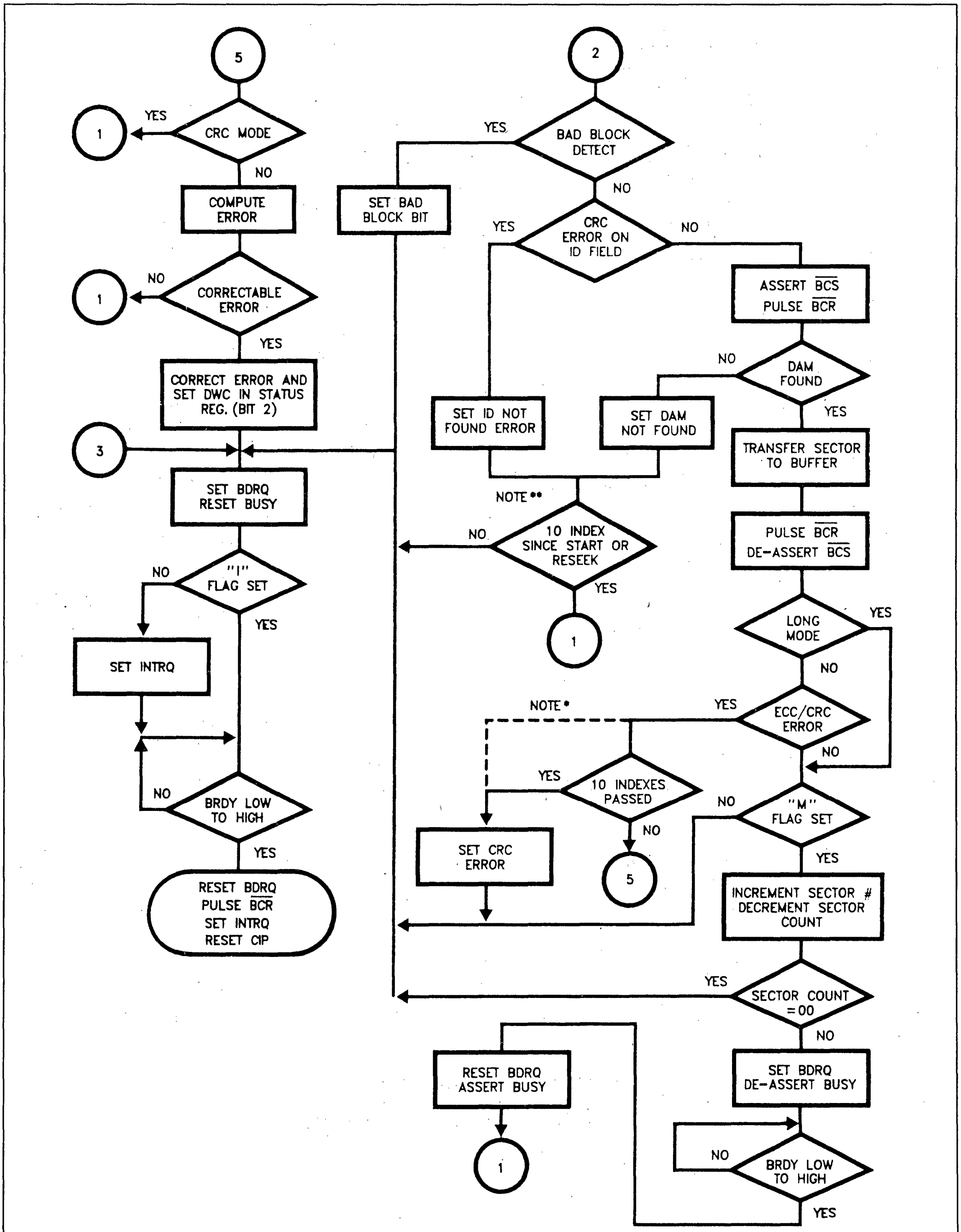
If the 82064 detects a change in the drive number since the last command, an auto-scan ID is performed. This updates the internal cylinder position register to reflect the current drive before the seek begins.

After the 82064 senses SC (with or without an implied seek) BDRQ and DRQ are asserted and the host begins filling the sector buffer with data. When BRDY is asserted, a search for the ID field with the correct cylinder number, head, sector size, and CRC is initiated. If retries are enabled (T = 0), ten attempts are made to find the correct ID field. If there is still an error on the tenth try, an auto-scan ID and auto-seek are performed. Then ten more retries are attempted before setting the ID Not Found error bit. When retries are disabled (T = 1) only two tries are made. No auto-scan or auto-seek operations are performed.



*If T bit of command = 1 then dashed path is taken after 2 index pulses.

Figure 12a. Read Sector Command Flow



*If T bit of command = 1 then dashed path is taken.
**If T bit of command = 1 then test is for 2 index pulses.

231242-14

Figure 12b. Read Sector Command Flow (Continued)

When the correct ID is found, WR GATE is asserted and data is written to the disk. When the CRC/ECC bit (SDH Register, bit 7) is zero, the 82064 generates a two byte CRC character to be appended to the data. When the CRC/ECC bit is one, four ECC bytes replace the CRC character. When L = 1, the polynomial generator is inhibited and neither CRC or ECC bytes are generated. Instead four bytes of data supplied by the host are written.

During a WRITE MULTIPLE SECTOR command (M = 1), the SECTOR NUMBER register is incremented and the SECTOR COUNT register is decremented. If BRDY is asserted after the first sector is read from the sector buffer, the 82064 continues to read data from the sector buffer for the next sector. If BRDY is deasserted, the 82064 asserts BDRQ and waits for the host to place more data in the sector buffer.

In summary then, the WRITE SECTOR operation is as follows:

When M = 0,1

1. HOST: Sets up parameters. Issues WRITE SECTOR command.
2. 82064: Asserts BDRQ and DRQ.
3. HOST: Loads sector buffer with data.
4. 82064: Waits for rising edge of BRDY.
5. 82064: Finds specified ID field. Writes sector to disk.
6. 82064: If M = 0, asserts INTRQ. End.
7. 82064: Increments SECTOR NUMBER. Decrements SECTOR COUNT.
8. 82064: IF SECTOR COUNT = 0, assert INTRQ. End.
9. 82064: Go to 2.

A flowchart of the WRITE SECTOR command is shown in Figure 13.

SCAN ID

The SCAN ID command is used to update the SDH, SECTOR NUMBER, and CYLINDER NUMBER LOW/HIGH registers.

After the command is loaded, the SC line is sampled until it is valid. The DRDY and WR FAULT lines are also monitored throughout execution of the command. If a fault occurs the command is aborted and the appropriate error bits are set. When the first ID field is found, the ID information is loaded into the SDH, SECTOR NUMBER, and CYLINDER NUMBER registers. The internal cylinder position register is also updated. If this is an auto-scan caused by a change in drive numbers, only the internal position register is updated. If a bad block is detected, the BAD BLOCK bit will also be set.

If an ID field is not found, or if a CRC error occurs, and if retries are enabled (T = 0), ten attempts are made to read it. If retries are disabled (T = 1), only two tries are made. There is no auto-seek in this command and the sector buffer is not disturbed.

A flowchart of the SCAN ID command is shown in Figure 14.

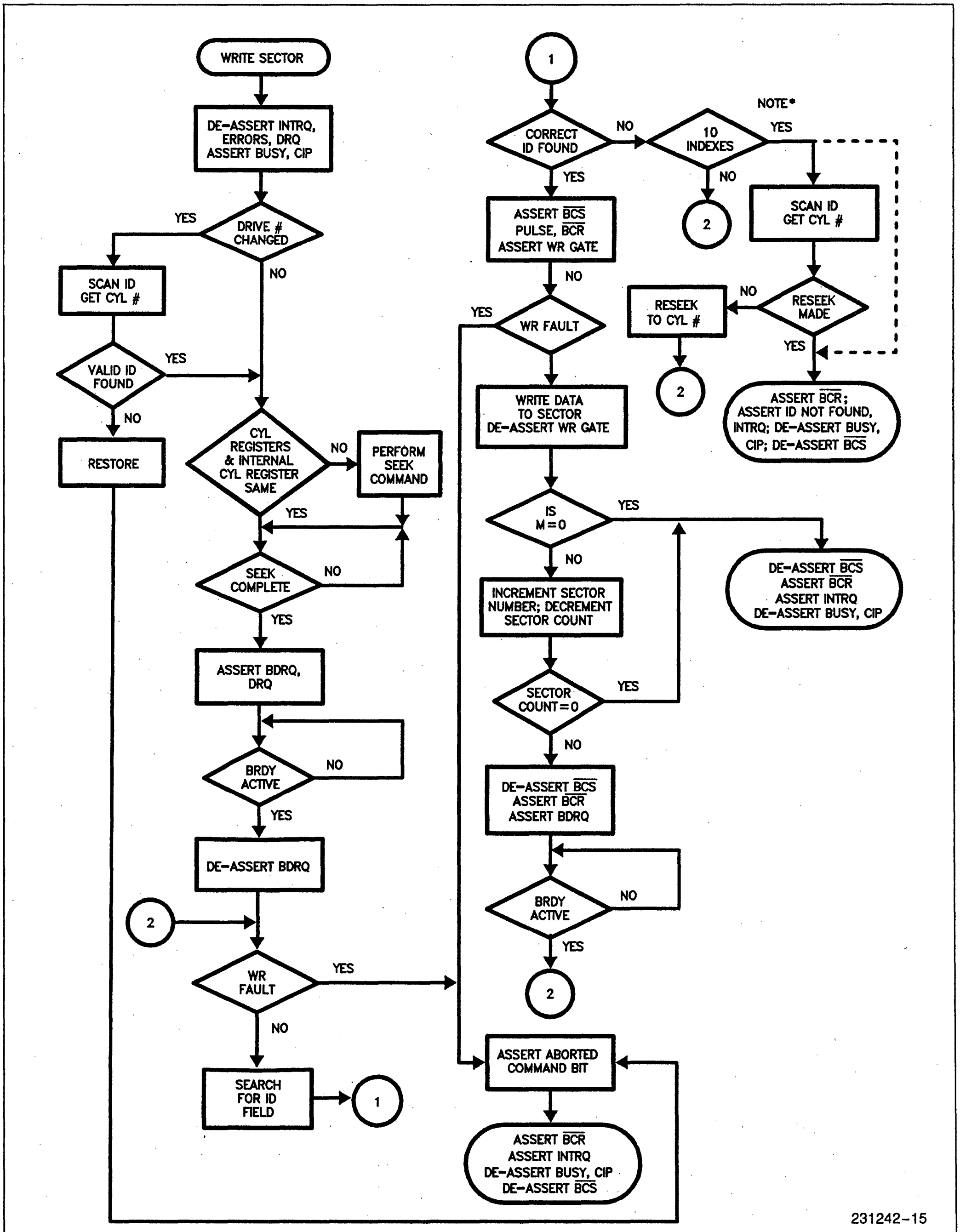
WRITE FORMAT

The WRITE FORMAT command is used to format one track using information in the Task Register File and the sector buffer. During execution of this command, the sector buffer is used for additional parameter information instead of data. Shown in Figure 15 is the contents of a sector buffer for a 32 sector track with an interleave factor of two.

Each sector requires a two byte sequence. The first byte designates whether a bad block mark is to be recorded in the sector's ID field. An 00H is normal; an 80H indicates a bad block mark for that sector. In the example of Figure 15, sector 04 will get a bad block mark recorded. The second byte indicates the logical sector number to be recorded. This allows sectors to be recorded with any interleave factor desired. The remaining memory in the sector buffer may be filled with any value; its only purpose is to generate a BRDY to tell the 82064 to begin formatting the track.

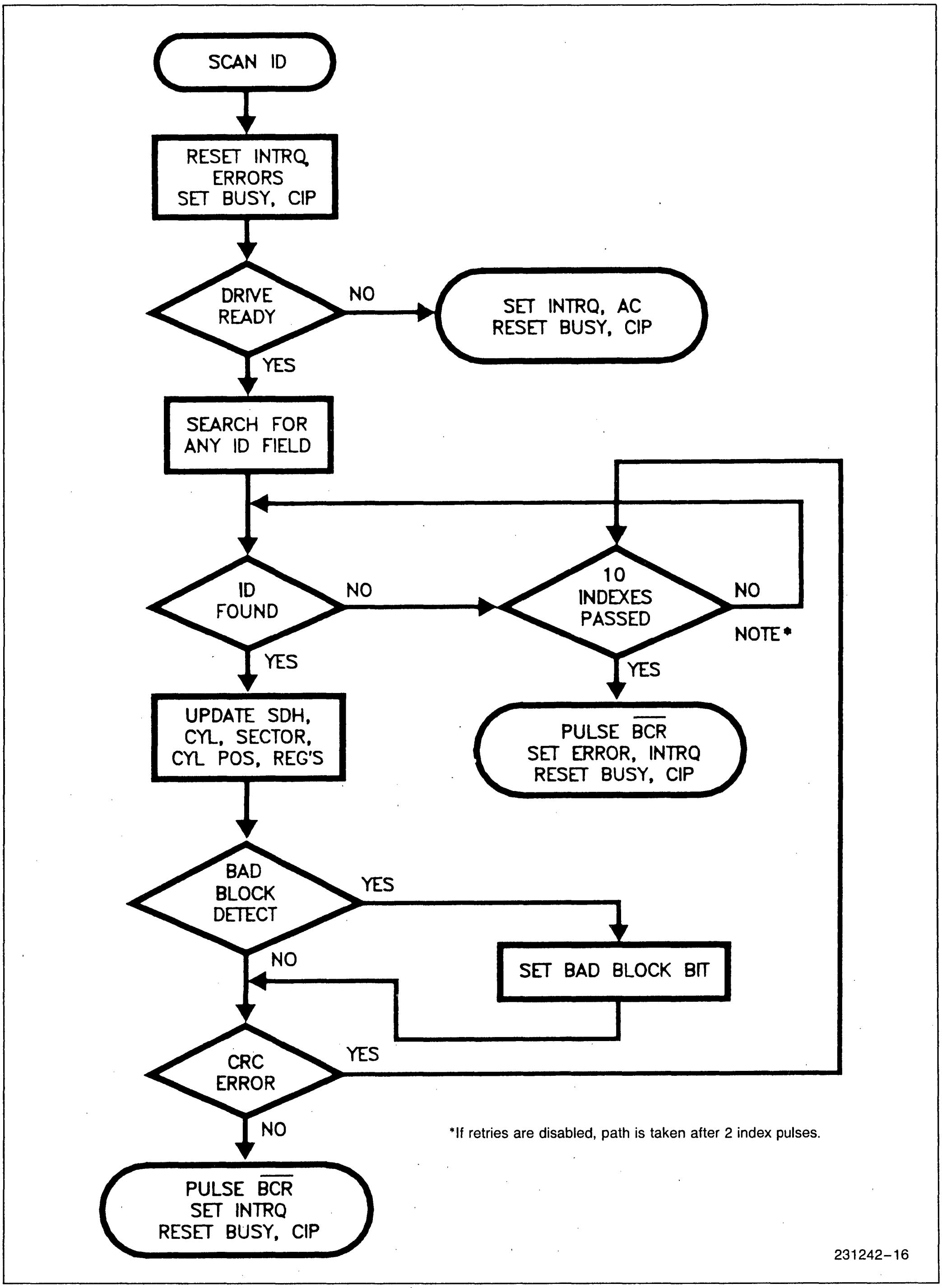
If the drive number has been changed since the last command, an auto-restore is initiated, positioning the heads to track 000. The internal cylinder position register is set to zero and the heads seek to the track specified in the Task Register File CYLINDER NUMBER register. This prevents an ID Not Found error from occuring due to an incompatible format, or the track having been erased. A normal implied seek is also in effect for this command.

The SECTOR COUNT register is used to hold the total number of sectors to be formatted (FFH = 255 sectors), while the SECTOR NUMBER register holds the number of bytes, minus three, to be used for Gap 1 and Gap 3. If, for example, the SECTOR COUNT register value is 02H and the SECTOR NUMBER register value is 00H, then 2 sectors are formatted and 3 bytes of 4EH are written for Gap 1 and Gap 3. The data fields are filled with FFH and the CRC or ECC is automatically generated and appended. After the last sector is written the track is filled with 4EH.


WRITE SECTOR
DE-ASSERT INTRQ, ERRORS, DRQ ASSERT BUSY, CIP
DRIVE # CHANGED
YES
NO
SCAN ID GET CYL #
VALID ID FOUND
YES
NO
RESTORE
CYL REGISTERS & INTERNAL CYL REGISTER SAME
NO
PERFORM SEEK COMMAND
YES
SEEK COMPLETE
NO
YES
ASSERT BDRQ, DRQ
BRDY ACTIVE
NO
YES
DE-ASSERT BDRQ
2
WR FAULT
YES
NO
SEARCH FOR ID FIELD
1
1
CORRECT ID FOUND
NO
YES
10 INDEXES
NOTE*
YES
NO
2
SCAN ID GET CYL #
RESEEK MADE
NO
YES
RESEEK TO CYL #
2
ASSERT $\overline{BCR}$; ASSERT ID NOT FOUND, INTRQ; DE-ASSERT BUSY, CIP; DE-ASSERT $\overline{BCS}$
ASSERT $\overline{BCS}$ PULSE, $\overline{BCR}$ ASSERT WR GATE
NO
WR FAULT
YES
WRITE DATA TO SECTOR DE-ASSERT WR GATE
IS M=0
YES
NO
DE-ASSERT $\overline{BCS}$ ASSERT $\overline{BCR}$ ASSERT INTRQ DE-ASSERT BUSY, CIP
INCREMENT SECTOR NUMBER; DECREMENT SECTOR COUNT
SECTOR COUNT=0
YES
NO
DE-ASSERT $\overline{BCS}$ ASSERT $\overline{BCR}$ ASSERT BDRQ
BRDY ACTIVE
NO
YES
2
ASSERT ABORTED COMMAND BIT
ASSERT $\overline{BCR}$ ASSERT INTRQ DE-ASSERT BUSY, CIP DE-ASSERT $\overline{BCS}$
231242-15

*If retries disabled then dashed path is taken after 2 index pulses.

Figure 13. Write Sector Command Flow



*If retries are disabled, path is taken after 2 index pulses.

Figure 14. Scan ID Command Flow

ADDR	DATA 0	1	2	3	4	5	6	7
00	00	00	00	10	00	01	00	11
08	00	02	00	12	00	03	00	13
10	80	04	00	14	00	05	00	15
18	00	06	00	16	00	07	00	17
20	00	08	00	18	00	09	00	19
28	00	0A	00	1A	00	0B	00	1B
30	00	0C	00	1C	00	0D	00	1D
38	00	0E	00	1E	00	0F	00	1F
40	FF	FF	FF	FF	FF	FF	FF	FF
				: :				
F0	FF	FF	FF	FF	FF	FF	FF	FF

Figure 15. Format Command Buffer Contents

The user may select a value of 4EH or AAH for Gaps 1, 3 and pad bytes. This is done by setting bit 2 (Gap Filler Byte) of the format command to "0" for a value of 4EH or "1" for a value of AAH. AAH provides better frequency discrimination with MFM decoding, allowing for simpler circuitry.

The Gap 3 value is determined by the drive motor speed variation, data sector length, and the interleave factor. The interleave factor is only important when 1:1 interleave is used. The formula for determining the minimum Gap 3 length is:

$$\text{Gap 3} = (2*M*S)+K+E$$

where:

M = motor speed variation (e.g., 0.03 for + 3%)
S = sector length in bytes
K = 18 for an interleave factor of 1
0 for any other interleave factor
E = 2 if ECC is enabled (SDH register, bit 7 = 1)

As for all commands, if WR FAULT is asserted or DRDY is deasserted during execution of the command, the command terminates and the Aborted Command bit in the ERROR register is set.

Figure 16 shows the format which the 82064 will write on the disk.

A flowchart of the WRITE FORMAT command is shown in Figure 17.

COMPUTE CORRECTION

The COMPUTE CORRECTION command determines the location and pattern of a single burst error, but.does not correct it. The host, using the data provided by the 82064, must perform the actual correction. The COMPUTE CORRECTION command is used following a data field ECC error. The command initiating the read must specify no retries (T = 1).

The COMPUTE CORRECTION command first writes the four syndrome bytes from the internal ECC register to the sector buffer. Then the ECC register is clocked. With each clock, a counter is incremented and the pattern examined. If the pattern is correctable, the procedure is stopped and the count and pattern are written to the sector buffer, following the syndrome. The process is also stopped if the count exceeds the sector size before a correctable pattern is found.

When the command terminates the sector buffer contains the following data:

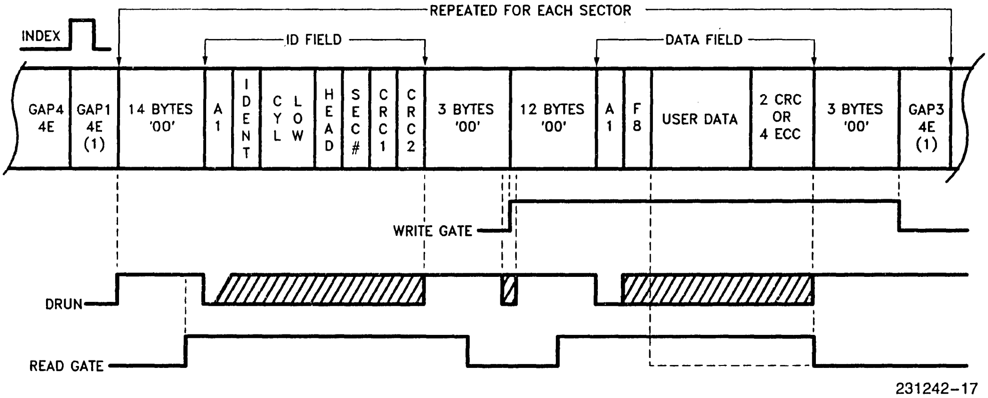
Syndrome MSB
Syndrome
Syndrome
Syndrome LSB
Error Pattern Offset
Error Pattern Offset
Error Pattern MSB
Error Pattern
Error Pattern LSB

As an example, when the Error Pattern Offset is zero the following procedure may correct the error. The first data byte of the sector is exclusive OR'd with the MSB of the Error Pattern, the second data byte with the second byte of the Error Pattern, and the third data byte with the LSB of the Error Pattern.

If the sector buffer count exceeds the sector size, or if the error burst length is greater than that selected by the Set Parameter command, the ECC/CRC error in the ERROR register and the Error bit in the STATUS register is set.

SET PARAMETER

This command selects the correction span to be used for the error correction process. A 5-bit span is selected when bit zero of the command equals 0, and an 11-bit span when bit zero equals 1. The 82064 defaults to a 5-bit span after a RESET.



ID FIELD		
A1	=	A1H with 0AH Clock

IDENT	=	Bits 3, 1, 0 = Cylinder High
		FE = 0-255 Cylinders
		FF = 256-511 Cylinders
		FC = 512-767 Cylinders
		FD = 768-1023 Cylinders
		F6 = 1024-1279 Cylinders
		F7 = 1280-1535 Cylinders
		F4 = 1536-1791 Cylinders
		F5 = 1792-2047 Cylinders

HEAD	=	Bits 0, 1, 2 = Head Number Bits 3, 4 = 0 Bits 5, 6 = Sector Size Bit 7 = Bad Block Mark
Sec #	=	Logical Sector Number
DATA FIELD		
A1	=	A1H with 0AH clock
F8	=	Data Address Mark; Normal Clock
USER	=	Data Field 128 to 1024 Bytes

NOTE:
1. GAP 1 and 3 length determined by Sector Number Register contents during formatting.

Figure 16. Track Format

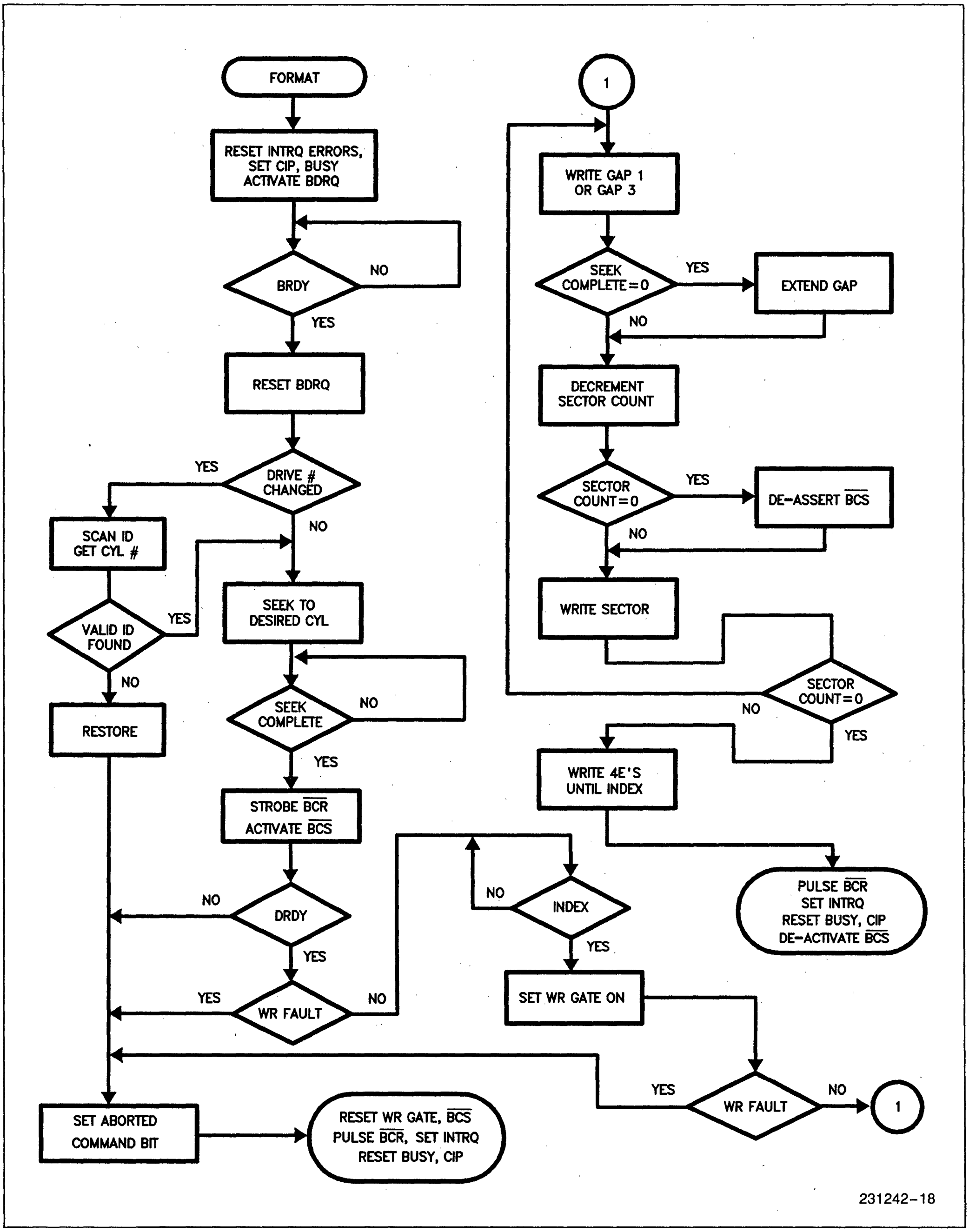


Figure 17. Write Format Command Flow

ELECTRICAL CHARACTERISTICS
ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ...... 0°C to 70°C
Storage Temperature .... −65°C to +150°C
Supply Voltage .... −0.5V to +8V
Voltage on Any Input......... GND − 2V to +6.5V
Voltage on Any Output . GND − 0.5V to V_{CC} + 0.5V
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.*

NOTICE: Specifications contained within the following tables are subject to change.

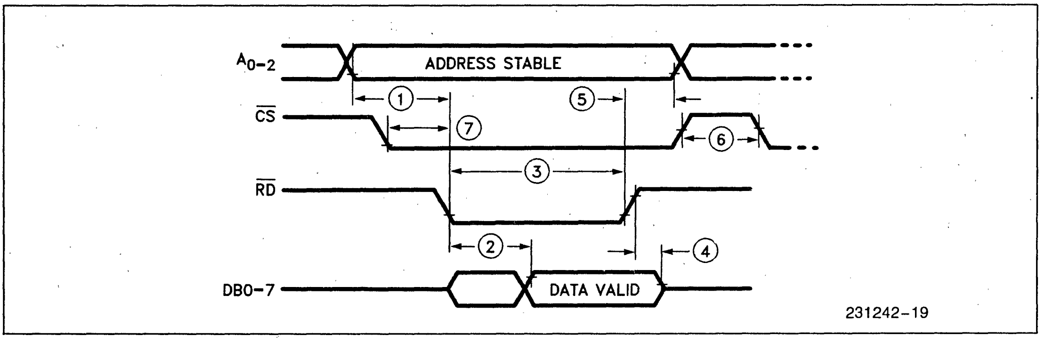
D.C. CHARACTERISTICS T_A = 0°C to 70°C; V_{CC} = +5V ±10%; GND = 0V

Symbol	Parameter	Min	Max	Units	Test Conditions
I_{IL}	Input Leakage Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Leakage Current		±10	μA	V_{OUT} = V_{CC} to 0.45V
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.5	V	
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{OH}	Output High Voltage	V_{CC} − 0.4 3.0		V	I_{OH} = −100 μA I_{OH} = −2.5 mA
V_{OL}	Output Low Voltage		0.4 0.45	V	I_{OL} = 2.5 mA 6.0 mA P21, 22, 23
I_{CC}	Supply Current		20 45	mA	See Note 10 See Note 11
I_{CCSB}	Standby Supply Current		2	mA	See Note 12
C_{IN}	Input Capacitance		10	pF	f_c = 1 MHz
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to GND
For Pins 25, 34, 37, 39 (WR CLOCK, DRUN, READ DATA, READ CLOCK)					
TRS	Rise Time		30	ns	0.9V to 4.2V

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

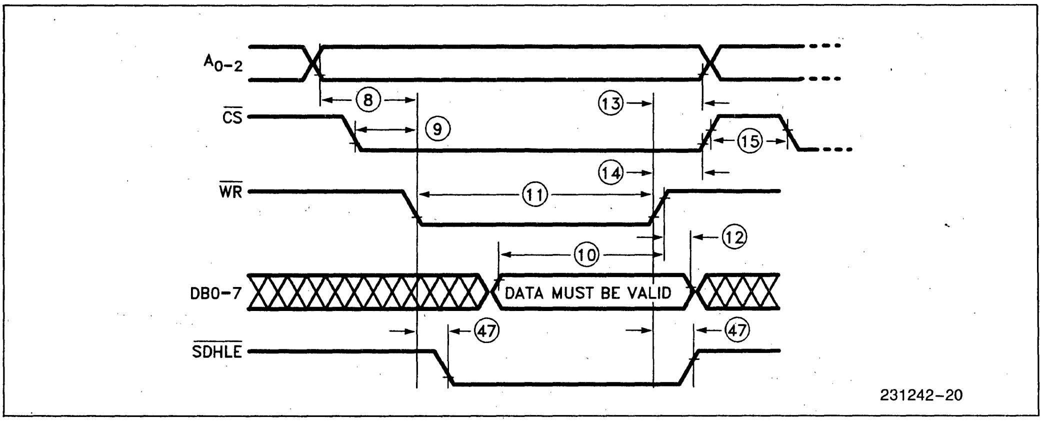
HOST READ TIMING WR CLOCK = 5.0 MHz

Symbol	Parameter	Min	Max	Units	Test Conditions
1	Address Stable Before $\overline{RD}$ ↓	0		ns	
2	Data Delay from $\overline{RD}$ ↓		150	ns	
3	$\overline{RD}$ Pulse Width	100		ns	
4	Data Valid after RD ↑	10	100	ns	
5	Address Hold Time after $\overline{RD}$ ↑	0		ns	
6	Read Recovery Time	300		ns	
7	$\overline{CS}$ Stable before $\overline{RD}$ ↓	0		ns	See Note 6



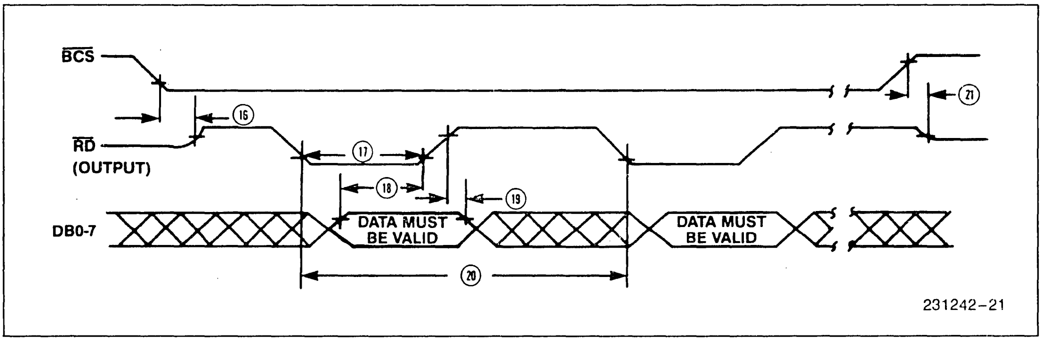
HOST WRITE TIMING WR CLOCK = 5.0 MHz

Symbol	Parameter	Min	Max	Units	Test Conditions
8	Address Stable Before $\overline{WR}$ ↓	0		ns	
9	$\overline{CS}$ Stable Before $\overline{WR}$ ↓	0		ns	
10	Data Setup Time Before $\overline{WR}$ ↑	75		ns	
11	$\overline{WR}$ Pulse Width	100	10000	ns	
12	Data Hold Time After $\overline{WR}$ ↑	0		ns	
13	Address Hold Time After $\overline{WR}$ ↑	0		ns	
14	$\overline{CS}$ Hold Time After $\overline{WR}$ ↑	0		ns	See Note 7
15	Write Recovery Time	300		ns	
47	$\overline{SDHLE}$ Propagation Delay	20	150	ns	



BUFFER READ TIMING (WRITE SECTOR COMMAND) WR CLOCK = 5.0 MHz

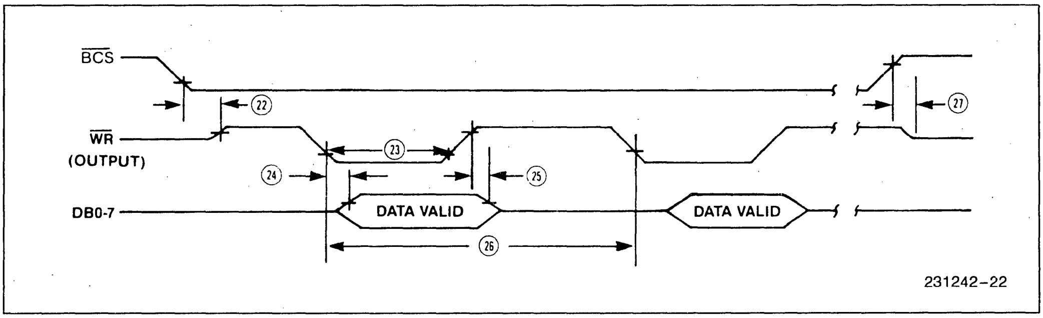
Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
16	$\overline{BCS}$ ↓ to $\overline{RD}$ Valid	0		100	ns	
17	$\overline{RD}$ Output Pulse Width	300	400	500	ns	See Note 3
18	Data Setup to $\overline{RD}$ ↑	140			ns	
19	Data Hold from $\overline{RD}$ ↑	0			ns	
20	$\overline{RD}$ Repetition Rate	1.2	1.6	2.0	μs	See Note 8
21	$\overline{RD}$ Float from $\overline{BCS}$ ↑	0		100	ns	



231242–21

BUFFER WRITE TIMING (READ SECTOR COMMAND) WR CLOCK = 5.0 MHz

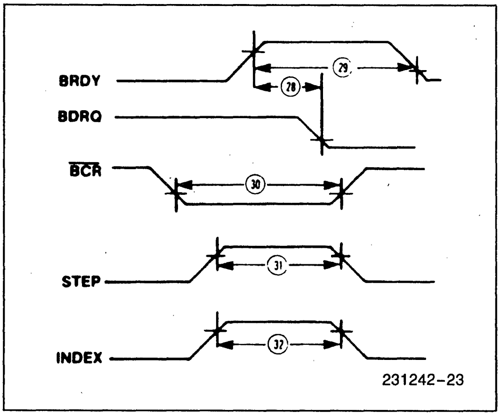
Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
22	$\overline{BCS}$ ↓ to $\overline{WR}$ Valid	0		100	ns	
23	$\overline{WR}$ Output Pulse Width	300	400	500	ns	See Note 3
24	Data Valid from $\overline{WR}$ ↓			150	ns	
25	Data Hold from $\overline{WR}$ ↑	60		200	ns	
26	$\overline{WR}$ Repetition Rate	1.2	1.6	2.0	μs	See Note 8
27	$\overline{WR}$ Float from $\overline{BCS}$ ↑	0		100	ns	



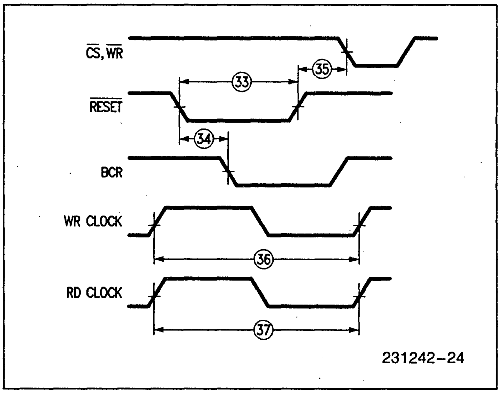
231242–22

MISCELLANEOUS TIMING

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
28	BDRQ Reset from BRDY	20		200	ns	
29	BRDY Pulse Width	400			ns	See Note 4
30	$\overline{BCR}$ Pulse Width	1.4	1.6	1.8	μs	See Notes 9, 13, 15
31	STEP Pulse Width	1.5	1.6	1.7	μs	Step Rate = 3.2 μs/step
		7.6	8.0	8.4	μs	All other step rates, See Notes 14, 15
32	INDEX Pulse Width	500			ns	
33	$\overline{RESET}$ Pulse Width	24			WR CLK	See Note 2
34	$\overline{RESET}$ ↓ to $\overline{BCR}$ ↓	0	1.6	3.2	μs	See Notes 1, 15
35	$\overline{RESET}$ ↑ to $\overline{WR}$, $\overline{CS}$ ↓	6.4			μs	See Note 1
36	WR CLOCK Frequency	0.25	5.0	5.25	MHz	50% Duty Cycle
37	RD CLOCK Frequency	0.25	5.0	5.25	MHz	See Note 5



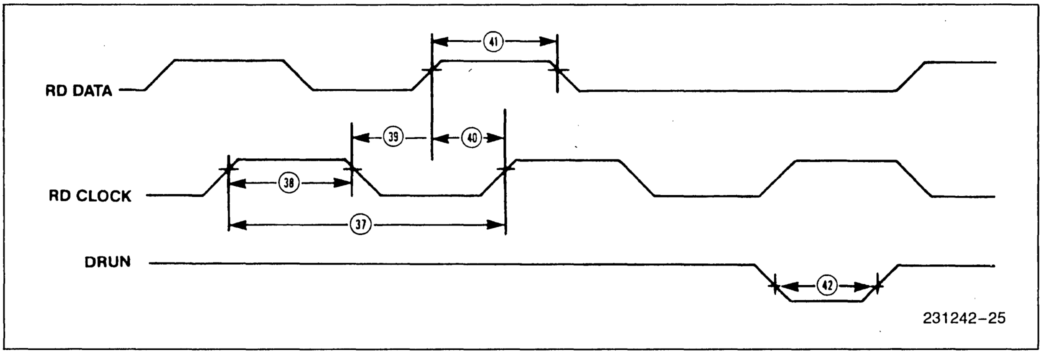
231242-23



231242-24

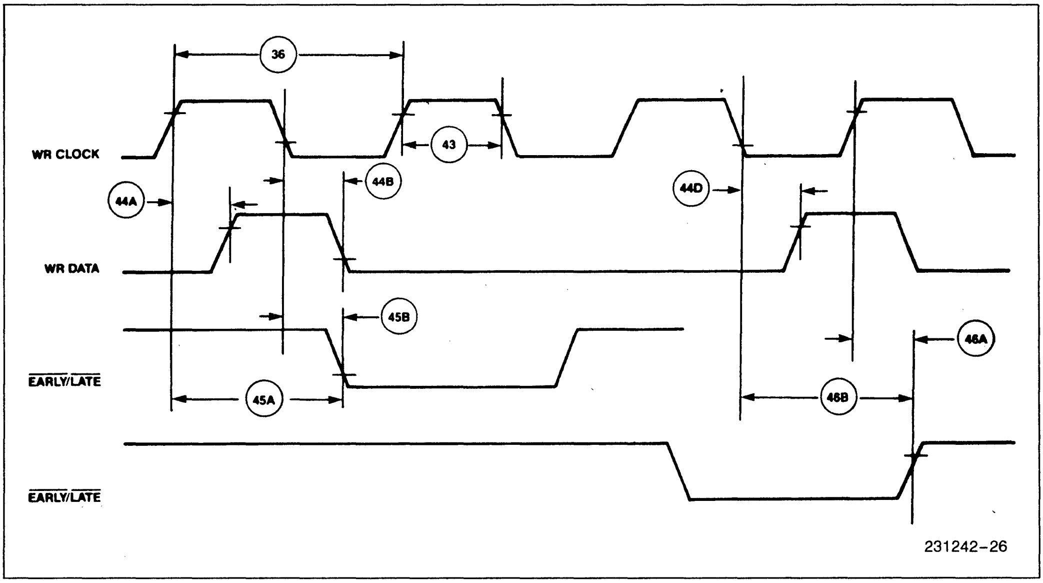
READ DATA TIMING WR CLOCK = 5.0 MHZ

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
38	RD CLOCK Pulse Width	95		2000	ns	50% Duty Cycle
39	RD DATA after RD CLOCK ↓	10			ns	
40	RD DATA before RD CLOCK ↑	20			ns	
41	RD DATA Pulse Width	40		T38/2	ns	
42	DRUN Pulse Width	30			ns	

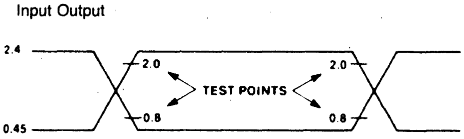


WRITE DATA TIMING WR CLOCK = 5.0 MHZ

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
43	WR CLOCK Pulse Width	95		2000	ns	50% Duty Cycle
	Propagation Delay					
44A	WR CLOCK ↑ to WR DATA ↑	10		65	ns	
44B	WR CLOCK ↓ to WR DATA ↓					
44D	WR CLOCK ↓ to WR DATA ↑					
45A	WR CLOCK ↑ to $\overline{\text{EARLY}}/\overline{\text{LATE}}$ ↓	10		65	ns	
45B	WR CLOCK ↓ to $\overline{\text{EARLY}}/\overline{\text{LATE}}$ ↓					
46A	WR CLOCK ↑ to $\overline{\text{EARLY}}/\overline{\text{LATE}}$ ↑	10		65	ns	
46B	WR CLOCK ↓ to $\overline{\text{EARLY}}/\overline{\text{LATE}}$ ↑					



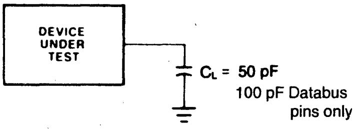
A.C. TESTING INPUT, OUTPUT WAVEFORM



231242-27

AC 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



231242-28

C_L Includes Jig Capacitance

NOTES

1. Based on WR CLOCK = 5.0 MHz
2. 24 WR CLOCK periods = 4.8 μs at 5.0 MHz.
3. 2 WR CLOCK periods ± 100 ns.
4. Previous restrictions on BRDY no longer apply. There are no restrictions on when BRDY may come. BRDY may be connected directly to BDRQ.
5. WR CLOCK Frequency = RD CLOCK Frequency ± 15%.
6. $\overline{RD}$ may be asserted before $\overline{CS}$ as long as it remains active for at least the minimum T3 pulse width after $\overline{CS}$ is asserted.
7. $\overline{WR}$ may be asserted before $\overline{CS}$ as long as it remains active for at least the minimum T11 pulse width after $\overline{CS}$ is asserted.
8. 8 WR CLOCK periods ±2 WR CLOCK periods.
9. 8 WR CLOCK periods ±1 WR CLOCK period.
10. V_{IL} = GND, V_{IH} = V_{CC}, Outputs Open.
11. V_{IL} = 0.8V, V_{IH} = 2.0V, Outputs Open.
12. WR CLOCK & RD CLOCK = DC, V_{IL} = 0V, V_{IH} = V_{CC}, all output open, $\overline{CS}$ inactive.
13. This specification is for $\overline{BCR}$ pulse width during command execution. $\overline{BCR}$ is also triggered by $\overline{RESET}$. In this case, $\overline{BCR}$ pulse width is greater than $\overline{RESET}$ pulse width.
14. 40 WR Clocks ±2.
15. Specification represents actual functionality of 82064 and WD2010. Previous datasheets contain typographical errors.

intel®

APPLICATION NOTE

AP-402

November 1987

Multimodule™ Winchester Controller Using the CHMOS 82064

J. SLEEZER
TECHNICAL MARKETING

Order Number: 231927-001

1.0 INTRODUCTION

The 82064 Winchester Disk Controller (WDC) was developed to ease the complex task of interfacing Winchester disk drives to microprocessor systems. Specifically, the 82064 WDC interfaces to drives that conform to the ST506 specification, which is the dominant interface for 5¼ inch drives. This Application Note provides some background on the 82064 WDC, the drive interfaces and general software routines. It concludes with a design example using the 82064 WDC interfaced to the SBX™ bus. Appendix B contains the listing of the software necessary to operate this controller board.

1.1 ST506 Winchester Drive Overview

Since the 82064 WDC interfaces only to drives conforming to the ST506 specification, this overview will limit itself to those drives. A summary of the ST506 specification is shown in Appendix A for those who are not familiar with it. The ST506 Winchester Disk contains from 1 to 8 hard disks (or platters) with the average being 2 to 3 disks. These disks are made from aluminum (hence the term hard disk) and are coated with some type of recording media. The recording media is typically made of magnetic-oxide, which is similar to the material used on floppy disks and cassette tapes. Each side of a hard disk is coated with recording media and each side can store data. Each surface of a disk has its own read/write head.

Hard disk drives are sealed units because the R/W heads actually fly above the disk surface at about 8 to 20 microinches. A piece of dust or dirt, which appears as a boulder to the gap between the heads and the disk surface, will wreak havoc upon the disk media.

The R/W heads are mechanically connected together and move as a single unit across the surface of the disk. There are 2 basic methods for positioning the heads. The first is with stepper motors, which is the most common method and is also used on most floppy disk drives. These positioners are used mainly because of their low cost.

The second method of positioning the heads is to use a voice-coil mechanism. These units do not move in steps but swing across the disk. These mechanisms generally permit greater track density than steppers, but also require complex feedback electronics which increases the cost of the drive. Generally, voice-coil head positioners use closed loop servo positioning, as compared to the open loop positioning used with stepper motors.

The surface of a disk is divided logically into concentric circles radiating from the center as shown in Figure 1. Each concentric circle is called a track.

The group of tracks, all in the same position, on all of the disks (platters) in the drive is collectively called a cylinder. The number of tracks on a surface (which affects storage density) is determined by the head positioners. Typically, stepper head positioners have fewer tracks than drives that use a voice coil positioner. Which type of positioner is used is irrelevant to the 82064 as positioners are part of the drive electronics. The 82064 can access up to 2048 tracks per surface.

Once the surface is divided into cylinders it is further divided radially (as with a pie). The area between the radial spokes is referred to as a sector. The number of sectors per track is determined by many variables, but is basically determined by the number of data bytes and the length of the ID field (which locates a sector). Figure 2 shows one manufacturer's specifications for their drive. The manufacturer formats the drive with 32–256 byte sectors per track. Alternatively, the drive could be reformatted to contain 17–512 byte sectors per track. This second option has fewer sectors per track but stores more data. Determining how many bytes each sector contains is done by extensive analysis of the hardware and operating system. The 82064 WDC is programmable for sector size during formatting.

The order in which sectors are logically numbered on the track is called interleaving. An interleave factor of four would have three sectors separating logically sequential sectors. Starting at the index pulse, an example of four way interleaving is:

Sector 1, Sector X, Sector Y, Sector Z, Sector 2, Sector . . .

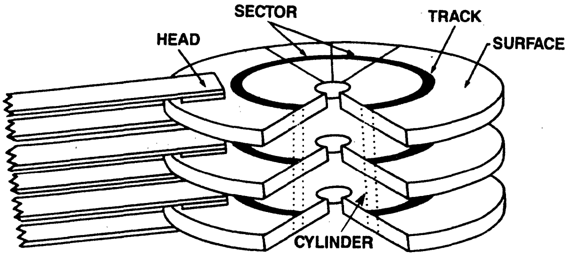


Figure 1

Capacity	
Unformatted	
Per Drive	6.38 Megabytes
Per Surface	1.59 Megabytes
Per Track	10416 Bytes
Formatted	
Per Drive	5.0 Megabytes
Per Surface	1.25 Megabytes
Per Track	8192 Bytes
Per Sector	256 Bytes
Sectors per Track	32
Transfer Rate	5.0 Megabits per second
Access Time	
Track to Track	3 ms
Average (Inc. Settle)	170 ms
Maximum (Inc. Settle)	500 ms
Settling Time	15 ms
Average Latency	8.33 ms
Functional Specifications	
Rotational speed	3600 rpm ± 1%
Recording density	7690 bpi max
Flux density	7690 fci
Track density	255 tpi
Cylinders	153
Tracks	612
R/W Heads	4
Disks	2

Figure 2. A Typical Drive Specification

Interleaving is used primarily because one sector at a time is transferred from disk to sector buffer to system RAM. This transferring of data takes time, and causes a delay between the first sector transferred and sectors that follow it. Without interleaving, the delay in transferring data would result in sectors on the disk rotating past the heads before they could be read. The operating system would then have to wait one disk revolution to get to the next sector (a 16.7 msec delay). With interleaved sectors, the next logical sector would be positioned beneath the heads after the previous sector of data had been transferred to the system RAM. Interleaving unfortunately slows down the overall transfer rate from the disk. A 5 Mbit/second transfer rate averages out to a 1.25 Mbit/second transfer rate when many sectors are transferred with four way interleaving. Again, how much interleaving to use is determined by extensive hardware/software benchmarking.

Whenever data is stored on a multiple platter disk drive, the same track on all surfaces whould be used before repositioning the heads to another track. Repositioning the heads generates a longer delay due to the mechanical delay of moving the heads. Switching to another head incurs no mechanical positioning delay. Only one head can be selected at a time.

Hard disk drives tend to be faster than floppies for two reasons. The speed at which the disk spins is about 10 times faster than the floppy (a floppy spins at 360 rpm for the popular double density disk drives). This yields an immediate one-tenth reduction in access times for the same size drive. While both ST506 drives and floppies use stepper motors, the steppers utilized by the hard disk drives are approximately twice as fast as those used by floppies.

2.0 82064 WINCHESTER DISK CONTROLLER

The 82064 WDC provides most of the functions necessary to interface between a microprocessor and an ST506 compatible disk drive. The 82064 converts the high level commands and parallel data of a microprocessor bus into ST506 compatible disk control signals and serial MFM encoded data. This section presents a detailed description of the 82064 and a discussion of various techniques which can be used to interface the 82064 to a microprocessor.

The internal structure of the 82064 is divided into several sections as shown in Figure 3. They are:

1. the microprocessor interface which includes the status and task registers;
2. sector buffer control;
3. the drive interface;
4. the data transfer section, which includes the MFM encoding/decoding of microprocessor data;
5. and CRC/ECC generation and checker.

2.1 Clock Inputs

The 82064 has two clock inputs: read clock (RD CLOCK) and write clock (WR CLOCK). The PLA controller, the processor interface, buffer control and MFM encoding sections operate off the WR CLOCK input. The RD CLOCK input is used only for decoding the MFM data stream. The clocks may be asynchronous to one another. Both clocks have non-TTL compatible inputs. The easiest method to interface to TTL requires a pull-up resistor to satisfy their input voltage needs. The resistor's value must be compatible with the VIL specification of these pins. See the Pin Descriptions Section for more specific information.

2.2 Microprocessor Interface

The microprocessor interface of the 82064 contains the control logic which permits commands and data to be transferred between the host and the 82064. The interface consists of an 8 bit, tri-state, bidirectional data bus; the task registers; a 3 to 8 address decoder for selecting one of the seven registers; and the general read, write, and chip select logic. Externally, the 82064 expects a buffer equal in size to a sector on the disk, and tri-state

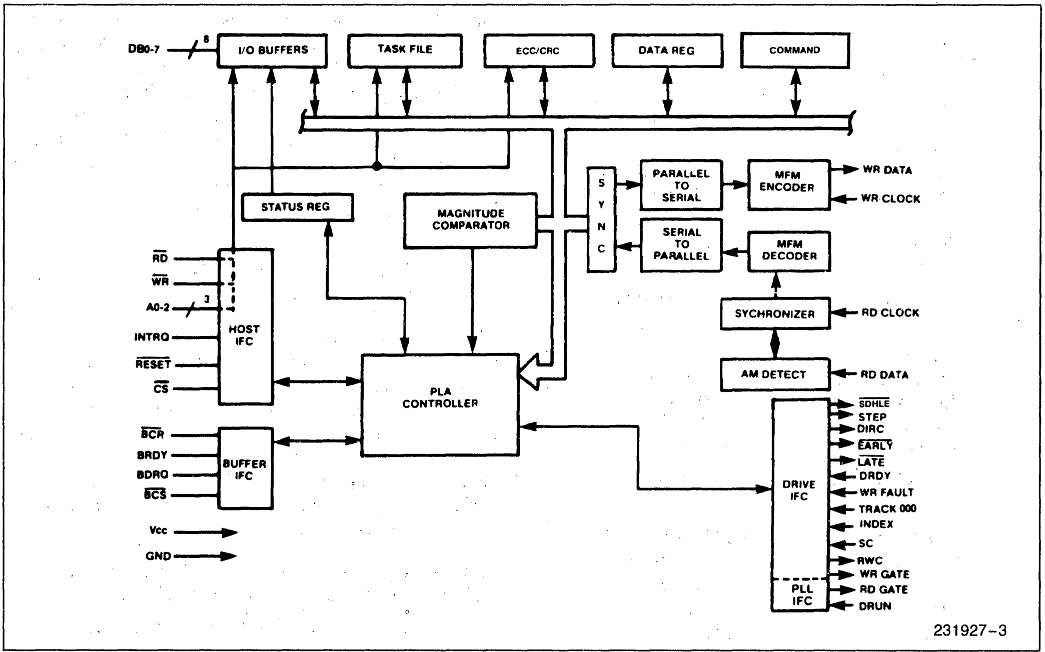


Figure 3. 82064 Internal Block Diagram

transceivers between the sector buffer and the microprocessors data bus in order to isolate itself from the microprocessor during disk data transfers.

A0–A2, Data Bus

These three address lines are active high signals and select one of the seven register locations in the 82064. They are not latched internally. If the three addresses are equal to 0 and the 82064 is selected, the data bus is kept tri-stated to ease interfacing to a sector buffer. The 82064's data bus is controlled by both the microprocessor and the 82064. The microprocessor has control for loading the registers and command. During disk reads or writes, control switches to the 82064 so that it may access the local sector buffer when transferring data between the disk and the buffer.

$\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{CS}}$

The chip select ($\overline{\text{CS}}$) is typically decoded from the higher order address lines. $\overline{\text{CS}}$ only permits data to be placed into, or read from, the 82064's task registers. Once a disk operation starts, $\overline{\text{CS}}$ no longer efffects the 82064. $\overline{\text{RD}}$ and $\overline{\text{WR}}$ are bidirectional lines and are used to read or write the 82064's registers by the host microprocessor and are valid only if $\overline{\text{CS}}$ is present. The 82064 will drive $\overline{\text{RD}}$ and $\overline{\text{WR}}$ when transferring data between the sector buffer and the disk. A signal is provided to tri-state the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ lines from the host during a buffer access. This is covered in the Sector Buffer Control Section.

Interrupts

An interrupt is issued at the end of all commands, and the interrupt is cleared by reading any register. For the Read Sector command only, the 82064 allows the user the option of an interrupt either at the termination of the command, as is the case with all other commands, or when data needs to be transferred to the host from the sector buffer. This is discussed further in the Interrupt Mode Section. When selecting the data transfer option, the interrupt line will go active at the same time as the BDRQ line and the interrupt will be removed only when the proper handshake occurs with the sector buffer.

Task Registers

The Task Register File contains the command, status, track number, sector number, and other information necessary to properly execute a command. These registers are accessed with A0–A2, $\overline{RD}$ (or $\overline{WR}$), and $\overline{CS}$ being valid and are not cleared by a reset. The registers are covered in detail in the Task Register File Section.

2.3 Sector Buffer Control

The 82064 was designed to operate with an external buffer equal in size to one sector. To ease the design-in of this buffer, the 82064 provides all of the control signals it needs to operate the buffer. This buffer must be isolated from the system bus, using tri-state buffers, during disk transfers to prevent contention during the period that the 82064 is accessing the buffer. A sector buffer is generally used to ease interfacing to the system due to the high disk data rates (625 kbytes/sec), although it is not required.

$\overline{BCS}$

The Buffer Chip Select ($\overline{BCS}$) line goes active whenever the 82064 is accessing the sector buffer. This signal should remove the microprocessors ability to access the 82064 and sector buffer and must enable the sector buffer for use by the 82064.

At a 5 Mbit/sec disk data rate, the 82064 will access the buffer every 1.6 microseconds (8 bits × 200 ns/bit). $\overline{BCS}$ will remain low the entire time the 82064 is accessing the buffer. The 82064 will pulse the appropriate $\overline{RD}$ or $\overline{WR}$ line for each byte transferred.

$\overline{BCR}$

Buffer Counter Reset ($\overline{BCR}$) goes active each time that $\overline{BCS}$ changes state. Its purpose is to reset the address counter of the sector buffer back to zero before and after the 82064 uses the sector buffer. Its function is optimized for single sector transfers. Multiple sector transfers should use a software controlled buffer counter reset and not use $\overline{BCR}$ as the sector buffer will be reset to the beginning after each sector is transferred.

BDRQ, BRDY

Buffer Data Request (BDRQ) and Buffer Ready (BRDY) provide the handshake needed to transfer data between the sector buffer and the host. BDRQ signals that data must be moved to/from the sector buffer and the host. BRDY has two functions. Once the transfer signaled by BDRQ is finished, asserting BRDY will inform the 82064 that the transfer is completed and that it may finish executing the command. BRDY is also used in multiple sector commands. BRDY going high during a multiple sector transfer indicates that the buffer is full (or empty—depending upon the command) and the transfer should wait until the buffer is serviced. The sector that was being transferred will finish and the 82064 will deactivate $\overline{BCS}$ and activate BDRQ. The host microprocessor must then transfer the data between the buffer and system memory. When this transfer is finished, asserting BRDY will cause the 82064 to resume the command.

The handshaking between BDRQ and BRDY occurs only in full sector increments—not on a byte basis. A high on BDRQ indicates a full sector's worth of data is required; BRDY going high indicates a full sector of data is available to the 82064 without interruption.

Only the rising edge of BRDY is valid. A falling edge may occur at any time without effect. $\overline{BCR}$ will pulse and $\overline{BCS}$ will go active eight byte times (8 bytes × 8 bits/byte × 200 ns/bit = 12.8 microseconds) before the first data byte is transferred from the sector buffer to the disk.

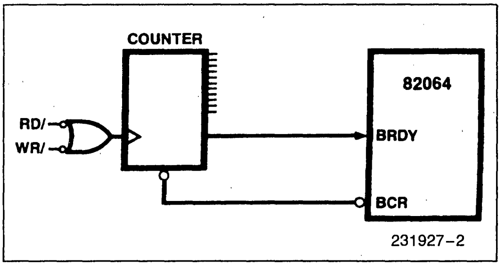


Figure 4. BRDY Generation Logic

2.4 Data Transfer Logic

This section of the 82064 is responsible for conversion of serial disk data to parallel data (and vice versa); encoding/decoding of the disk's MFM serial bit stream; and detecting the address mark.

Polled Interface

Since the 82064 isolates itself from the host during several commands, the host cannot read the status register during some periods to determine what course should be taken. In Figure 10, trying to read the status register when $\overline{BCS}$ is active will return indeterminate data. To prevent the microprocessor from reading this indeterminate data, a hardware generated "Busy" pattern should be driven onto the data bus if $\overline{BCS}$ is active. This is shown in Figure 11. The status register contains a data request (DRQ) bit whose timing is equal to the BDRQ output signal, thus making a polled operation possible. DRQ will stay set in the status register until a BRDY is generated.

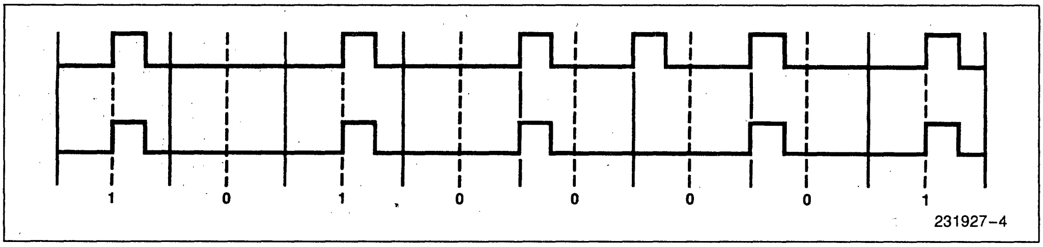


Figure 5. Data Address Mark

One design issue with the polled interface occurs when the microprocessor is polling the status and the 82064 deactivates $\overline{BCS}$. The microprocessor would normally read the hardware busy pattern. If $\overline{BCS}$ is deasserted, the hardware pattern is disabled and the microprocessor will start to read the real status register. The read cycle may almost be finished, and the read access period of the 82064 will not be satisfied. The data returned to the microprocessor will be invalid.

Interrupt Interface

There are cases where the designer does not want to tie up the microprocessor with polling. The typical 82064 design will need two interrupts per command. One for a data transfer and one for the completion of the command. The 82064 has an output to issue an interrupt when the command has finished. However for data transfers an interrupt must be generated from the BDRQ line as shown in Figure 12 (whether a DMA controller is used or not). When a data transfer is needed, the 82064 will activate the BDRQ line. The microprocessor will be interrupted and do the data transfer function. BDRQ will stay active until BRDY is generated, so the system must either use edge triggered interrupts or must not write the end-of-interrupt byte until BDRQ is removed (this is true of Intel's 8259A).

MFM Encoding/Decoding

The MFM encoding section will receive 8 bit parallel data when a valid command has been recognized and BRDY has gone high. The parallel data is first serialized and converted to an intermediate, NRZ encoded, data stream. The serial NRZ data is sent to the MFM encoding section and then transferred to the disk. Decoding of the MFM bit stream (during disk reads) happens in reverse order.

The control logic operates off the write clock (WR CLOCK) running at a frequency of the desired tranfer rate. The MFM decoding portion operates off of the read clock (RD CLOCK) input, which is supplied by an external phase lock loop. The two clocks need not be synchronized to each other. Data is written (and hence read) with the most significant bit first.

Address Mark Detector

The address mark is a unique 2 byte code written at the beginning of each ID field and data field. This address mark serves two purposes. It tells the controller what type of data is about to be received so that internal computations can be performed, and to ensure that ID fields are not sent to the host. The second purpose is to align the serial data back to the original 8 bit boundaries that existed when data was written (there are no byte boundaries on a disk).

An address mark is always preceded by the all zeros synchronization field. The 82064 starts comparing the incoming data stream when the synchronization field ends. A high speed comparator is used since the 82064 does not yet know where the proper byte boundaries are. When a proper comparison of the address mark is made the controller starts assembling bytes, starting with the second byte of the address mark.

The first byte of the address mark is an "A1" Hex, but purposely violates the MFM encoding rules by removing a clock pulse. In Figure 5, the first example is of a normal MFM encoded A1H. The second example is of the address mark and shows the missing clock pulse. The non-MFM compatible A1 is to prevent the host from issuing a similar data byte and possibly confusing detection logic.

The second byte specifies either an ID or data field and is encoded according to normal MFM rules. It is either an "F8" Hex for a data field, or "FC" through "FF" for an ID field. The different values correspond to a range of cylinders on the drive in increments of 256 tracks. The 82064 makes no use of this information, but writes it for compatibility with the ST506 specification during formatting.

PLA Control

The PLA Controller interprets command sent by the microprocessor. Its operation is synchronized to the WR CLOCK input. The PLA controller is started when a command is written into the command register. It generates control signals and operates in a handshake mode when communicating with the MFM decoding block.

Magnitude Comparator

A 10 bit magnitude comparator is used to calculate the direction and number of step pulses needed to move the head from the present cylinder position to the desired position. A separate high speed equivalence comparator is used to compare ID field bytes when searching for a sector ID field.

2.5 CRC/ECC Generator and Checker

The 82064 provides two options for protecting the integrity of the data field. The data field may have either a CRC (SDH register, bit 7 = 0), or a 32-bit ECC (SDH register, bit 7 = 1) appended to it. The ID field is always protected by a CRC.

CRC Generation/Checking

The CRC generator computes and checks the cyclic redundancy check bytes that are appended to the ID and data fields. CRC generation/checking is always done on ID fields. Data fields have a choice between 82064 CRC, internal ECC or externally supplied ECC. The CRC mode is chosen by setting bit 7 of the SDH register low. The CRC mode provides a means of verifying the accuracy of the data read from the disk, but does not attempt to correct it. The CRC generator computes and checks cyclic redundancy check characters that are written and read from the disk after the ID and data fields.

The generator polynomial for the CRC-CCITT (CRC-16) code is:

x16 + x12 + x5 + 1 = (x + 1) (x15 + x14 + x13 + x12 + x4 + x3 + x2 + x + 1)

The code's capability is as follows:

a) Detects all occurrences of an odd number of bits in error.

b) Detects all single, double, and triple bit errors if the record length (including check bits) is less than 32,767 bits.

c) Detects all single-burst errors of sixteen bits or less.

d) Detects 99.99695% of all possible 17 bit burst errors, and 99.99847% of all possible longer burst, assuming all errors are possible and equally probable.

The CRC code has some double-burst capability when used with short records (sectors). For a 256 byte sector the code will detect double-bursts as long as the total number of bits in error does not exceed 7.

If the CRC character generated while reading the data does not equal the one previously written, an error exists. If an ID field CRC error occurs the "ID not found" bit in the error register will be set. If a data field CRC error occurs the "ECC/CRC" bit in the error register will be set.

ECC Generation/Checking

The ECC mode is only applicable to the data field. It provides the user with the ability to detect and correct errors in the data field automatically. The commands and registers which must be considered when ECC is used are:

1. SDH Register, bit 7 (CRC/ECC)
2. READ SECTOR Command, bit 0 (T)
3. READ SECTOR and WRITE SECTOR Commands, bit 1 (L)
4. COMPUTE CORRECTION Command
5. SET PARAMETER Command
6. STATUS Register, bit 2—error correction successful
7. STATUS Register, bit 0—error occurred
8. ERROR Register, bit 6—uncorrectable error

To enable the ECC mode, bit 7 of the SDH register must be set to one.

Bit 0 (T) of the READ Command controls whether or not error correction is attempted. When T = 0 and an error is detected, the 82064 tries up to 10 times to correct the error. If the error is successfully corrected, bit 2 of the STATUS Register is set. The host can interrogate the status register and detect that an error occurred and was corrected. If the error was not correctable, bit 6 of the ERROR Register is set. If the correction span was set to 5 bits, the host may now execute the SET PARAMETER Command to change the correction span to 11 bits, and attempt the read again. If the error persists, the host can read the data, but it will contain errors.

When T = 1 and an error is detected, no attempt is made to correct it. Bit 0 of the STATUS Register and bit 6 of the ERROR Register are set. The user now has two choices:

1. Ignore the error and make no attempt to correct it.
2. Use the COMPUTER CORRECTION Command to determine the location and pattern of the error, and correct it within the user's program.

When the COMPUTE CORRECTION Command is implemented, it must be done before executing any command which can alter the contents of the ECC Register. The READ SECTOR, WRITE SECTOR, SCAN ID, and FORMAT Commands will alter this register and correction will be impossible. The COMPUTE CORRECTION Command may determine that the error is uncorrectable, at which point the error bits in the STATUS and ERROR Registers are set.

Although ECC generation starts with the first bit of the F8H byte in the data ID field, the actual ECC bytes written will be the same as if the A1H byte was included. The ECC polynomial used is:

$$X^{32}+X^{28}+X^{26}+X^{19}+X^{17}+X^{10}+X^{6}+X^{2}+1$$

For automatic error correction, the external sector buffer must be implemented with a static RAM and counter, not with a FIFO.

The SET PARAMETER Command is used to select a 5-bit or 11-bit correction span.

2.6 Drive Interface

The drive interface of the 82064 contains the logic that makes possible the storage and reliable recovery of data. This interface consists of the drive and head select logic, the disk control signals, and read and write data logic as shown in Figure 6. This section describes the external circuitry which is required to complete the 82064's drive interface.

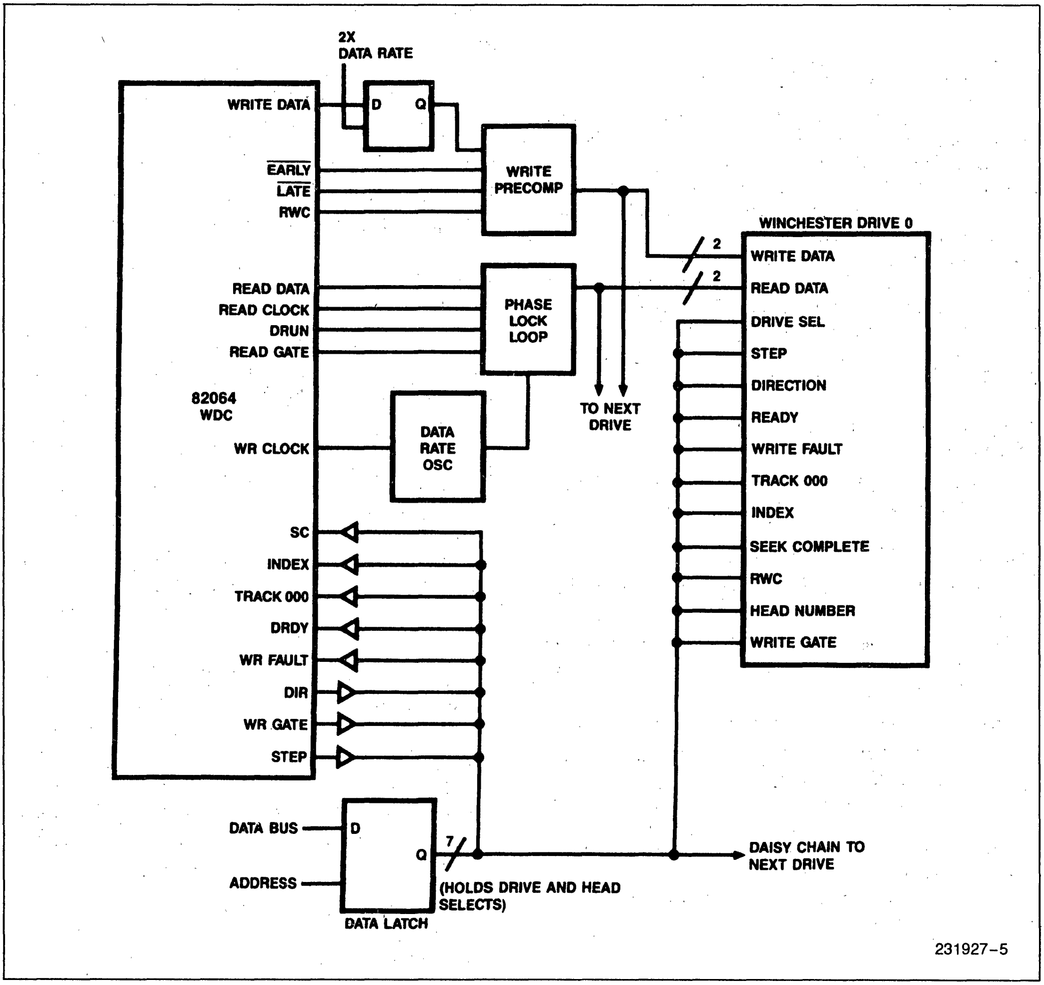


Figure 6. Drive Interface

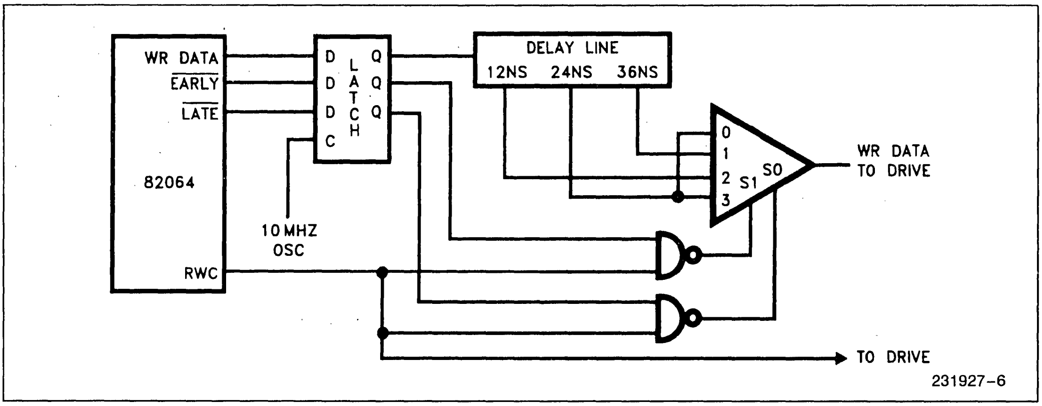


Figure 7. Write Precompensation Logic

Drive/Head Select

The 82064 has no outputs for selecting the head or drive. Therefore these signals must be generated by the user as shown in Figure 6. Data bits 0–4 should be latched whenever the SDH register is written. Bits 0–2 would then be driven onto the drive cable with open collector buffers. Bits 3 and 4 would be decoded after being latched, then buffered for the cable. The head information written to the 82064's SDH register is used to write the proper ID fields during formatting. Changing the drive bits in the SDH register will cause a Scan ID to be performed by the 82064 to update non user accessible registers.

Drive Control

The drive control (STEP, DIR, WR FAULT, TRACK 000, INDEX, SC, RWC, and WR GATE) signals are merely conditioned for transmission over the drive cable. The purpose of each pin can be found in the section on Pin Descriptions and their use in the Command Section.

WR DATA, $\overline{\text{EARLY}}$, $\overline{\text{LATE}}$

Figure 7 is a diagram of the interface required on the write data line. The final stage of the MFM encoding requires applying the WR DATA to an external flip-flop clocked at 10 MHz. The 82064 monitors the serial write data output for particular bit patterns that require precompensation to prevent bit shifting. $\overline{\text{EARLY}}$ and $\overline{\text{LATE}}$ are active on all cylinders and will normally require that RWC be factored into them to activate the data precompensation on the proper cylinder.

A delay line is required to generate the delayed data for precompensation since the actual delay varies between drive manufacturers. $\overline{\text{EARLY}}$ and $\overline{\text{LATE}}$ go active in the same clock period that generates the data bit to be shifted.

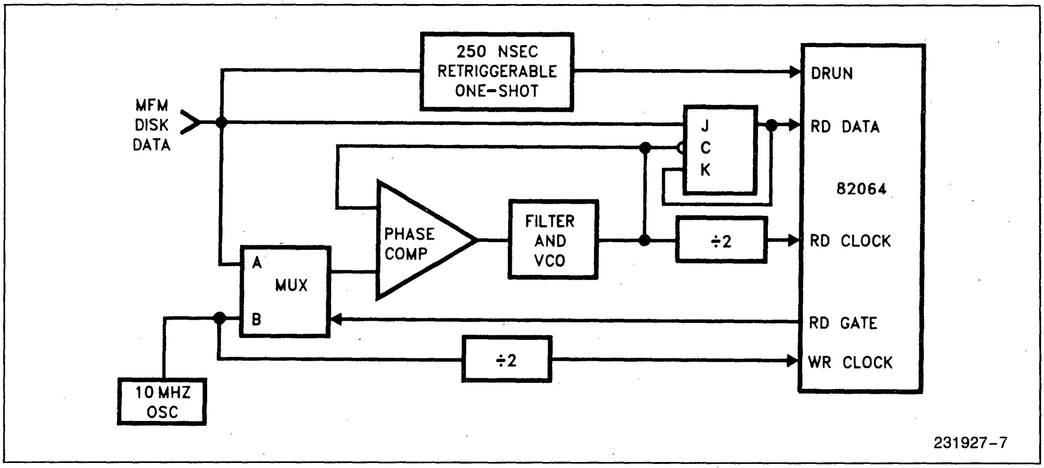


Figure 8. Data Separator Circuit

RD Data, DRUN, RD Gate

The read data interface is shown in Figure 8, and consists of the data run (DRUN) signal and a phase lock loop to generate the RD CLOCK input to decode the serial data. DRUN is generated from a retriggerable one-shot with a period just exceeding one bit cell. A sync field consisting of a string of clock pulses will continually retrigger the one-shot producing a steady high level on DRUN. The 82064 counts off 16 clock pulses internally, and if DRUN is still active, will make RD GATE active. Any byte other than an address mark will deactivate RD GATE and the sequence starts over.

The phase lock loop generates RD CLOCK which is used to decode the incoming serial data. Until RD GATE is activated by the 82064, the phase lock loop (PLL) should be locked onto a local 10 MHz clock to minimize PLL lock-up times. When RD GATE is activated, the PLL starts locking onto the incoming data stream, which should consist of the all zeros sync field. Once the PLL locks onto this synch field, the 82064 will start examining the serial data for a non-zero byte. A non-zero byte will be indicated by DRUN dropping since the address mark follows the sync field and is an "A1" Hex. This sequence is shown in Figure 9. If the address mark is detected, and if it was preceded by at least 9 bytes of zeros, RD GATE will stay active. The 82064 will then assemble bytes of data, and ensure the proper ID field is found. If a non-zero or non-address mark byte was detected, RD GATE will go inactive for a minimum of 2 byte times. If a data field or the wrong ID field is detected, or the ID field was not preceded by 8 bytes of zeros, then RD GATE goes inactive and the sequence starts over with the 82064 examining the DRUN input.

2.7 Microprocessor Interfaces

This section shows the general 82064 interfaces to a microprocessor system. There are essentially four interfaces which consist of a combination of polled, DMA, and interrupts. While the 82064 was designed to interface directly to one type, it accommodates all with minor additional logic.

DMA Interface

The 82064 is designed to use a DMA controller for data transfer between its sector buffer and the host system, and to interrupt the host when the command has finished. This interface is shown in Figure 10.

When the 82064 determines that a transfer is needed between the sector buffer and the host (either at the beginning of a command or through BRDY going active in a multiple sector transfer), it will assert BDRQ. BDRQ will initiate a DMA transfer via the DMA request input. The DMA controller will generate reads or writes which will increment an address counter. BRDY indicates that the data transfer has finished and is issued off the carry-out line (or high order address line) of the counter. The 82064 will assert BDRQ at this point and activate $\overline{\text{BCS}}$ to prevent the host from intefering with disk/buffer transfers. There can be no polling for a data transfer or a register read without an interrupt in this scheme.

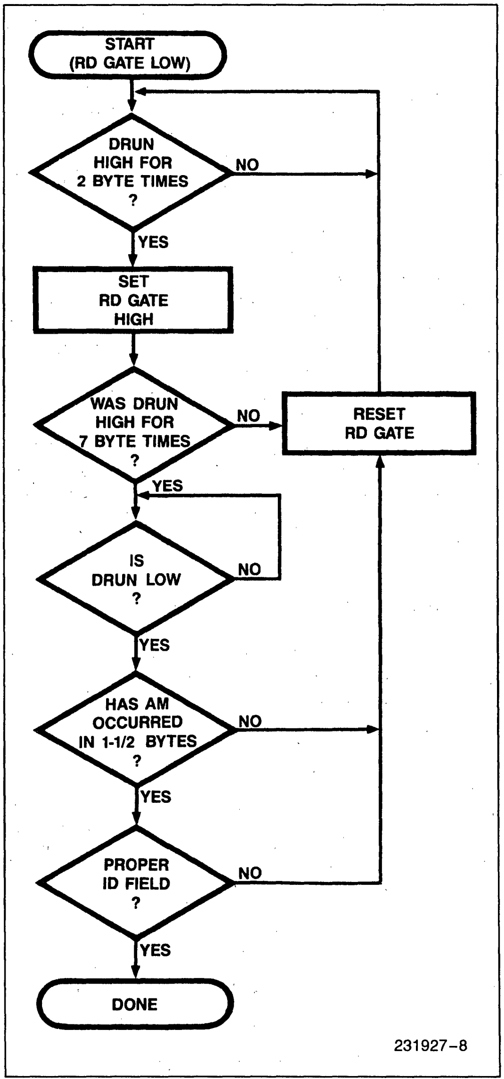


Figure 9. PLL Control Sequence

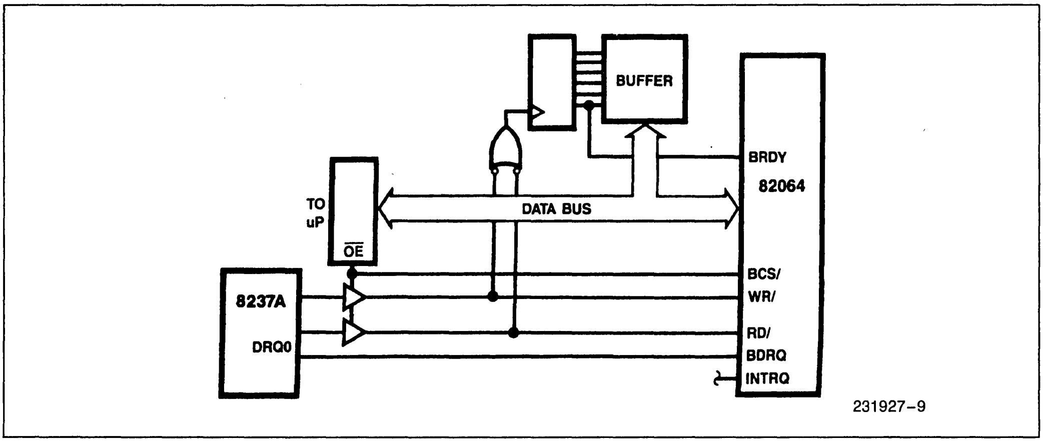


Figure 10. 82064 DMA Interface

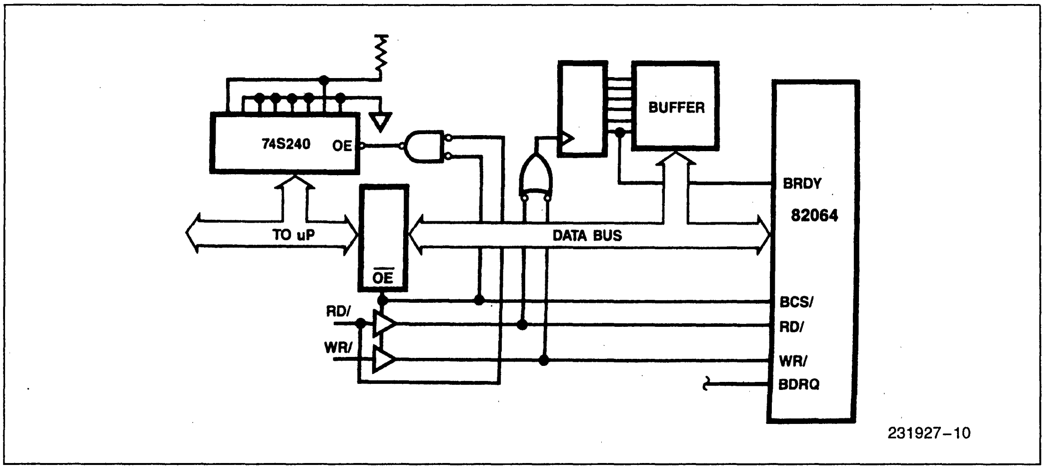


Figure 11. 82064 Polled Interface

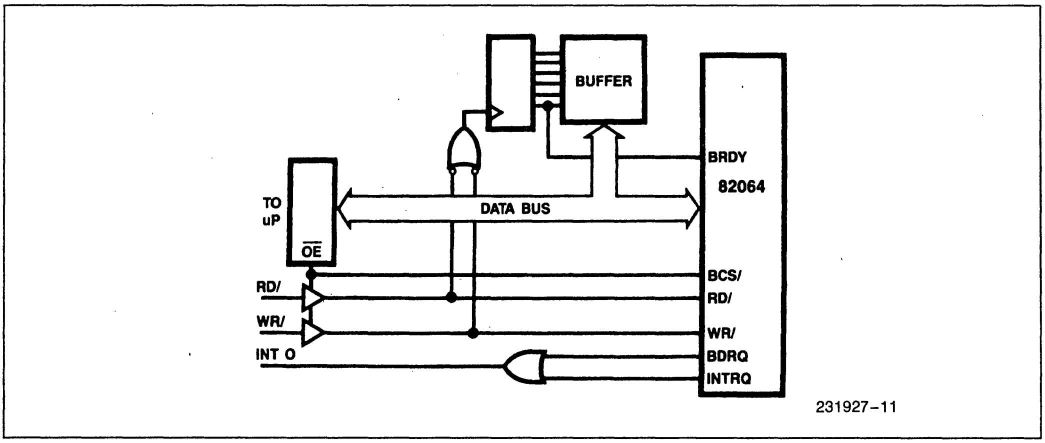


Figure 12. 82064 Interrupt Interface

3.0 PIN DESCRIPTIONS

Symbol	Pin No.		Type	Name and Function
	DIP	PLCC		
$\overline{\text{BCS}}$	1	1	O	**BUFFER CHIP SELECT:** Output used to enable reading or writing of the external sector buffer by the 82064. When low, the host should not be able to drive the 82064 data bus, $\overline{\text{RD}}$, or $\overline{\text{WR}}$ lines.
$\overline{\text{BCR}}$	2	2	O	**BUFFER COUNTER RESET:** Output that is asserted by the 82064 prior to read/write operation. This pin is asserted whenever $\overline{\text{BCS}}$ changes state. Used to reset the address counter of the buffer memory.
INTRQ	3	3	O	**INTERRUPT REQUEST:** Interrupt generated by the 82064 upon command termination. It is reset when the STATUS register is read, or a new command is written to the COMMAND register. Optionally signifies when a data transfer is required on Read Sector commands.
$\overline{\text{SDHLE}}$	4	4	O	$\overline{\text{SDHLE}}$ is asserted when the SDH register is written by the host.
$\overline{\text{RESET}}$	5	7	I	**RESET:** Initializes the controller and clears all status flags. Does not clear the Task Register File.
$\overline{\text{RD}}$	6	8	I/O	**READ:** Tri-state, bi-directional signal. As an input, $\overline{\text{RD}}$ controls the transfer of information from the 82064 registers to the host. $\overline{\text{RD}}$ is an output when the 82064 is reading data from the sector buffer ($\overline{\text{BCS}}$ low).
$\overline{\text{WR}}$	7	9	I/O	**WRITE:** Tri-state, bi-directional signal. As an input, $\overline{\text{WR}}$ controls the transfer of command or task information into the 82064 registers. $\overline{\text{WR}}$ is an output when the 82064 is writing data to the sector buffer ($\overline{\text{BCS}}$ low).
$\overline{\text{CS}}$	8	10	I	**CHIP SELECT:** Enables $\overline{\text{RD}}$ and $\overline{\text{WR}}$ as inputs for access to the Task Registers. It has no effect once a disk command starts.
A_0–A_2	9–11	11–13	I	**ADDRESS:** Used to select a register from the task register file.
DB_0–DB_7	12–19	14–16 18–22	I/O	**DATA BUS:** Tri-state, bi-directional 8-bit Data Bus with control determined by BCS. When BCS is high the microprocessor has full control of the data bus for reading and writing the Task Register File. When BCS is low the 82064 controls the data bus to transfer data to or from the buffer.
V_{SS}	20	23		Ground
WR DATA	21	24	O	**WRITE DATA:** Output that shifts out MFM data at a rate determined by Write Clock. Requires an external D flip-flop clocked at 10 MHz. The output has an active pullup and pulldown that can sink 4.8 mA.
$\overline{\text{LATE}}$	22	25	O	**LATE:** Output used to derive a delay value for write precompensation. Valid when WR GATE is high. Active on all cylinders.
$\overline{\text{EARLY}}$	23	26	O	**EARLY:** Output used to derive a delay value for write precompensation. Valid when WR GATE is high. Active on all cylinders.
WR GATE	24	27	O	**WRITE GATE:** High when write data is valid. WR GATE goes low if the WR FAULT input is active. This output is used by the drive to enable head write current.
WR CLOCK	25	29	I	**WRITE CLOCK:** Clock input used to derive the write data rate. Frequency = 5 MHz for the ST506 interface.

3.0 PIN DESCRIPTIONS (Continued)

Symbol	Pin No.		Type	Name and Function
	DIP	PLCC		
DIR	26	30	O	**DIRECTION:** High level on this output tells the drive to move the head inward (increasing cylinder number). The state of this signal is determined by the 82064's internal comparison of actual cylinder location vs. desired cylinder.
STEP	27	31	O	**STEP:** This signal is used to move the drive head to another cylinder at a programmable frequency. Pulse width = 1.6 μs for a step rate of 3.2 μs/step, and 8.4 μs for all other step rates.
DRDY	28	32	I	**DRIVE READY:** If DRDY from the drive goes low, the command will be terminated.
INDEX	29	33	I	**INDEX:** Signal from the drive indicating the beginning of a track. It is used by the 82064 during formatting, and for counting retries. Index is edge triggered. Only the rising edge is valid.
WR FAULT	30	34	I	**WRITE FAULT:** An error input to the 82064 which indicates a fault condition at the drive. If WR FAULT from the drive goes high, the command will be terminated.
TRACK 000	31	35	I	**TRACK ZERO:** Signal from the drive which indicates that the head is at the outermost cylinder. Used to verify proper completion of a RESTORE command.
SC	32	36	I	**SEEK COMPLETE:** Signal from the drive indicating to the 82064 that the drive head has settled and that reads or writes can be made. SC is edge triggered. Only the rising edge is valid.
RWC	33	37	O	**REDUCED WRITE CURRENT:** Signal goes high for all cylinder numbers above the value programmed in the Write Precomp Cylinder register. It is used by the precompensation logic and by the drive to reduce the effects of bit shifting.
DRUN	34	38	I	**DATA RUN:** This signal informs the 82064 when a field of all ones or all zeroes has been detected in the read data stream by an external one-shot. This indicates the beginning of an ID field. RD GATE is brought high when DRUN is sampled high for 16 clock periods.
BRDY	35	39	I	**BUFFER READY:** Input used to signal the controller that the buffer is ready for reading (full), or writing (empty), by the host μP. Only the rising edge indicates the condition.
BDRQ	36	40	O	**BUFFER DATA REQUEST:** Activated during Read or Write commands when a data transfer between the host and the 82064's sector buffer is required. Typically used as a DMA request line.
RD DATA	37	41	I	**READ DATA:** Single ended input that accepts MFM data from the drive.
RD GATE	38	42	O	**READ GATE:** Output that is asserted when a search for an address mark is initiated. It remains asserted until the end of the ID or data field.
RD CLOCK	39	43	I	**READ CLOCK:** Clock input derived from the external data recovery circuits.
V_{CC}	40	44	I	**D.C. POWER:** +5V.
NC	—	5, 6, 17, 28		**NO CONNECTS**

4.0 TASK REGISTER FILE

The Task Register File is a bank of registers used to hold parameter information pertaining to each command. These registers and their addresses are:

A2	A1	A0	READ	WRITE
0	0	0	(Bus Tri-Stated)	(Bus Tri-Stated)
0	0	1	Error Flags	Reduce Write Current
0	1	0	Sector Count	Sector Count
0	1	1	Sector Number	Sector Number
1	0	0	Cylinder Low	Cylinder Low
1	0	1	Cylinder High	Cylinder High
1	1	0	SDH	SDH
1	1	1	Status Register	Command Register

NOTE:
Registers are not cleared by $\overline{\text{RESET}}$

4.1 Error Register

This read-only register contains specific error status after the completion of a command. If any bit in this register is set, then the Error bit in the Status Register will also be set. The bits are defined as follows:

7	6	5	4	3	2	1	0
BBD	CRC/ECC	0	ID	0	AC	TK000	DM

Bit 7 - Bad Block Detect (BBD)

This bit is set when an ID field has been encountered that contains a bad block mark. The bad block bit is set only during formatting. The 82064 will terminate a command if an attempt is made to read a sector that contains this bit.

Bit 6 - CRC/ECC Data Field Error (CRC/ECC)

When in the CRC mode (SDH register, bit 7 = 0), this bit is set when a CRC error occurs in the data field. When retries are enabled, ten more attempts are made to read the sector correctly. If none of these attempts are successful bit 0 in the STATUS register is also set. If one of the attempts is successful, the CRC/ECC error bit remains set to inform the host that a marginal condition exists; however, bit 0 in the STATUS register is not set.

When in the ECC mode (SDH register, bit 7 = 1), this bit is set when the first non-zero syndrome is detected. When retries are enabled, up to ten attempts are made to correct the error. If the error is successfully corrected, this bit remains set; however, bit 2 of the STATUS register is also set to inform the host that the error has been corrected. If the error is not correctable, the CRC/ECC error bit remains set and bit 0 of the STATUS register is also set.

The data may be read even if uncorrectable errorS exist.

NOTE:
If the long mode (L) bit is set in the READ or WRITE command, no error checking is performed.

Bit 5 - Reserved.

Not used. Set to zero.

Bit 4 - ID Not Found

This bit is set to indicate that the correct cylinder, head, sector, or size parameter could not be found, or that a CRC error occurred in the ID field. Ths bit is set on the first failure and remains set even if the error is recovered on a retry. When recovery is unsuccessful, the Error bit (bit 0) of the STATUS register is also set.

For a SCAN ID command with retries enabled (T = 0), the Error bit in the STATUS register is set after ten unsuccessful attempts have been made to find the correct ID. With retries disabled (T = 1), only two attempts are made before setting the Error bit.

For a READ or WRITE command with retries enabled (T = 0), ten attempts are made to find the correct ID field. If there is still an error on the tenth try, an auto-scan and auto-seek are performed. Then ten more retries are made before setting the Error bit. When retries are disabled (T = 1), only two tries are made. No auto-scan or auto-seek operations are performed.

Bit 3 - Reserved.

Not used. Set to zero.

Bit 2 - Aborted Command

This bit is set if a command was issued or in progress while DRDY (Pin 28) was deasserted or WR FAULT (Pin 30) was asserted. The Aborted Command bit will also be set if an undefined command is written into the COMMAND register, but an implied seek will be executed.

Bit 1 - TRACK 000 Error (TK000)

This bit is set only by the RESTORE command. It indicates that TRACK 000 (Pin 31) has not gone active after the issuance of 2048 stepping pulses.

Bit 0 - Data Address Mark

This bit is set during a READ SECTOR command if the Data Address Mark is not found after the proper Sector ID is read.

4.2 Reduce Write Current Register

This register is used to define the cylinder number where RWC (Pin 33) is asserted:

<table>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">CYLINDER NUMBER / 4</td></tr>
</table>

The value (0–255) written into this register is internally multiplied by 4 to specify the actual cylinder where RWC is asserted. Thus a value of 01H will cause RWC to activate on cylinder 4, 02H on cylinder 8 and so on. RWC will be asserted when the present cylinder is greater than or equal to the cylinder indicated by this register. For example, one ST506 compatible drive requires precompensation on cylinder 128 (80H) and above. Therefore the REDUCE WRITE CURRENT register should be loaded with 32 (20H). A value of FFH will keep the RWC output inactive regardless of the actual cylinder number.

4.3 Sector Count Register

This register is used to define the number of sectors that need to be transferred to the buffer during a READ MULTIPLE SECTOR or WRITE MULTIPLE SECTOR command.

<table>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8"># OF SECTORS</td></tr>
</table>

The value contained in the register is decremented after each sector is transferred to/from the sector buffer. A zero represents a 256 sector transfer, a one a 1 sector transfer, etc. This register is ignored when single sector commands are specified in the Command register.

4.4 Sector Number

This register holds the sector number of the desired sector:

<table>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">SECTOR NUMBER</td></tr>
</table>

For a multiple sector command it specifies the first sector to be transferred. It is decremented after each sector is transferred to/from the sector buffer. The SECTOR NUMBER register may contain any value from 0 to 255. The ID Not Found bit will be set if the desired sector cannot be located on the track.

The SECTOR NUMBER register is also used to program the Gap 1 and Gap 3 lengths to be used when formatting a disk. See the WRITE FORMAT command description for further explanation.

4.5 Cylinder Number Low Register

This register holds the lower byte of the desired cylinder number:

<table>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">LS BYTE OF CYLINDER NUMBER</td></tr>
</table>

It is used in conjunction with the CYLINDER NUMBER HIGH register to specify a range of 0 to 2048 tracks.

4.6 Cylinder Number High Register

This register holds the three most significant bits of the desired cylinder number:

7	6	5	4	3	2	1	0
x	x	x	x	x	(10)	(9)	(8)

x = ignored

The 82064 contains a pair of registers that store the actual position where the R/W head are located. The CYLINDER NUMBER HIGH and LOW registers are considered the cylinder destination registers for seeks and other commands. The 82064 compares its internal registers to the destination registers and issues the number of steps in the right direction to make both sets of registers equal. After a command is executed, the internal cylinder position registers' contents are equal to the cylinder high/low registers. If a drive number change is detected on a new command, the 82064 automatically reads an ID field to update its internal cylinder position registers. This affects all commands except a RESTORE.

When a RESTORE command is executed, the internal head location registers are reset to zero while DIR and STEP move the heads to track zero.

4.7 Sector/Drive/Head (SDH) Register

The SDH register contains the desired sector size, drive number, and head number parameters. The format is shown below.

<table>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>EXT</td><td colspan="2">SECT SIZE</td><td colspan="2">DRIVE</td><td colspan="3">HEAD #</td></tr>
</table>

Both head number and sector size are compared against the disk's ID field. Head select and drive select lines are not available as outputs from the 82064 and must be generated externally.

Bit 7, the extension bit (EXT), is used to select between the CRC or ECC mode. When bit 7 = 1, the ECC mode is selected for the data field. When bit 7 = 0, the CRC mode is selected. The CRC is checked on the ID field regardless of the state of EXT. The SDH byte written into the ID field is different than the SDH Register contents. The recorded SDH byte does not have the drive number (DRIVE) written but does have the BAD BLOCK mark written.

Note that use of the extension bit requires the gap lengths to be modified as described in the WRITE FORMAT command description.

4.8 Status Register

The status register is a read-only register which informs the host of certain events. This register is a flow-through latch until the microprocessor reads it at which point the drive status lines are latched. The INTRQ line will be reset when this register is read. The format is:

7	6	5	4	3	2	1	0
BUSY	READY	WF	SC	DRQ	DWC	CIP	ERROR

Bit 7 - Busy

This bit is asserted when a command is written into the COMMAND register and, except for the READ command, is deasserted at the end of the command. When executing a READ command, Busy will be deasserted when the sector buffer is full. Commands should not be loaded into the COMMAND register when Busy is set. When the Busy bit is set, no other bits in the STATUS or ERROR registers are valid.

During other non-data transfer commands, Busy should be ignored as it will go active for short periods.

Bit 6 - Ready

This bit reflects the state of the DRDY (Pin 28) line at the time the microprocessor reads the status register. Transitions on the DRDY line will abort a command and set the aborted command bit in the error register.

Bit 5 - Write Fault

This bit reflects the state of the WR FAULT (Pin 30) line. Transitions on this line will abort a command and set the aborted command bit in the error register.

Bit 4 - Seek Complete

This bit reflects the state of the SC (Pin 32) line. Commands which initiate a seek will pause until Seek Complete is set. This bit is latched after an aborted command error.

Bit 3 - Data Request

The Data request bit (DRQ) reflects the state of the BDRQ (Pin 36) line. It is set when the sector buffer should be loaded with data or read by the host processor, depending upon the command. The DRQ bit and the BDRQ line remain high until BRDY indicates that the sector buffer has been filled or emptied, depending upon the command. BRDQ can be used for DMA.

Bit 2 - Data Was Corrected (DWC)

When set, this bit indicates that an ECC error has been detected during a read operation, and that the data in the sector buffer has been corrected. This provides the user with an indication that there may be a marginal condition within the drive before the errors become uncorrectable. This bit is forced to zero when not in the ECC mode.

Bit 1 - Command in Progress

When this bit is set, a command is being executed and a new command should not be loaded until it is cleared. Although a command may be executing, the sector buffer is still available for access by the host processor. When the 82064 is no longer busy (bit 7 = 0) the status register can be read. If CIP is set, only the status register can be read regardless of which register is selected.

Bit 0 - Error

This bit is a logical OR of the contents of the error register. Any bit being set in the error register sets this bit. The host must read the ERROR register to determine what type of error occurred. This bit is cleared when a new command is loaded.

4.9 Command Register

This write-only register is loaded with the desired command:

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">COMMAND</td></tr>
</table>

The 82064 begins to execute immediately upon loading any value into this register. This register should not be written while the Busy or Command in Progress bits are set in the STATUS register. The INTRQ line (Pin 3) if set, will be cleared by a write to the COMMAND register.

Instruction Set

The 82064 WDC instruction set contains six commands. Prior to loading the command register, the host processor must first set up the Task Register File with the information needed for the command. Except for the COMMAND register, the registers may be loaded in any order. If a command is in progress, a subsequent write to the COMMAND register will be ignored. A command is finished when the command in progress (CIP) bit in the STATUS register is cleared. See the Command Section for an explanation of each command.

COMMAND	7	6	5	4	3	2	1	
RESTORE	0	0	0	1	R3	R2	R1	R0
SEEK	0	1	1	1	R3	R2	R1	R0
READ SECTOR	0	0	1	0	I	M	0	T
WRITE SECTOR	0	0	1	1	0	M	0	T
SCAN ID	0	1	0	0	0	0	0	T
WRITE FORMAT	0	1	0	1	0	0	0	0
COMPUTE CORRECTION	0	0	0	0	1	0	0	0
SET PARAMETER	0	0	0	0	0	0	0	S
R 3−0 = Rate Field								
For 5 MHz WR Clock: 0000 — ≈35 μs 0001 — 0.5 ms 0010 — 1.0 ms 0011 — 1.5 ms 0100 — 2.0 ms 0101 — 2.5 ms 0110 — 3.0 ms 0111 — 3.5 ms 1000 — 4.0 ms 1001 — 4.5 ms 1010 — 5.0 ms 1011 — 5.5 ms 1100 — 6.0 ms 1101 — 6.5 ms 1110 — 3.2 μs 1111 — 16 μs								

COMMAND	7	6	5	4	3	2	1
T =	Retry Enable						
T = 0 T = 1	Enable Retries Disable Retries						
M =	Multiple Sector Flag						
M = 0 M = 1	Transfer 1 Sector Transfer Multiple Sectors						
I =	Interrupt Enable						
I = 0 I = 1	Interrupt at BDRQ time Interrupt at end of command						
S =	Error Correction Span						
S = 0 S = 1	5-bit Span 11-bit Span						

5.0 PROGRAMMING THE 82064

This section consists of two parts. The first part gives an explanation of each command, a flowchart showing the 82064's sequence of events, and the commands' sequence of events as seen by the host microprocessor. The second section shows flowcharts of general software routines and their PLM equivalent, for both polled and interrupt driven software.

The designer must remember that the 82064 expects a full sector buffer that can be isolated from the host during data transfers between the 82064 and the disk. Since the 82064 assumes a full sector buffer is available, it does not check for data overrun or underrun error conditions. If such a condition occurs, corruption of data will happen and the host will have no indication of an error. The design must guarantee against over-run and under-run conditions when not using the sector buffer approach.

5.1 Commands

A command is placed into the command register only after the Task Registers have been written with proper values. The Task Registers may be loaded in any order. A command, once started, can only be terminated by a hardware reset to the 82064. This may corrupt data on the disk by removing necessary control signals out of sequence.

The general sequence of a command is as follows:

- The host loads the Task Registers
- The host loads the Command Register
- The 82064 locates the correct cylinder
- Data transfer takes place
- The 82064 issues an interrupt

Restore Command – 0 0 0 1 R3 R2 R1 R0

The Restore command is used to position the heads to track 0. This command is usually issued to the 82064 on power-up to initialize internal registers. The user specified rate field (R3–R0) is stored internally for FUTURE use in commands with implied seeks.

The step rate value is not used with this command. The actual stepping rate used is dependent upon the handshake delay between the 82064 issuing a step pulse and the drive returning a seek complete for each track. After each step pulse is issued, the 82064 waits for a rising edge on the Seek Complete (SC) line before issuing the next pulse. If 8 index pulses are received without a rising edge on SC, the 82064 will switch to sampling the level of the SC line. If after 2048 step pulses the Track 00 signal has not gone active, the 82064 will terminate

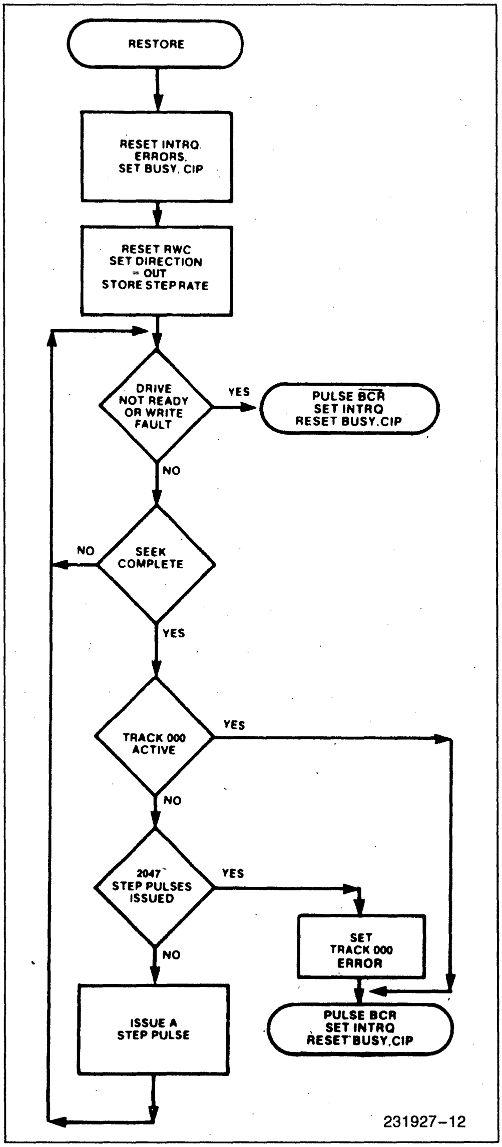


Figure 13. Restore Command Flow

the command, assert INTRQ and set the TRACK 000 bit in the Error Register. The command will terminate if WR Fault goes active or DRDY goes inactive at any time. Figure 13 is a flow chart of the command.

Seek Command – 0 1 1 1 R3 R2 R1 R0

The Seek command positions the heads to the cylinder specified in the Task Registers.,The direction and number of step pulses issued is calculated by comparing the cylinder high/low registers to an internal "present position" cylinder register. The present position register is updated after all step pulses are issued and the command is terminated.

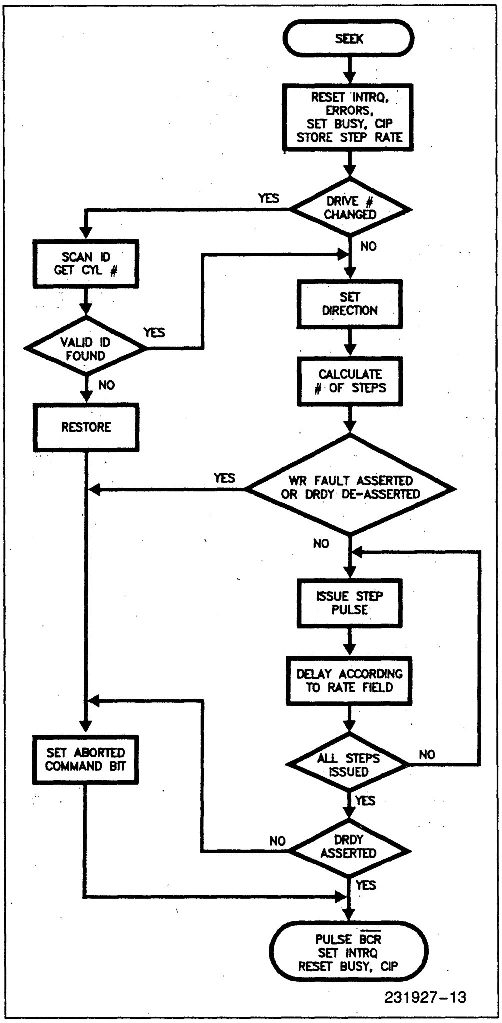


Figure 14. Seek Command Flow

The actual stepping rate is taken from the rate field bits (R3–R0) and stored for future use. The command terminates at once if WR FAULT goes active or DRDY goes inactive at any time. Figure 14 is a flowchart of the command.

Since the data transfer commands feature implied seeks, this command is of use mainly to those using multiple drives and software that can take advantage of overlapped seeks.

Scan ID Command - 0100000T

The Scan ID command is used by both the 82064 and the host to update the SDH, the Sector Number, Cylinder and internal present position registers. Once the command is issued, the Seek Complete line is sampled until valid. The first ID field found, as indicated by the address mark, is loaded into the previously mentioned registers. The Bad Block bit will be set if detected, and the command will terminate. ID CRC errors will start the search sequence over for a maximum of 10 index pulses, but the registers will be loaded with whatever data the 82064 had perceived as ID information. Improper states on WR Fault on DRDY will terminate the command. Figure 15 is the flow chart of the command.

The main use for this command is to determine where the heads are currently located and what size the sectors are (i.e. 256, 512 etc.). Without this command, it would be necessary to recall the heads to track zero and then step out to the desired cylinder each time a drive was changed. Specifying the wrong sector size would yield an ID not found error. This command enables the system to read the disk drive to determine what size sectors were recorded.

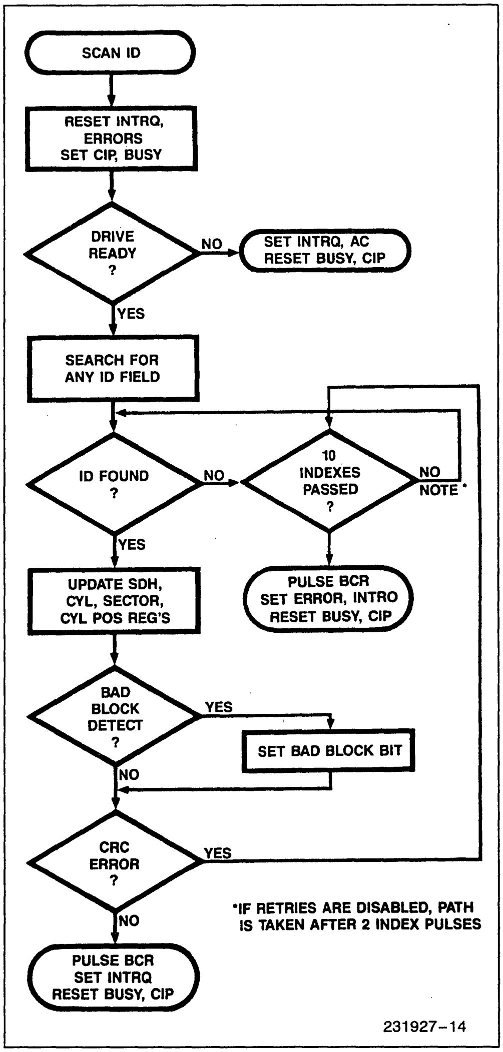

SCAN ID
RESET INTRQ, ERRORS SET CIP, BUSY
DRIVE READY ?
NO
SET INTRQ, AC RESET BUSY, CIP
YES
SEARCH FOR ANY ID FIELD
ID FOUND ?
NO
10 INDEXES PASSED ?
NO
NOTE*
YES
UPDATE SDH, CYL, SECTOR, CYL POS REG'S
PULSE BCR SET ERROR, INTRO RESET BUSY, CIP
BAD BLOCK DETECT ?
YES
SET BAD BLOCK BIT
NO
CRC ERROR ?
YES
NO
PULSE BCR SET INTRQ RESET BUSY, CIP
*IF RETRIES ARE DISABLED, PATH IS TAKEN AFTER 2 INDEX PULSES
231927-14

Figure 15. Scan ID Command Flow

Read Sector Command - 00101M0T

The READ SECTOR command is used to transfer one or more sectors of data from the disk to the sector buffer. Upon receipt of the READ SECTOR command, the 82064 checks the CYLINDER NUMBER LOW/HIGH register pair against an internal cylinder position register to see if they are equal. If not, the direction and number of steps are calculated and a seek takes place. If an implied seek is performed, the 82064 will search until a rising edge of SC is received. The WR FAULT and DRDY lines are monitored throughout the command.

Once the Seek Complete (SC) line is high (with or without an implied seek having occurred), the search for an ID field begins. If T = 0 (retries enabled), the 82064 must find an ID with the correct cylinder number, head, sector size, and CRC within 10 revolutions, or a Scan ID and re-Seek will be performed. The search for the proper ID will again be tried for up to 10 revolutions. If the correct sector is still not found, the appropriate error bits will be set and the command terminated. Data CRC errors will also be retried for up to 10 revolutions (if T = 0).

If T = 1 (retries disabled), the ID search must find the correct sector within 2 revolutions or the appropriate error bits will be set and the command terminated.

Both the READ SECTOR and WRITE SECTOR commands feature a "simulated completion" to ease programming. DRQ/BDRQ will be generated upon detecting an error condition. This allows the same program flow for successful or unsuccessful completion of a command.

When the data address mark is found, the 82064 is ready to tranfer data to the sector buffer. After the data has been transferred, the I bit is checked. If I = 0, INTRQ is made active coincident with BDRQ, indicating that a transfer of data from the buffer to the host processor is required. If I = 1, INTRQ will occur at the end of the command, i.e. after the buffer is unloaded by the host.

In summary then, READ SECTOR operation is as follows:

When M = 0 (READ SECTOR)

(1) Host: Sets up parameters; issues READ SECTOR command.
(2) 82064: Strobes $\overline{\text{BCR}}$.
(3) 82064: Finds sector specified; asserts $\overline{\text{BCR}}$ and $\overline{\text{BCS}}$; transfers data to buffer.
(4) 82064: Sets $\overline{\text{BCR}}$ = 1, $\overline{\text{BCS}}$ = 0.
(5) 82064: Sets BDRQ = 1; DRQ = 1.
(6) 82064: If I bit = 1 go to (9).
(7) Host: Reads contents of sector buffer.
(8) 82064: Waits for BRDY, then sets INTRQ = 1: END.
(9) 82064: Sets INTRQ = 1.
(10) Host Reads out contents of buffer; END.

When M = 1 (READ MULTIPLE SECTOR)

(1) Host: Sets up parameters; issues READ SECTOR command.
(2) 82064: Assert $\overline{\text{BCR}}$.
(3) 82064: Finds sector specified; asserts $\overline{\text{BCR}}$ and $\overline{\text{BCS}}$; transfers data to buffer.
(4) 82064: Strobes $\overline{\text{BCR}}$; sets $\overline{\text{BCS}}$ = 0.
(5) 82064: Sets BDRQ = 1; DRQ = 1.
(6) Host: Reads out contents of buffer.
(7) 82064: Waits for BRDY; Decrements SECTOR COUNT; increments SECTOR NUMBER.
(8) 82064: When BRDY = 1, if Sector Count = 0 then go to (10).
(9) 82064: Go to (2).
(10) 82064: Set INTRQ = 1; End.

A flowchart of the READ SECTOR command is shown in Figures 16A and 16B.

The M bit is set for multiple sector transfers. When M = 0, one sector is transferred and the SECTOR COUNT register is ignored. When M = 1, multiple sectors are transferred. After each sector is transferred, the 82064 decrements the SECTOR COUNT register and increments the SECTOR NUMBER register. The next logical sector will be transferred regardless of any interleave. Sectors are numbered at format time.

Multiple sector transfers continue until the SECTOR COUNT register equals zero, or the BRDY line goes active (low to high). If the SECTOR COUNT register is non-zero (indicating more sectors are to be transferred but the buffer is full), BDRQ will be made active and the host must unload the buffer. After this occurs, the buffer will again be free to accept the remaining sectors from the 82064. This scheme enables the user to transfer more sectors than the buffer memory has capacity for.

Write Sector Command – 01110M0T

The WRITE SECTOR command is used to write one or more sectors of data to the disk from the sector buffer. Upon receipt of a WRITE SECTOR command the 82064 checks the CYLINDER NUMBER LOW/HIGH register pair against the internal cylinder position register to see if they are equal. If not, the direction and number of steps calculation is performed and a seek takes place. The WR FAULT and DRDY lines are checked throughout the command.

When the Seek Complete (SC) line is found to be true (with or without an implied seek having occurred), the BDRQ signal is made active and the host proceeds to load the buffer. Once BRDY goes high, the ID field with the specified cylinder number, head, and sector size is searched for. Once found, WR GATE is made active and the data is written to the disk. If retries are enabled (T = 0), and if the ID field cannot be found within 10 revolutions, a Scan ID and re-Seek are performed. If the correct ID field is not found within 10 additional revolutions, the ID Not Found error bit is set and the command is terminated. If retries are disabled, (T = 1) and if the ID field cannot be found within 2 revolutions, the ID Not Found error bit is set and the command is terminated.

During a WRITE MULTIPLE SECTOR command (M = 1), the SECTOR NUMBER register is decremented and the SECTOR COUNT register is incremented after the transfer to the disk takes place. During multiple sector transfers if BRDY is asserted after the first sector is transferred from the buffer, the 82064 will transfer the next sector before issuing BDRQ. The 82064 will set BDRQ and wait for the host processor to place more data in the buffer.

In summary then, the WRITE SECTOR operation is as follows:

When M = 0, 1 (WRITE SECTOR)

(1) Host: Sets up parameters; issues WRITE SECTOR command.
(2) 82064: Sets BDRQ = 1, DRQ = 1.
(3) Host: Loads sector buffer with data.
(4) 82064: Waits for BRDY = 0 to 1.
(5) 82064: Finds specified ID field; writes sector to disk.
(6) 82064: If M = 0, then set INTRQ = 1; END.
(7) 82064: Increment SECTOR NUMBER register; decrement SECTOR COUNT register.
(8) 82064: If SECTOR = 0, then set INTRQ = 1; END.
(9) 82064: Go to (2).

A flowchart of the WRITE SECTOR command is shown in Figure 17.

Write Format Command 01010000

The WRITE FORMAT command is used to format one track using the Task Register File and the sector buffer. During execution of this command, the sector buffer is used for additional parameter information instead of sector data. Shown in Figure 18 is the contents of the sector buffer for a 32 sector/track format with an interleave factor of two. Each sector requires a two byte sequence. The first byte designates whether a bad block mark is to be recorded in the sector's ID field. A 00 Hex is normal; an 80H indicates a bad block mark for the sector. In the example of Figure 18, sector 04 will get a back block mark recorded. Any attempt to access sector 4 in the future will terminate the command.

The second byte indicates the logical sector number to be recorded. This allows sectors to be recorded with any interleave factor desired. The remaining memory in the sector buffer may contain any value. Its only purpose is to generate a BRDY to tell the 82064 to begin formatting the track. An implied seek is in effect on this command. As for other commands, if the drive number has been changed an ID field will be scanned for cylinder position information before the implied seek is performed. If no ID field can be read (because the track had been erased or because an incomplete format had been used), an ID Not Found error will result and the WRITE FORMAT command will be aborted. This can be avoided by issuing a RESTORE command before formatting.

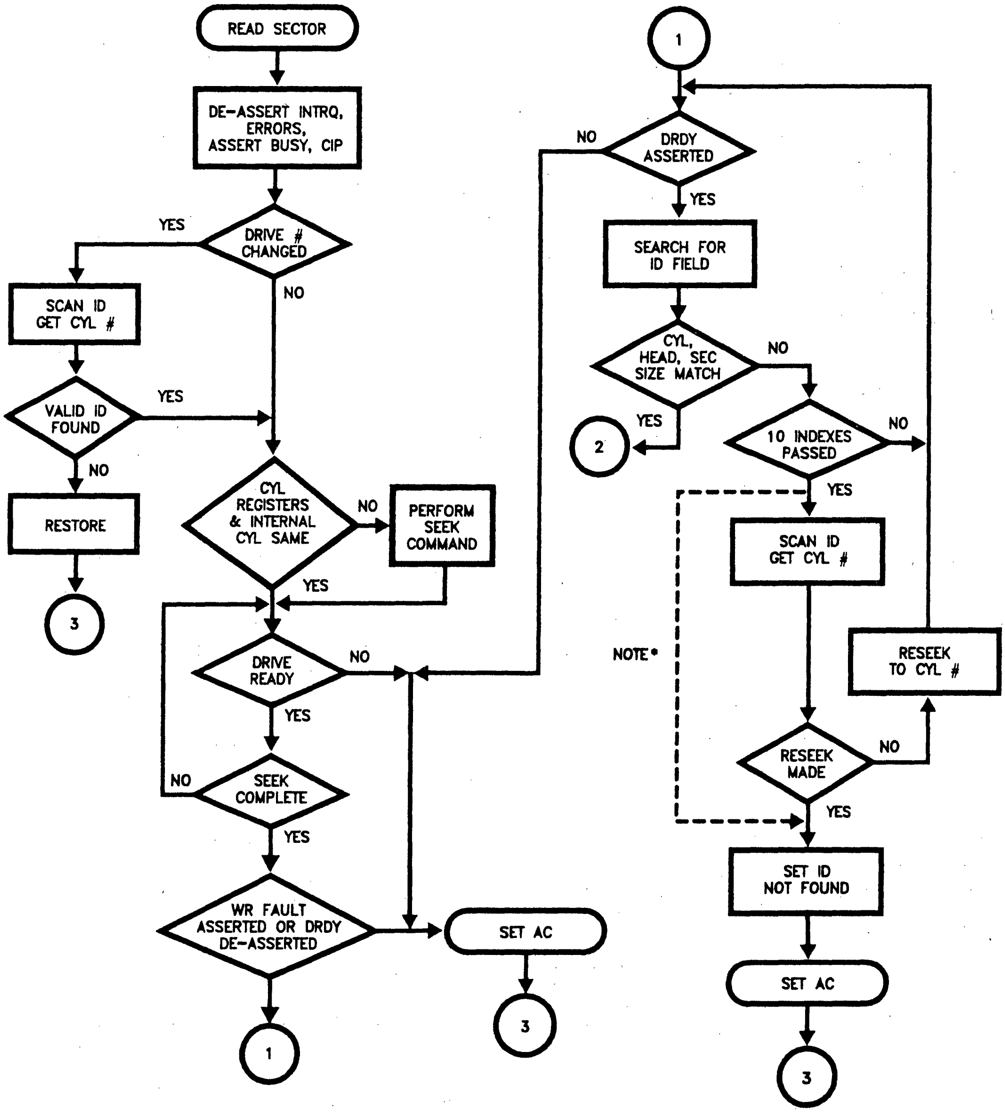
The SECTOR COUNT register is used to hold the total number of sectors to be formatted (01H = 1 sector; 00H = 256 sectors), while the SECTOR NUMBER register holds the number of bytes (minus three) to be used for Gap 1 and Gap 3. For instance, if the SECTOR COUNT register value is 02H and the SECTOR NUMBER register value is 00H, then 2 sectors are written on a track and 3 bytes of 4EH are written for Gap 1 and Gap 3. The data fields are filled with FFH and the CRC is automatically generated and appended. All gaps are filled with 4EH. After the last sector is written, the track is filled with 4EH until the index pulse terminates the write. The Gap 3 value is determined by the drive motor speed variation, data sector length, and the interleave factor. The interleave factor is only important when 1:1 (no) interleave is used. The formula for determining the minimum Gap 3 length value is:

$$\text{Gap } 3 = (2 * M * S) + K + E$$

M = motor speed variation (e.g., 0.03 for ±3%)
S = sector length in bytes
K = 25 for interleave factor of 1
K = 0 for any other interleave factor
E = 7 if the sector is to be extended

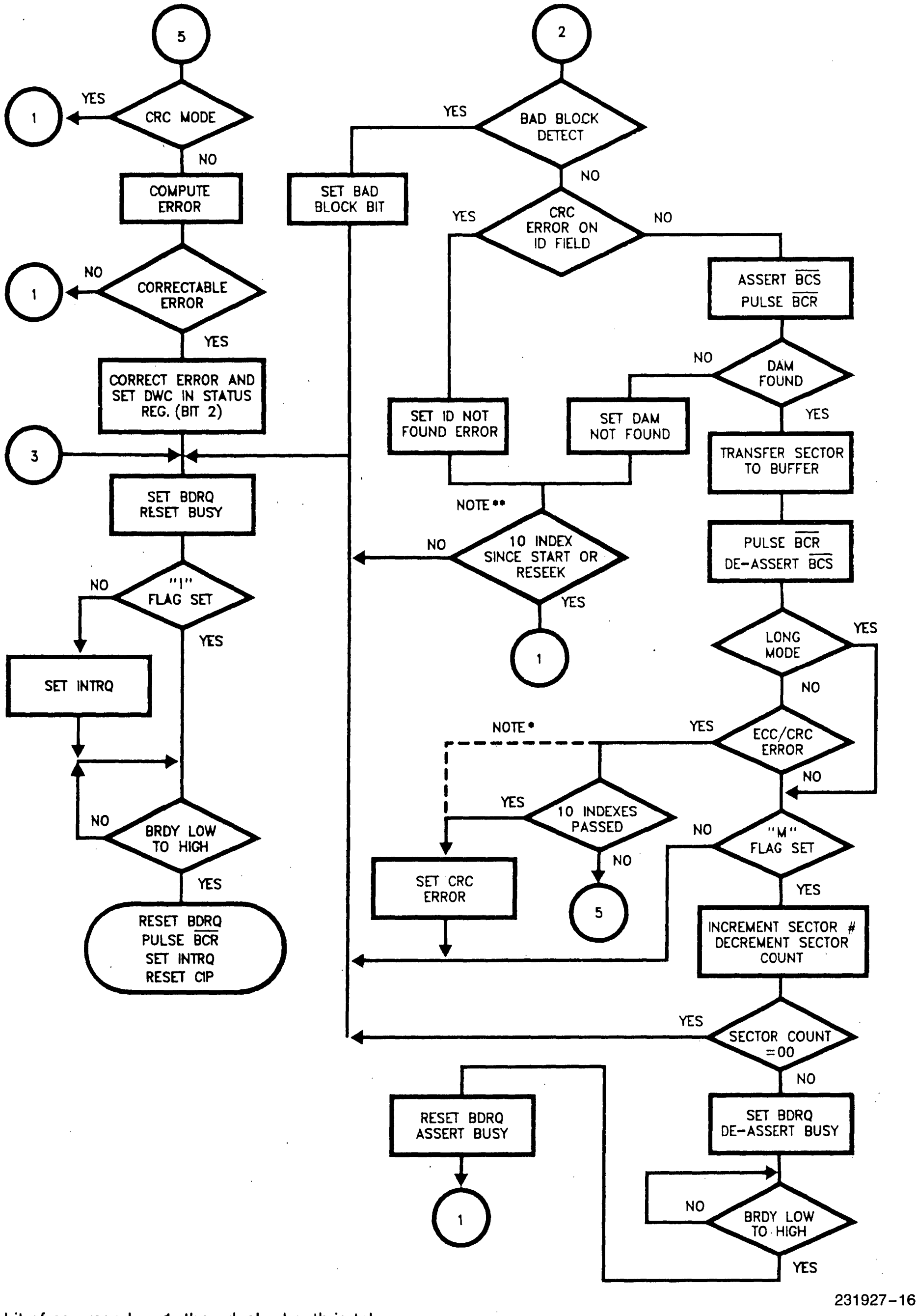
As with all commands, a WR FAULT or drive not ready condition, will terminate execution of the WRITE FORMAT command. Figure 19 shows the format that the 82064 will write on the disk. The extend bit in the SDH register must not be set during the Format command.

A flowchart of the WRITE FORMAT command is shown in Figure 20.



NOTE:
* If T = 1, then "dashed" path is taken after 2 index pulses.

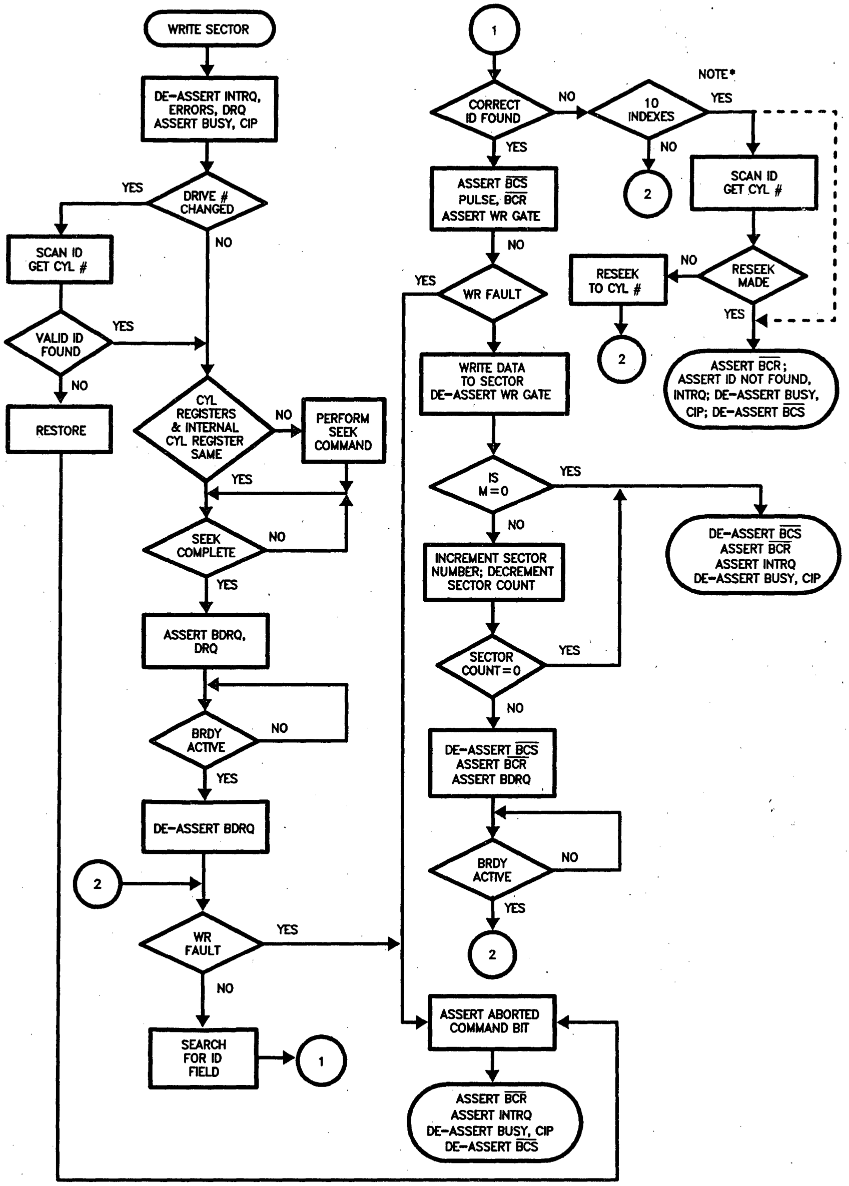
Figure 16A. Read Sector Command Flow



231927-16

* If T bit of command = 1, then dashed path is taken.
** If T bit of command = 1, then test is for 2 index pulses.

Figure 16B. Read Sector Command Flow (Continued)



WRITE SECTOR
DE-ASSERT INTRQ, ERRORS, DRQ ASSERT BUSY, CIP
DRIVE # CHANGED
YES
NO
SCAN ID GET CYL #
VALID ID FOUND
YES
NO
RESTORE
CYL REGISTERS & INTERNAL CYL REGISTER SAME
NO
PERFORM SEEK COMMAND
YES
SEEK COMPLETE
NO
YES
ASSERT BDRQ, DRQ
BRDY ACTIVE
NO
YES
DE-ASSERT BDRQ
2
WR FAULT
YES
NO
SEARCH FOR ID FIELD
1
1
CORRECT ID FOUND
NO
10 INDEXES
NOTE*
YES
NO
2
YES
ASSERT BCS PULSE, BCR ASSERT WR GATE
NO
WR FAULT
YES
SCAN ID GET CYL #
RESEEK MADE
NO
RESEEK TO CYL #
2
YES
ASSERT BCR; ASSERT ID NOT FOUND, INTRQ; DE-ASSERT BUSY, CIP; DE-ASSERT BCS
WRITE DATA TO SECTOR DE-ASSERT WR GATE
IS M=0
YES
NO
DE-ASSERT BCS ASSERT BCR ASSERT INTRQ DE-ASSERT BUSY, CIP
INCREMENT SECTOR NUMBER; DECREMENT SECTOR COUNT
SECTOR COUNT=0
YES
NO
DE-ASSERT BCS ASSERT BCR ASSERT BDRQ
BRDY ACTIVE
NO
YES
2
ASSERT ABORTED COMMAND BIT
ASSERT BCR ASSERT INTRQ DE-ASSERT BUSY, CIP DE-ASSERT BCS
231927-17

NOTE:
*If retries are disabled, the "dashed" path is taken after 2 index pulses.

Figure 17. Write Sector Command Flow

```
00  00  00  10  00  01  00  11  00  02  00  12  00  03  00  13
80  04  00  14  00  05  00  15  00  06  00  15  00  07  00  17
00  08  00  18  00  09  00  19  00  0A  00  19  00  0B  00  1B
00  0C  00  1C  00  0D  00  1D  00  0E  00  1E  00  0F  00  1F
FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF
AA  AA  AA  AA  AA  AA  AA  AA  AA  AA  AA  AA  AA  AA  AA  AA
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
```

Figure 18. Sector Buffer Contents For Format

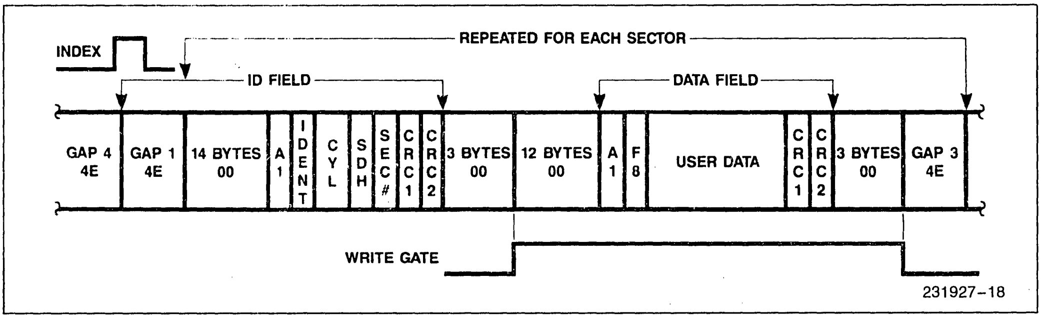


Figure 19. 82064 Sector Format

5.2 Software Section: General Programming

This section describes the software needed to communicate with the 82064 in order to store and retrieve data. This chapter describes the software in a general manner and Appendix B contains the actual implementation used to exercise the 82064 SBX board.

Polled Mode

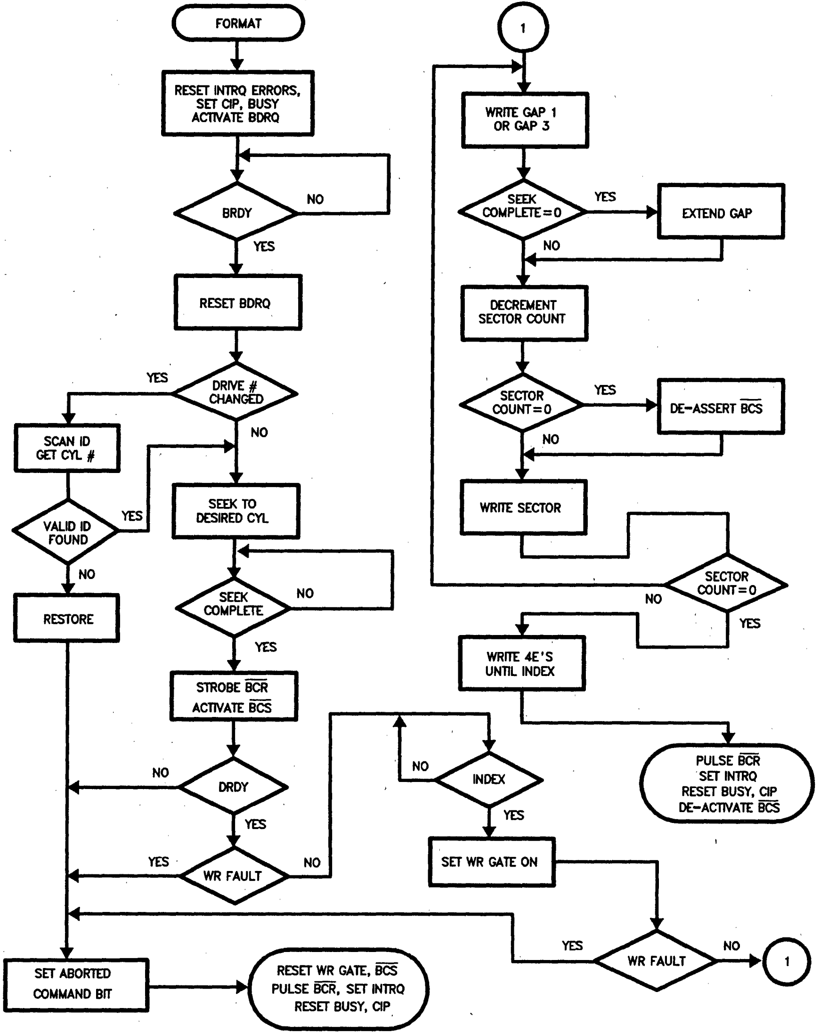
As discussed in the Polled Interface Section, the 82064 does not directly support polled operation for data transfers without the addition of hardware. This section is based upon the polled interface as described in the Polled Interface Section.

The six 82064 commands can be divided into two groups, those with data transfers and those without. The commands that do not use the sector buffer are: Restore, Seek and Scan ID. The functions of each command are explained in the Commands Section. Figure 21 is a flowchart of a polled operation and a PLM example.

The last status that was read will contain any error conditions that might have occurred during the command.

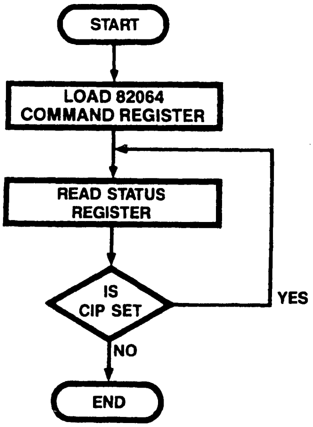
For commands that do make use of the sector buffer, the size of the sector buffer will affect the software. If the sector buffer is equal in size to one sector, then a carry out of an address counter (for the sector buffer) as the buffer is being filled will indicate to the 82064 that the command should continue. If the sector buffer size is equal to two or more disk sectors, and only one sector is being transferred, then the carry out signal would not go active, and the 82064 will be forever waiting for BRDY. In this case an I/O port would have to be used to generate this signal for the 82064 so that command execution can finish. Figure 22 is a flowchart of the READ SECTOR command, and its PLM representation. The WRITE SECTOR and FORMAT TRACK commands are equivalent in terms of software interfacing. Their flowcharts and their PLM equivalents are shown in Figure 23.

Once the command register is written the 82064 requests a data transfer before locating the proper track. Once the buffer is filled and BRDY is asserted, the 82064 will locate the target track and sector. If the ID is not located before the selected number of retries have occurred, the 82064 will terminate the command. The data transferred to the sector buffer will not have been used. Once the command has finished (i.e., CIP = 0), the status and error registers will inform the host of an error.



231927-19

Figure 20. Write Format Command Flow

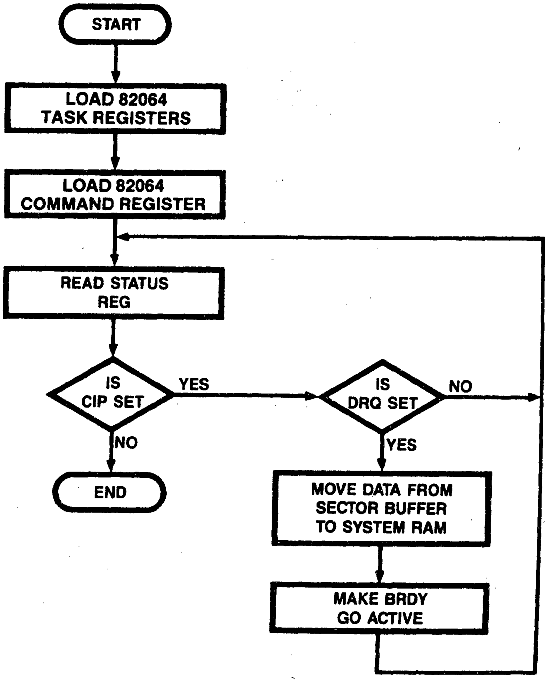


231927-20

```
Disk$Operation: Procedure;
   Call Write$82064$Task$Reg's; /* Write Task Registers */
   Output (Command$Reg) = Command;
   Status = Input (Status$Reg); /* Read Status Reg */
   Do while Status and CIP = CIP; /* Wait until command finishes */
      Status = Input (Status$Reg);
   End;
End Disk$Operation;
```

Figure 21. Polling Status

READ SECTOR COMMAND

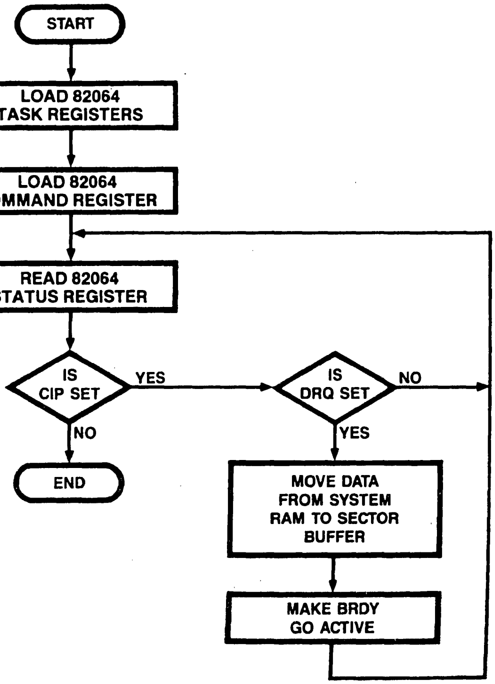


231927-21

```
Disk$Operation: Procedure;
    Call Write$82064$Task$Regs;
    Output (Command $ Reg) = Command;
    Status = Input (Status$Reg);
    Do while Status and CIP = CIP;
        If Status and DRQ = DRQ then Do;
          Call Read$Data$From$Buffer;
          Output (BRDY$PORT) = 01;
        End;
        Status = Input (Status$Port)
    End;
End Disk$Operation;
```

Figure 22. Polling For Read Data

WRITE, FORMAT COMMANDS


START
LOAD 82064 TASK REGISTERS
LOAD 82064 COMMAND REGISTER
READ 82064 STATUS REGISTER
IS CIP SET
YES
NO
END
IS DRQ SET
NO
YES
MOVE DATA FROM SYSTEM RAM TO SECTOR BUFFER
MAKE BRDY GO ACTIVE

231927-22

```
Disk$Operation: Procedure;
  Call Write$82064$Task$Regs;
  Output (Command$Reg) = Command;
  Status = Input (Status$Reg);
  Do while status and CIP = CIP;
    If status and DRQ = DRW then do;
      Call Write$Data$to$Buffer;
      Output (BRDY$Port) = 01; /* Make BRDY go active */
    End;
    Status = Input (Status$Reg)
  End;
End Disk$Operation;
```

Figure 23. Polling For Write Data

```
Disk$Operation: Procedure;
   Call Write$82064$Task$Regs; /* Write registers */
   Output (Command$Reg) = Command; /* Start command */
   Status = Input (Status$Reg); /* Read status */
   Do while status and CIP = CIP; /* Is a command in progress */
     If status and DRQ = DRQ then do; /* Data transfer? = yes */
       If command = Read$Sector then
        Call Read$Data$From$Buffer; /* Remove data */
       Else Call Write$Data$to$Buffer; /* Send data */
       Output (BRDY$PORT) = 01; /* Toggle BRDY 0 to 1 */
     End;
End Disk$Operation;
```

Figure 24. Complete Polled Flow

```
Start$Disk$Operation: Procedure;
    Call Write$82064$Task$Reg's;
    Output (Command $ Reg) = Command;
End Start$Disk$Operation;
```

Figure 25. Interrupt Mode; Starting a Disk Transfer

Figure 24 is the PLM routine that allows for all six of the commands. It differs from the READ and WRITE routines in that the direction that data is to be transferred is determined by the command.

Figure 24 also works for multiple sector transfers. However, the BRDY signal must be generated in hardware (the carry-out of an address counter).

Interrupt Mode

Interrupt driven software is chosen when the microprocessor must execute other tasks and cannot sit waiting for the disk to reposition its heads, as in a polled environment. The delay in repositioning heads can be anything from a couple of milliseconds to a second or more.

The 82064's interrupt (INTRQ) pin goes active to indicate that the command has finished. The READ SECTOR command provides the programmable choice of having the interrupt occur at the end of the data transfer or the normal end of the command. The reason for this option is that when the 82064 signals that a data transfer is required (via BDRQ, DRQ) the disk has been read and the data has been placed in the buffer. The host would remove the data and issue BRDY. The 82064 would then issue an interrupt indicating that the command has finished. The interrupt procedure would only have to read the status register. If the interrupt is issued at BDRQ the host would remove the buffer data and generate BRDY. At this point the status and error registers contain valid information. Generating an interrupt at BDRQ time may save some systems some software overhead.

The WRITE SECTOR and FORMAT commands do not have this option because the sector buffer is filled before the track and sector are located. Hence, there can be significant delays between asking for data and the command terminating.

In an interrupt driven environment, the 82064 can interface to a DMA controller for data transfers between the sector buffer and the host's RAM. If a DMA controller is not available an interrupt must be generated via the BDRQ line. However, BDRQ can stay active for long periods of time (until BRDY is generated). The interrupt sensing logic must take this into account to avoid being retriggered constantly. Intel's 8259A Interrupt Controller 8259A provides that capability. It should be programmed for edge triggered interrupts or the end of interrupt byte must not be issued until BDRQ is removed to prevent retriggering.

Figure 25 is a PLM example of starting a disk operation in an interrupt driven environment. The command starts, and some indefinite amount of time later an interrupt would be generated, indicating service is required.

```
End$of$Transfer: Procedure Interrupt;
    Status = Input (Status$Register);
    Output (8259A PIC) = End$of$Interrupt;
End End$of$Transfer;
```

Figure 26. Checking Status via Interrupt

```
Service$Disk$Controller: Procedure Interrupt;
    Status = Input (Status$Port);
    If Status and DRQ = DRQ then
        Call Transfer$Data$To/From$Buffer;/* Enable DMAC */
    Output (8259A PIC) = End$of$Interrupt;
End Service$Disk$Controller;
```

Figure 27. Complete Interrupt Procedure

If a DMA controller is used, it would have to be programmed and initialized before the command is issued to the 82064. Recall that once a data transfer between the microprocessor and 82064 has finished, BRDY must be set high. As long as BRDY is generated from hardware, no microprocessor intervention is needed. If BRDY is generated by an I/O port the microprocessor will have to perform this function (this will be the case with any system that has a sector buffer larger than one sector). (One option could be to generate an interrupt from the terminal count pin of the DMA controller. The microprocessor would then issue a BRDY.) Data transfers between host RAM and the sector buffer would be handled without microprocessor intervention. The interrupt would then signal that the command has finished as shown in Figure 26. The only operation the host processor would perform is to check the status register of the 82064 for any error conditions.

If BDRQ is used to generate an interrupt in addition to the normal interrupt, then the routines shown in Figure 27 will check the status register to see if a data transfer should be executed or if the command is finished. If DRQ is not set, the command has finished and any error conditions would be in the status register.

Another possibility would be to have separate interrupt routines for the two possible sources of interrupts (INTRQ, BRDQ). There would then be no need to test the status to see which interrupt had occurred.

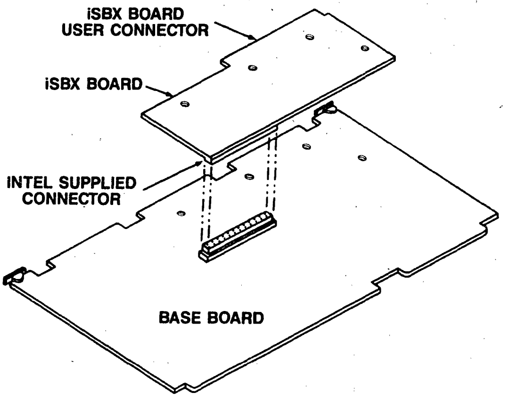
6.0 APPLICATION EXAMPLE

This section shows an application using the 82064 interfaced to the SBX bus. A quick overview of the SBX bus is provided (pin descriptions, general wave forms) as a background for the application. Designing the 82064 onto an SBX Multimodule board was chosen to highlight the size and complexity differences between earlier TTL, MSI, LSI-based disk controller boards and what is possible using the 82064. Both the hardware and software sections will be applicable to most other designs using the 82064. This design example is called SBX82064 and does not represent a real product offered by Intel Corporation. Appendix C contains the schematic of the SBX board.

The advantage of the SBX Multimodule is that it permits the system to be tailored for specific needs with a minimum of effort. The advantage of an SBX based disk controller is that a current system can make use of the capacity, reliability and speed of a hard disk with no (or minimal) hardware redesign.

6.1 iSBX Bus Multimodule Boards

The iSBX Multimodule boards are small, specialized, I/O mapped boards which plug onto base boards. The iSBX boards connect to the iSBX bus connector and convert the iSBX bus signals to a defined I/O interface.



231927–23

Figure 28. iSBX Multimodule Board Concept (Double Wide)

Base Boards

The base board decodes I/O addresses and generates the chip selects for the iSBX Multimodule boards. In 8-bit systems, the base board decodes all but the lower three addresses in generating the iSBX Multimodule board chip selects. In 16-bit systems, the base board decodes all but the lower order four addresses in generating the iSBX Multimodule board chip selects. Thus, a base board would normally reserve two blocks of 8 I/O ports for each iSBX socket it provides.

There are two classes of base boards, those with Direct Memory Access (DMA) support and those without. Base boards with DMA support are boards with DMA controllers on them. These boards, in conjunction with an iSBX Multimodule board (with DMA capability), can perform direct I/O to memory or memory to I/O operations.

iSBX Bus Interface

The iSBX bus interface can be grouped into six functional classes:

1. Control Lines
2. Address and Chip Select Lines
3. Data Lines
4. Interrupt Lines
5. Option Lines
6. Power Lines

Control Lines

The following signals are classified as control lines:

COMMANDS:
    $\overline{\text{IORD}}$ (I/O Read)
    $\overline{\text{IOWRT}}$ (I/O Write)
DMA:
    MDRQT (DMA Request)
    $\overline{\text{MDACK}}$ (DMA Acknowledge)
    TDMA (Terminate DMA)
INITIALIZE:
    RESET
CLOCK:
    MCLK (iSBX Multimodule Clock)
SYSTEM CONTROL:
    $\overline{\text{MWAIT}}$
    $\overline{\text{MPST}}$ (iSBX Multimodule Board Present)

Command Lines ($\overline{\text{IORD}}$, $\overline{\text{IOWRT}}$)

The command lines are active low signals which provide the communication link between the base board and the iSBX Multimodule board. An active command line, conditioned by chip select, indicates to the iSBX Multimodule board that the address lines are valid and the iSBX Multimodule board should perform the specified operation.

DMA Lines (MDRQT, $\overline{\text{MDACK}}$, TDMA)

The DMA lines are the communication link between the DMA controller device on the base board and the iSBX Multimodule board. MDRQT is an active high output signal from the iSBX Multimodule board to the base board's DMA device requesting a DMA cycle. $\overline{\text{MDACK}}$ is an active low input signal to the iSBX Multimodule board from the base board DMA device acknowledging that the requested DMA cycle has been granted. TDMA is an active high output signal from the iSBX Multimodule board to the base board. TDMA is used by the iSBX Multimodule board to terminate DMA activity. The use of the DMA lines is optional as not all base boards will provide DMA channels and not all iSBX Multimodule boards will be capable of supporting a DMA channel.

Initialize Lines (Reset)

This input line to the iSBX Multimodule board is generated by the base board to put the iSBX Multimodule board into a known internal state.

Clock Lines (MCLK)

This input to the iSBX Multimodule board is a timing signal. The 10 MHz (+0%, −10%) frequency can vary from base board to base board. This clock is asynchronous from all other iSBX bus signals.

System Control Lines ($\overline{\text{MWAIT}}$, $\overline{\text{MPST}}$)

These output signals from the iSBX Multimodule board control the state of the system.

An active $\overline{\text{MWAIT}}$ (Active Low) will put the CPU on the board into wait states providing additional time for the iSBX Multimodule board to perform the requested operation. $\overline{\text{MWAIT}}$ must be generated from address (address plus chip select) information only. If $\overline{\text{MWAIT}}$ is driven active due to a glitch on the CS line during address transitions, $\overline{\text{MWAIT}}$ must be driven inactive in less than 75 ns.

The iSBX Multimodule board present ($\overline{\text{MPST}}$) is an active low signal (tied to signal ground) that informs the base board I/O decode logic that an iSBX Multimodule board has been installed.

Address and Chip Select Lines

The address and chip select lines are made up of two groups of signals.

Address Lines:	MA0–MA2
Chip Select Lines:	$\overline{\text{MCS0}}$–$\overline{\text{MCS1}}$

The base board decodes I/O addresses and generates the chip selects for the iSBX Multimodule boards. The base board decodes all but the lower order three addresses in generating the iSBX Multimodule board chip selects.

Address Lines (MA0–MA2)

These positive true input lines to the iSBX Multimodule boards are generally the least three significant bits of the I/O address. In conjunction with the command and chip select lines, they establish the I/O port address being accessed. In 16-bit systems, MA0–MA2 may be connected to ADR1–ADR3 of the base board address lines.

Chip Select Lines (MCS0–MCS1/)

In an 8-bit system, these input lines to the iSBX Multimodule board are the result of the base board I/O decode logic. $\overline{\text{MCS}}$ is an active low signal which conditions the I/O command signals and thus enables communication with the iSBX Multimodule boards.

6.2 The SBX82064 Design Example

The SBX82064 Multimodule board will interface an ST506 compatible drive to any host board having an SBX connector. Two restrictions on the disk drive are that there is a maximum of 2048 cylinders and/or 8 heads. The SBX connector cannot supply the power-up current requirements of the drive. The drive must be connected directly to the power supply. The SBX82064 in Appendix C does not support DMA transfers. The version in Appendix D does support DMA transfers. Since this multimodule has a 2 kbyte sector buffer, the host microprocessor must generate a BRDY by accessing an I/O port during data transfers.

The software for communicating to the SBX board is intended to be interrupt driven. Polling for data transfers is not supported. Reading the status without an interrupt is not recommended. During the times the 82064 is accessing the sector buffer, the SBX82064 will isolate itself from the host. To support polling, a hardware generated busy pattern should be driven onto the host's data bus as is shown in the Polled Interface section. The sector buffer stores up to 2 kbytes of disk data, for multiple sector transfers. The SBX board only interfaces to one drive (for space reasons), but four drives could be used with the addition of a read data multiplexor (one IC) and the drive data cables.

Microprocessor Interface

Figure 29 is a block diagram of the SBX82064's microprocessor interface. The I/O port assignments are listed in Table 1. The functional blocks of the interface are:

Sector Buffer Isolation Logic

Wait State Logic

Sector Buffer

Sector/Drive/Head Register Logic

Table 6-1. I/O Port Assignments

Port Address	Read	Write
80H	Sector Buffer	Sector Buffer
82H	Error Reg	RWC Reg
84H	Sector Count	Sector Count
86H	Sector Number	Sector Number
88H	Cylinder Low	Cylinder Low
8AH	Cylinder High	Cylinder High
8CH	SDH Reg	SDH Reg
8EH	Status Reg	Command Reg
90H	None	None
92H	None	Asserts $\overline{\text{BCR}}$
94H	None	Asserts BRDY

NOTE:
Address assignments are determined by the host board.

Sector Buffer Isolation Logic

The host will be isolated from the SBX board whenever the 82064 is accessing its sector buffer which is enabled by $\overline{\text{BCS}}$. The host's control signals, $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{MCSO}}$, and $\overline{\text{MCS1}}$ and data bus are also disabled at the same time to prevent any data in the sector buffer from being corrupted. The host should wait for an interrupt before reading the 82064's Status register. Attempting to read the SBX board while $\overline{\text{BCS}}$ is active will return invalid data, since the SBX board will have the data bus tristated.

Wait State Logic

The wait state logic drives the 'not ready' line, $\overline{\text{MWAIT}}$, active whenever the host reads the SBX board. $\overline{\text{MWAIT}}$ does not go active for buffer or 82064 register writes. This logic was required for two reasons. First, a delayed read is generated, because the address setup to $\overline{\text{RD}}$ margin of the SBX bus is less than the 82064's needs (50 ns vs 100 ns). Second, the $\overline{\text{RD}}$ to data valid access period of the 82064 (375 ns), is greater than the SBX bus' full speed read cycle (275 ns) permits.

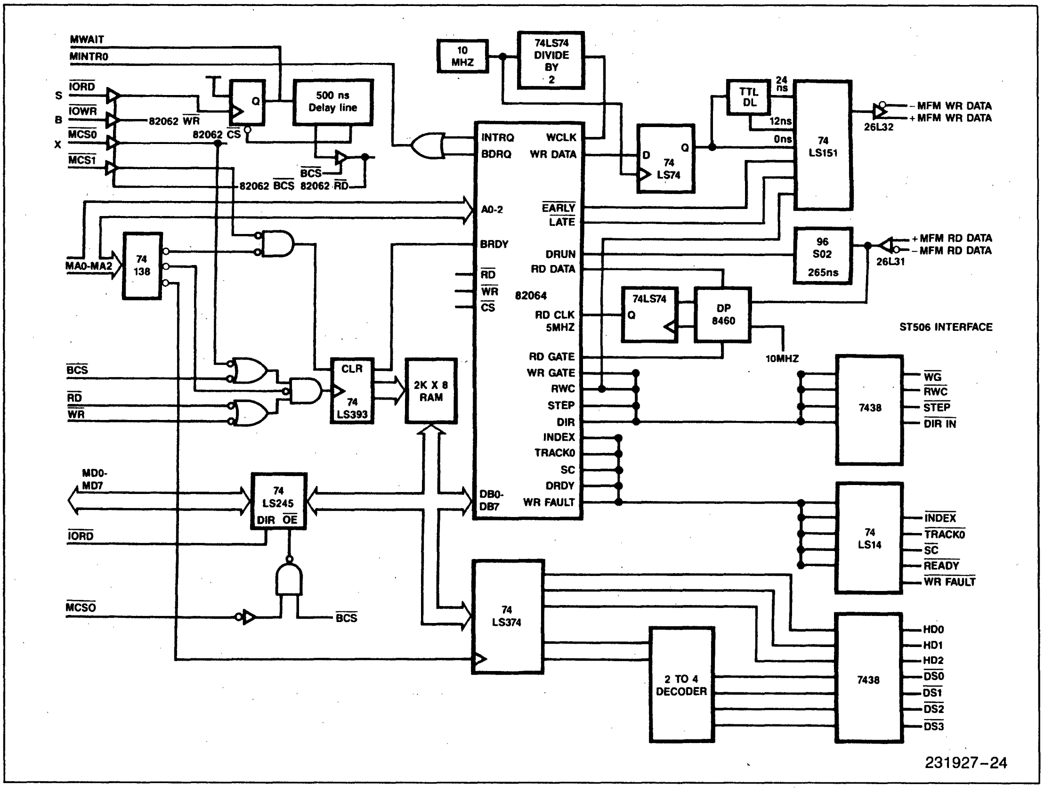


Figure 29. 82064 SBX Multimodule Diagram

MWAIT is deactivated after allowing for the delayed RD and the access period of the 82064. This delay is accomplished with a 500 ns delay line. The first tap at 100 ns generates the read request to allow for the address setup margin. The next tap 400 ns later removes MWAIT to allow the host to continue.

Sector Buffer

The sector buffer consists of an address counter (using '1s393's) and a 2 kbyte static RAM. The address counter is incremented on the trailing edge of a valid RD or WR cycle, either host microprocessor or 82064 initiated. The counter is reset by a hardware reset, the 82064 buffer reset BCR, or by accessing an I/O port to provide software control. The 82064 will issue BCR each time BCS changes state (i.e. twice per sector). Resetting the buffer counter can be put under software control for multiple sector transfers. BRDY going high tells the 82064 that the buffer is available for its use. BRDY is generated by the address counter, by filling or emptying the entire buffer in multiple sector transfers, or from an I/O port when single sector transfers are done (since single sectors won't use all 2 kbytes of the buffer, the hardware signal will not be generated). When the 82064 is using the buffer, BCS will be low, and the RD or WR line will be pulsed every 1.6 microseconds.

When the 82064 is using the buffer it prevents access by the host by tristating the read, write, select and data lines with a low on BCS.

SDH Register Logic

The drive and head select bits must be latched externally to the 82064, since these outputs are not provided. An 8 bit latch is strobed on the trailing edge of the WR pulse when the SDH register is selected. The two drive select bits are then demultiplexed to provide a one of four drive select line. If multiple drives are used then these outputs would also be used to select which disk's read data line would be gated into the PLL.

Interrupts

While the interrupt line is programmable (to notify of an end of command or data transfer request for the Read Sector command only), software will ensure that the interrupt from the 82064 signifies command termination. The BDRQ line is OR'ed with the 82064's INTRQ line or BDRQ can generate its own interrupt. BDRQ is also gated off-board for a DMA controller.

Disk Interface

Figure 30 is a block diagram of the interface between the 82064 and the disk drive. The functional blocks are:

Write Data Logic

Read Data Logic (PLL)

Drive Control

Write Data Logic

The WR DATA output requires a D flip-flop clocked at 10 MHz to complete the conversion of data to MFM. The output of this D flip-flop is true MFM and is sent to a delay line. A delay line determines the amount of delay for precompensation. No delay corresponds to shifting the data bit early; the first tap is approximately 12 ns of delay and is the "normal", or no delay and the second tap provides 12 ns of delay, referenced to the "normal" write data. Which output is selected is determined by the states on RWC, Early and Late. This function was generated with a 74s151 multiplexer. When RWC is inactive EARLY and LATE only select "normal" data since they are always active. The precompensated write data is then driven onto the data cable by an RS-422 driver.

Read Data Logic

The PLL generates the RD CLOCK that is used to decode the serial MFM data from the drive. A selected drive issues read data, unless WR GATE is active. A one-shot generates a pulse of 220–270 ns to provide the DRUN input. Only during an all zero's or one's field will the DRUN input stay high, as it will be retriggered every 200 ns (the minimum distance that separates continuous clock and data bits). As soon as DRUN is determined to be valid, the RD GATE output will go active, switching the PLL from the 10 MHz local clock input to disk data. The PLL will synchronize to the incoming serial data and generate a Read Clock of the proper timing and phase. The 82064 will then start to search for the address mark which is indicated by DRUN going low at the address mark.

No detail is provided herein on PLL design, as it is beyond the scope of this document. PLL design should be left to experienced designers, since minute changes in temperature and component values will drastically affect the soft error rate. As an alternative, several companies manufacture very high speed PLL chips for MFM encoded disk drives. Besides being fairly easy to design in, they reduce the number of components and board area needed for the sophisticated PLL.

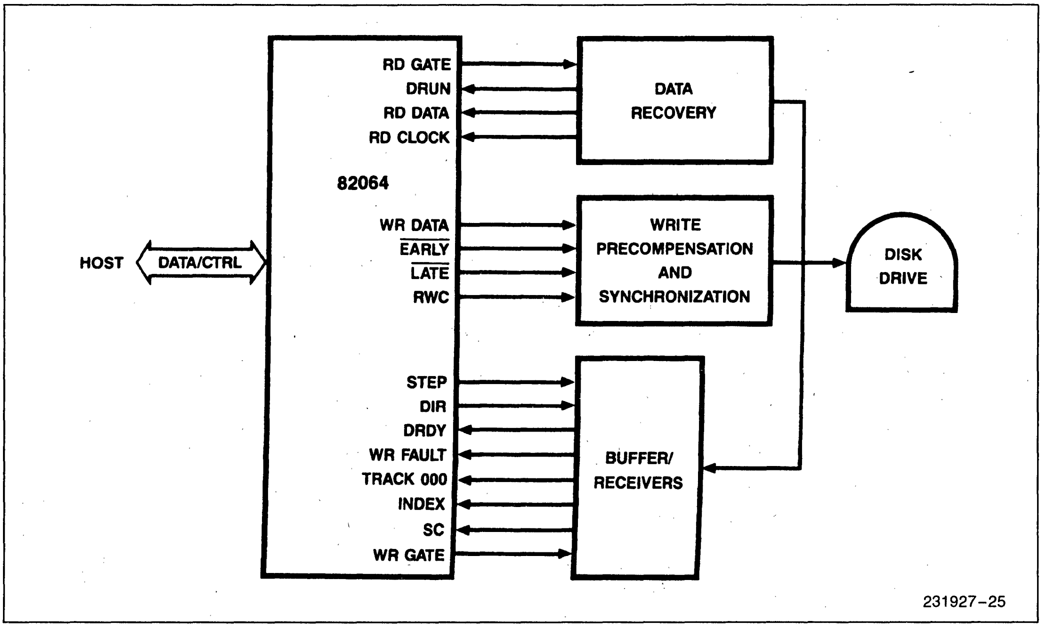


Figure 30. 82064 Disk Interface Block Diagram

6.3 Software Driver Overview

Presented in Appendix B is a listing of the software used to exercise the SBX 82064 board. Communication between the host software and the SBX driver routine is done through a structure located in system RAM. The host routine fills in required parameters, then passes the address of this communication block to the SBX driver routine. The driver routine pulls necessary values from this command block (CBL), executes a disk operation, then fills the CBL with the 82064's register contents, plus status and error information. The command block structure is shown in Figure 31.

Command	Byte
Rwc Reg	Byte
Sector Cnt.	Byte
Sector Num.	Byte
Cyl Low	Byte
Cyl High	Byte
SDH Reg	Byte
Status Reg	Byte
Error Reg	Byte
Host Buffer	Pointer

Figure 31

The host board did not have a DMA controller available, so an interrupt is issued from the BDRQ line and OR'ed with the 82064's interrupt line as interrupt sources were limited by the host. When an interrupt occurs, the interrupt procedure checks for either a data transfer, and executes it, or the completion of the command. If the interrupt signifies command completion, the interrupt procedure fills the command block with the 82064's task, status and error registers.

In this example, the host software examines one byte in the command block and until this byte is changed to a 00, no other command blocks will be passed to the disk driver routine. An alternative would be to issue a software interrupt to notify the microprocessor that the disk operation has finished and the command block contains parameters from the last operation and that a new disk command could start.

The driver for this example allows polling for non-data transfer commands, and must use interrupts for data transfers. As mentioned earlier, microprocessor intervention is required since the sector buffer is much larger than one sector and will not generate a BRDY. The microprocessor must write to an I/O port, which sets BRDY, after each host to sector buffer transfer. An actual software implementation would not include the polling and interrupt routines together, as only one method would generally be used.

The calling routine, which would normally be a directory program, places the values for which sector, number of sectors, etc., in the CBL. The disk routine is called and the address of this structure is passed on the stack. The disk driver places these parameters in the 82064's Task registers and initiates a command.

If the interrupt driven method was chosen, the disk driver routine returns to the calling routine. This permits other processing to be performed while the disk is executing a command. At some point, an interrupt will be generated, either from BRDY or INTRQ. Control will pass to the driver and the status register will be checked. If a data transfer is needed, either the microprocessor can transfer data or a DMA controller can perform the function. Once the transfer of data to the buffer is finished the microprocessor must set BRDY through an I/O port.

APPENDIX A
ST506 INTERFACE

THE ST506 INTERFACE

The ST506 interface is a modified version of Shugarts floppy disk drive interface and has been promoted by Seagate Technology. This interface is intended to be easy and low in cost to implement, yet provide a medium level of performance. The interface rigidly defines several areas: the hardware interconnects, the data transfer rate, the data encoding method, and how the disk is formatted.

Data Transfer Rate

The data transfer rate depends upon the linear bit density of the disk media and the speed at which the disk spins. ST506 specifies a 5 Mbit/second transfer rate. The typical ST506 drive has a nominal linear density of 10,416 bytes and a disk speed of 3600 rpm, which yields a 5 Mbit/second data transfer rate. No deviation from 5 M/bits second is allowed.

Increasing the linear density to increase storage capacity would require a decrease in disk speed. Otherwise, the data rate would increase. This decrease in disk speed would cause access times to increase, which many would deem unacceptable. To increase storage capacity, and remain ST506 compatible, either the number of cylinders and/or the number of platters can increase.

Data Encoding

ST506 requires that the serial data, sent between the drive and the controller, be encoded according to MFM rules. The basic unit of storage is a bit cell, which stores one bit information. This bit cell is divided into two halves, consisting of a clock bit and a data bit (see Figure A-1).

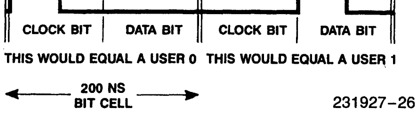


Figure A-1

The encoding rules for MFM are fairly simple:

1. A clock bit is written when the previous and the current bit cell does not contain a data bit.
2. A data bit is written whenever there is a "one" from the user.

Sync fields are composed of zeroes which generates a series of clock bits in the bit cell's. A phase lock loop locks on to the data stream during this period and generates a signal of the proper phase and frequency which is used to decode the combined clock and data serial data stream.

Disk Format

All disk media must be written with a specified format so that data may be reliably stored and retrieved. The smallest unit of controller accessible data is the sector which typically contains sync fields, ID fields, and a data field, and buffer fields.

The format of the disk required by ST506 is shown in Figure A-2. It should be noted that this format is fixed in the 82064. The user has options only for GAP1 and 3 length (when changing sector size or ECC) and whether to have 82064 CRC checking or user supplied ECC syndrome bits.

Gap 1 – Index Gap

Gap 1 serves two purposes. The first is to allow for variations in the index pulse timing due to motor speed variations. The second purpose is to allow a small delay to permit a different head to be selected without missing a sector. This is more of a data transfer optimization function and requires the disk controller to know which head is to be selected, when the last sector of a track has been read, and the next logical sector in the file exists on another platter. The 82064 does not switch heads automatically. Whether this scheme can be used or not depends upon the μP being able to alter one register in the 82064, before the next sector passes beneath the heads.

This gap is typically 12 bytes long and is written by the 82064 as 4E Hex.

Gap 2 - Write Splice Gap

This gap follows the CRC bytes of the ID field and continues up to the data field address mark. When updating a previously written sector, motor speed variations could turn on the write coil, as the head was passing over the ID field. This gap prevents this from occurring. The value written is OOH and also serves as the PLL sync field for the data field. The minimum value is determined by the "lock up" performance of the PLL. The 82064 writes sixteen bytes for this field once WG is activated. The user has no control over this field.

Gap 3 - Post Data Field Gap

Gap 3 is very similar to Gap 2 as it is used as a speed tolerance buffer also. Without this gap, and with the motor speed varying slightly, it would be possible for the upcoming sector's sync field and ID field to be overwritten. This value is '4E' H and is typically 15 bytes long. The 82064's Gap 3 length is programmable. The exact value is dependent upon several factors. Refer to 82064 Format command, Software Section: General Programming Section.

Gap 4 - Track Buffer Gap

This gap follows the last sector on a track and is written until an index pulse is received. Its purpose is to prevent the last sector from overflowing past the index gap, and absorb track length variations when ECC is used (ECC uses more bytes than CRC). The value is '4E' H and is about 320 bytes when CRC and 256 byte sectors are used. The 82064 writes this field only during formatting. The user has no control over the number of bytes written with the 82064.

ID Fields

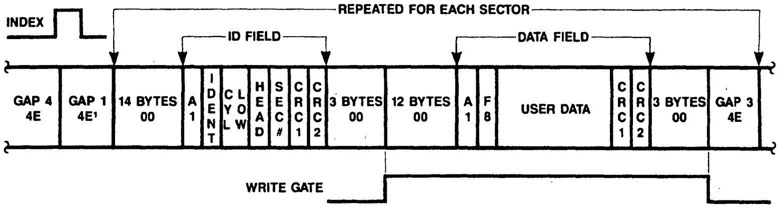
The controller uses ID fields to locate any individual sector. An address mark of two bytes precedes the ID field and the data field in a sector. An address mark tells the controller the nature of the upcoming information. ID fields are used by the disk controller and are not passed to the host.

Sector Interleaving

Sector interleaving occurs when logical sectors are in a non-sequential order, which is determined during formatting. An advantage is that there is a delay between logically sequential sectors. This delay can be used for data processing and then deciding if the next sector should be read. Without interleaving, the next sector could slip by, imposing a one revolution delay (approx. 16.7 ms). An additional benefit to this delay is that bus utilization is reduced by spreading the data transfer over a greater amount of time. The delay between sectors can be determined as follows:

$$\frac{\text{1 Revolution Period}}{\text{Sectors/Track}} \times (\text{Interleave factor} - 1) = \text{Delay}$$

For the typical ST506 drive with four-way interleaving this yields 1.57 ms of delay.



ID FIELD

A1 = A1 HEX WITH 0A HEX CLOCK
IDENT = 2LS.B. = CYLINDER HIGH
FE = 0-255 CYLINDERS
FF = 256-511 CYLINDERS
FC = 512-767 CYLINDERS
FD = 768-1023 CYLINDERS
F6 = 1024-1279 CYLINDERS
F7 = 1280-1535 CYLINDERS
F4 = 1536-1791 CYLINDERS
F5 = 1792-2047 CYLINDERS

HEAD = BITS 0, 1, 2 = HEAD NUMBER
BITS 3, 4 = 0
BITS 5, 6 = SECTOR SIZE
BIT 7 = BAD BLOCK MARK
SEC # = LOGICAL SECTOR NUMBER

DATA FIELD

A1 = A1 HEX WITH OA HEX CLOCK
F8 = DATA ADDRESS MARK; NORMAL CLOCK
USER = DATA FIELD 128 TO 1024 BYTES

231927-27

Figure A-2. Format Field

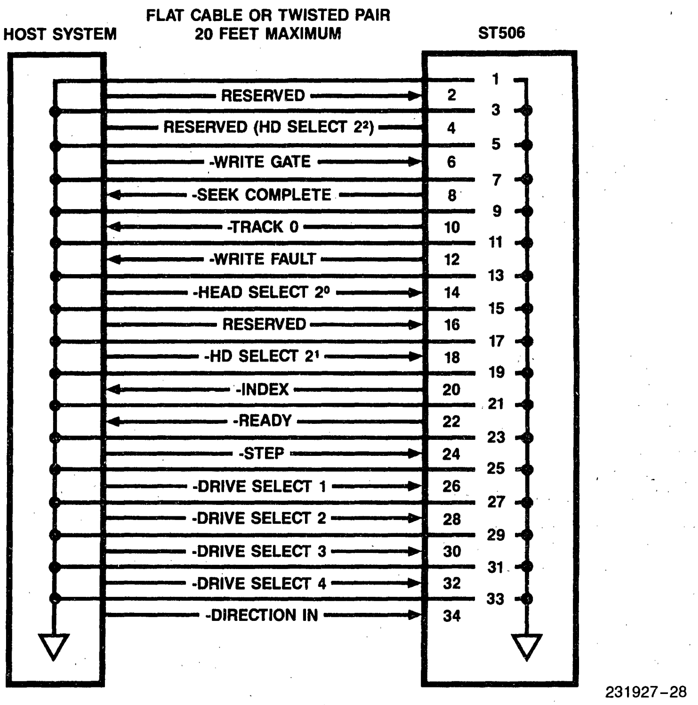


Figure A-3

The disadvantage to interleaving is that file transfers take longer, which may slow down the overall system. A four-way interleaved disk will have the transfer rate reduced to an average of 1.25 Mbit/sec.

The 82064 leaves the logical sector sequence to the user.

ELECTRICAL INTERFACE

The interface to the ST506 drive is divided into three categories and they are:

1. control signals,
2. data signals,
3. power.

Control Signals

The functions of the control signals are not covered in detail here. Their purpose can be found in the pin descriptions section. All control lines are digital in nature and either provide signals to the drive or inform the host of certain conditions. A diagram of the 34 pin control connector is shown in Figure A-3.

The driver/receiver logic diagram is shown in Figure A-4 and the electrical characteristics are:

	Voltage	Current
True	0.0 VDC to 0.4 VDC	−40 mA (IOL max.)
False	2,5 VDC to 5.25 VDC	250 μA (IOH open)

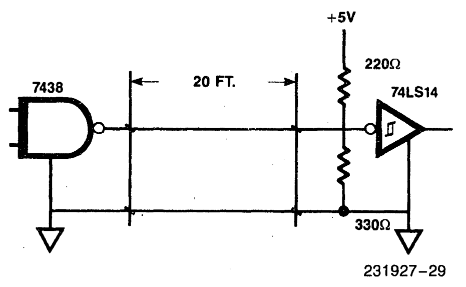


Figure A-4

Data Signals

The lines associated with the transfer of read/write data between the host and the drive are differential in nature and may not be multiplexed between drives. There is one pair of balanced lines for each read and write data line per drive and must conform to the RS-422 specification. Figure A-5 shows the receiver/transmitter combination.

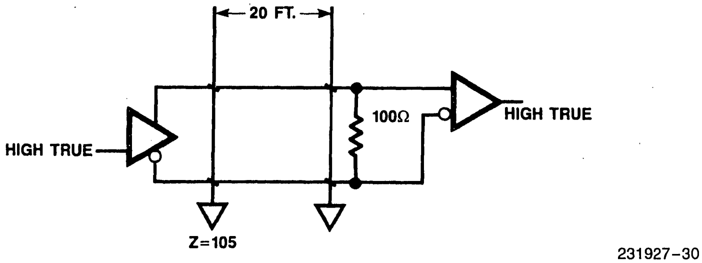
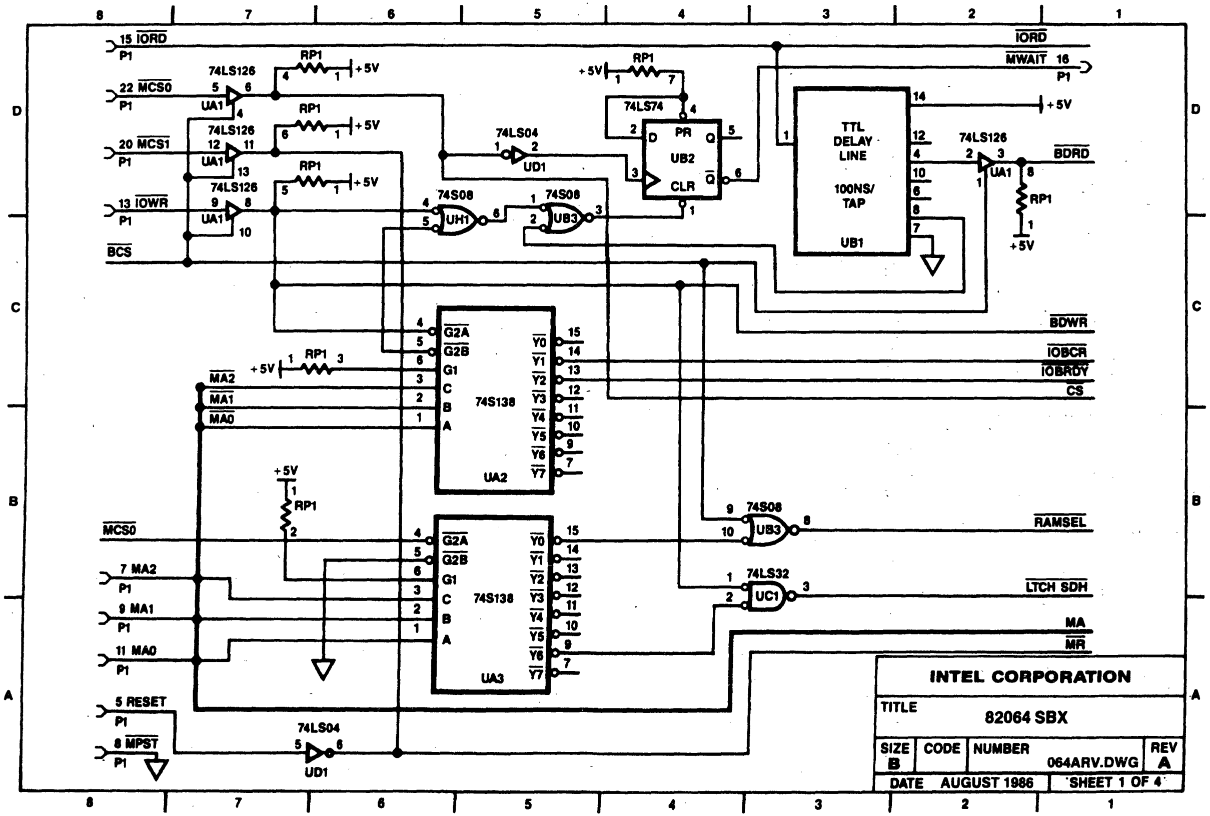
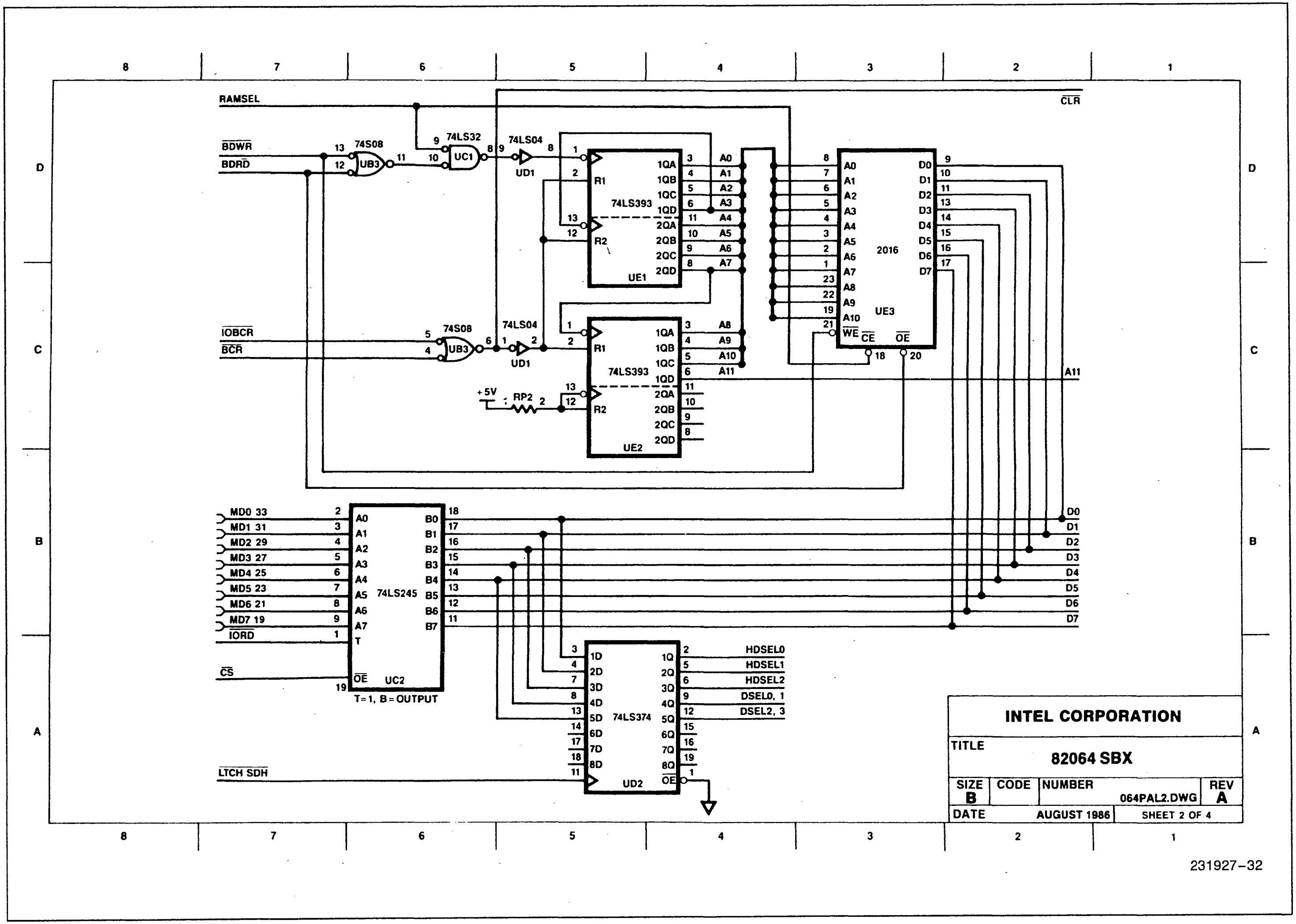


Figure A-5. E1A RS22 Driver/Receiver Pair Flat Ribbon or Twisted Pair

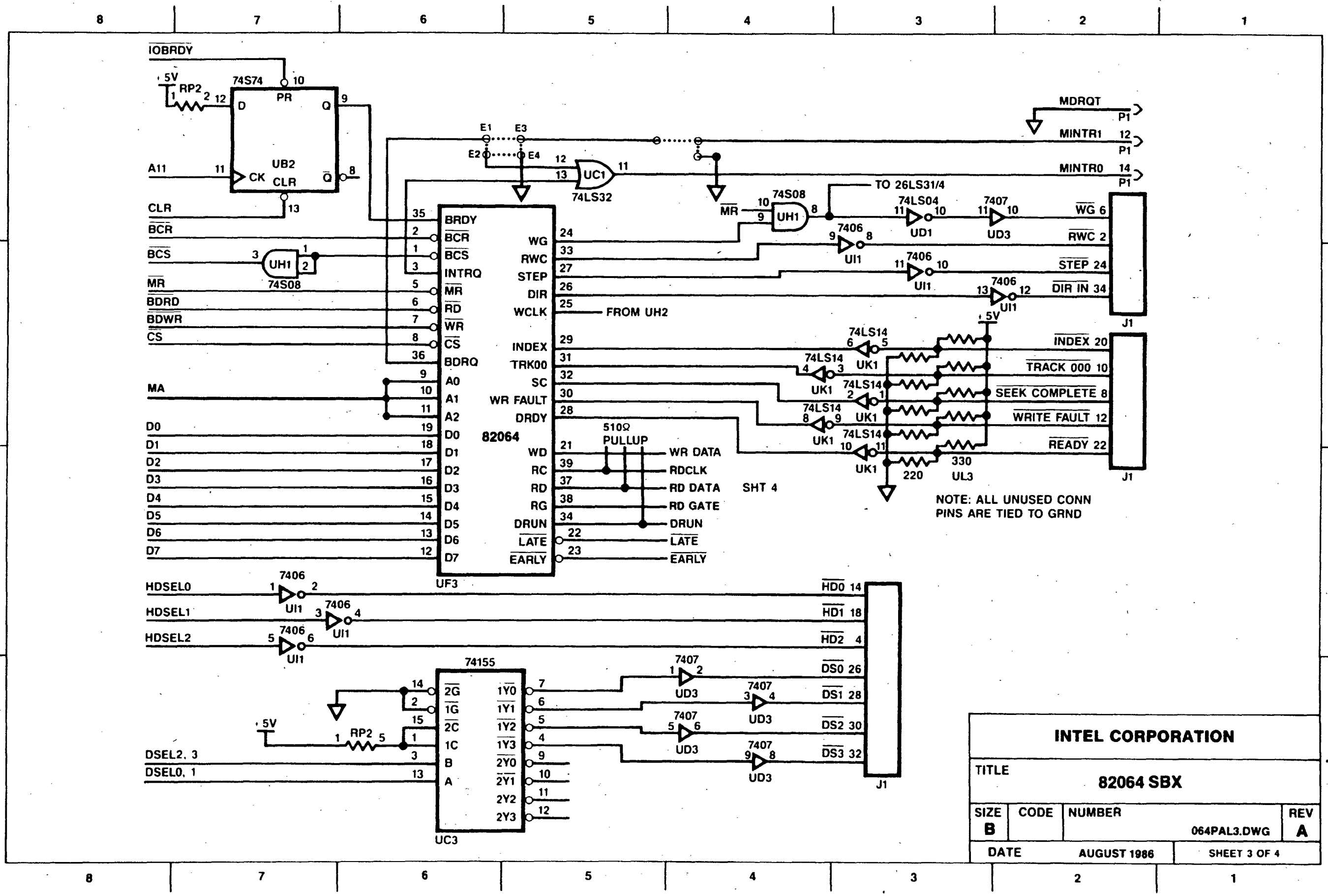
APPENDIX B SCHEMATICS



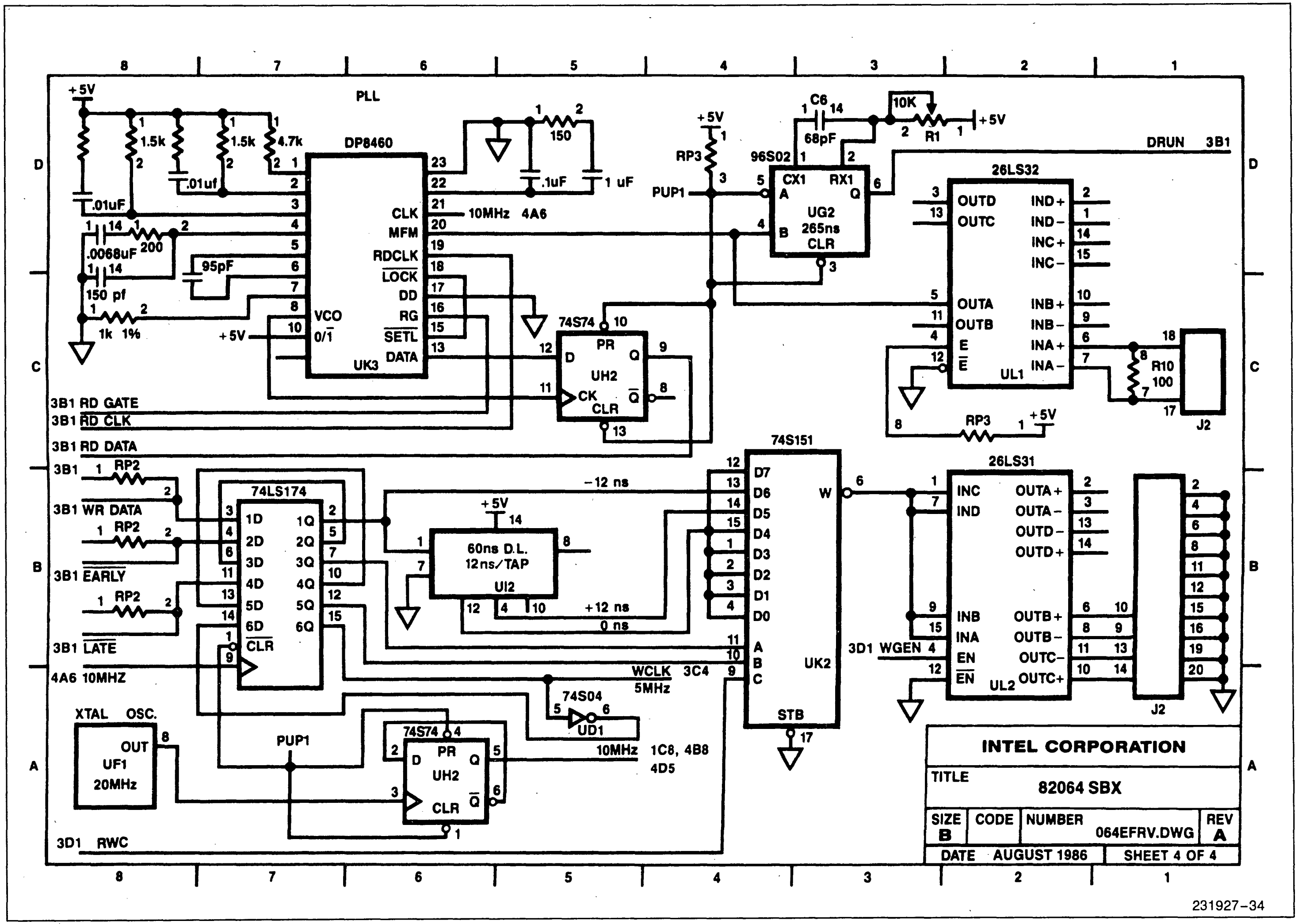
231927-31


RAMSEL
CLR
BDWR
BDRD
74S08
UB3
74LS32
UC1
74LS04
UD1
74LS393
UE1
1QA
1QB
1QC
1QD
2QA
2QB
2QC
2QD
R1
R2
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
2016
UE3
D0
D1
D2
D3
D4
D5
D6
D7
WE
CE
OE
IOBCR
BCR
UE2
+5V
RP2
MD0 33
MD1 31
MD2 29
MD3 27
MD4 25
MD5 23
MD6 21
MD7 19
IORD
CS
74LS245
UC2
T=1, B=OUTPUT
B0
B1
B2
B3
B4
B5
B6
B7
T
LTCH SDH
74LS374
UD2
1D
2D
3D
4D
5D
6D
7D
8D
1Q
2Q
3Q
4Q
5Q
6Q
7Q
8Q
HDSEL0
HDSEL1
HDSEL2
DSEL0, 1
DSEL2, 3
INTEL CORPORATION
TITLE
82064 SBX
SIZE
B
CODE
NUMBER
064PAL2.DWG
REV
A
DATE
AUGUST 1986
SHEET 2 OF 4

231927-32


IOBRDY
+5V
RP2
74S74
PR
D
Q
UB2
A11
CK
CLR
Q̄
CLR
BCR
BCS
UH1
74S08
MR
BDRD
BDWR
CS
MA
D0
D1
D2
D3
D4
D5
D6
D7
E1
E3
E2
E4
UC1
74LS32
BRDY
BCR
BCS
INTRQ
MR
RD
WR
CS
BDRQ
A0
A1
A2
D0
D1
D2
D3
D4
D5
D6
D7
82064
UF3
WG
RWC
STEP
DIR
WCLK
FROM UH2
INDEX
TRK00
SC
WR FAULT
DRDY
WD
RC
RD
RG
DRUN
LATE
EARLY
510Ω PULLUP
WR DATA
RDCLK
RD DATA
RD GATE
DRUN
LATE
EARLY
SHT 4
MDRQT
MINTR1
MINTR0
P1
TO 26LS31/4
74S08
UH1
74LS04
UD1
7407
UD3
7406
UI1
WG 6
RWC 2
STEP 24
DIR IN 34
J1
+5V
74LS14
UK1
INDEX 20
TRACK 000 10
SEEK COMPLETE 8
WRITE FAULT 12
READY 22
220
330
UL3
NOTE: ALL UNUSED CONN PINS ARE TIED TO GRND
HDSEL0
HDSEL1
HDSEL2
HD0 14
HD1 18
HD2 4
74155
UC3
2G
1G
2C
1C
B
A
1Y0
1Y1
1Y2
1Y3
2Y0
2Y1
2Y2
2Y3
DSEL2, 3
DSEL0, 1
DS0 26
DS1 28
DS2 30
DS3 32
INTEL CORPORATION
TITLE
82064 SBX
SIZE
B
CODE
NUMBER
064PAL3.DWG
REV
A
DATE
AUGUST 1986
SHEET 3 OF 4

231927-33



INTEL CORPORATION

TITLE	82064 SBX		
SIZE B	CODE	NUMBER 064EFRV.DWG	REV A
DATE AUGUST 1986		SHEET 4 OF 4	

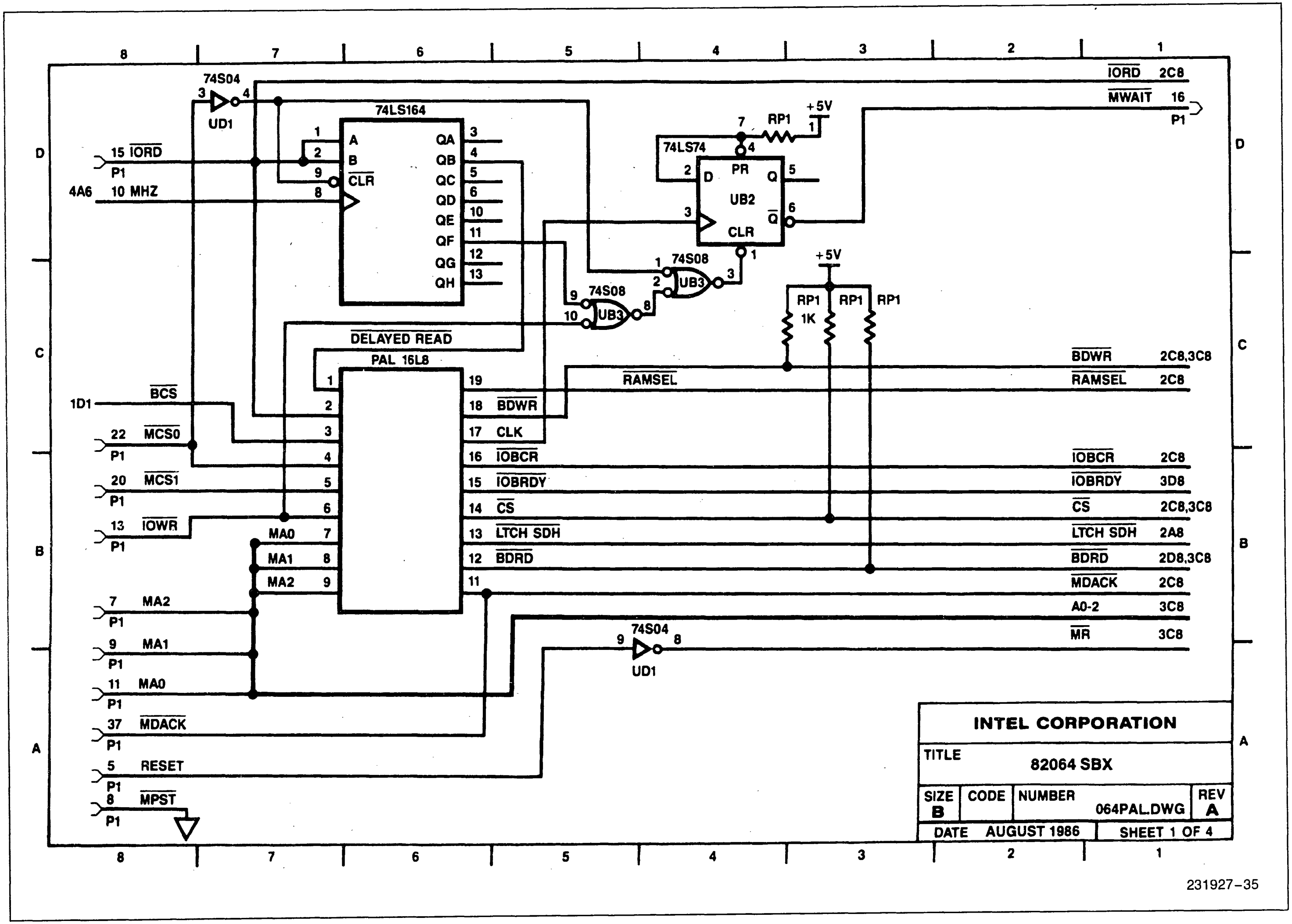
231927-34

APPENDIX C

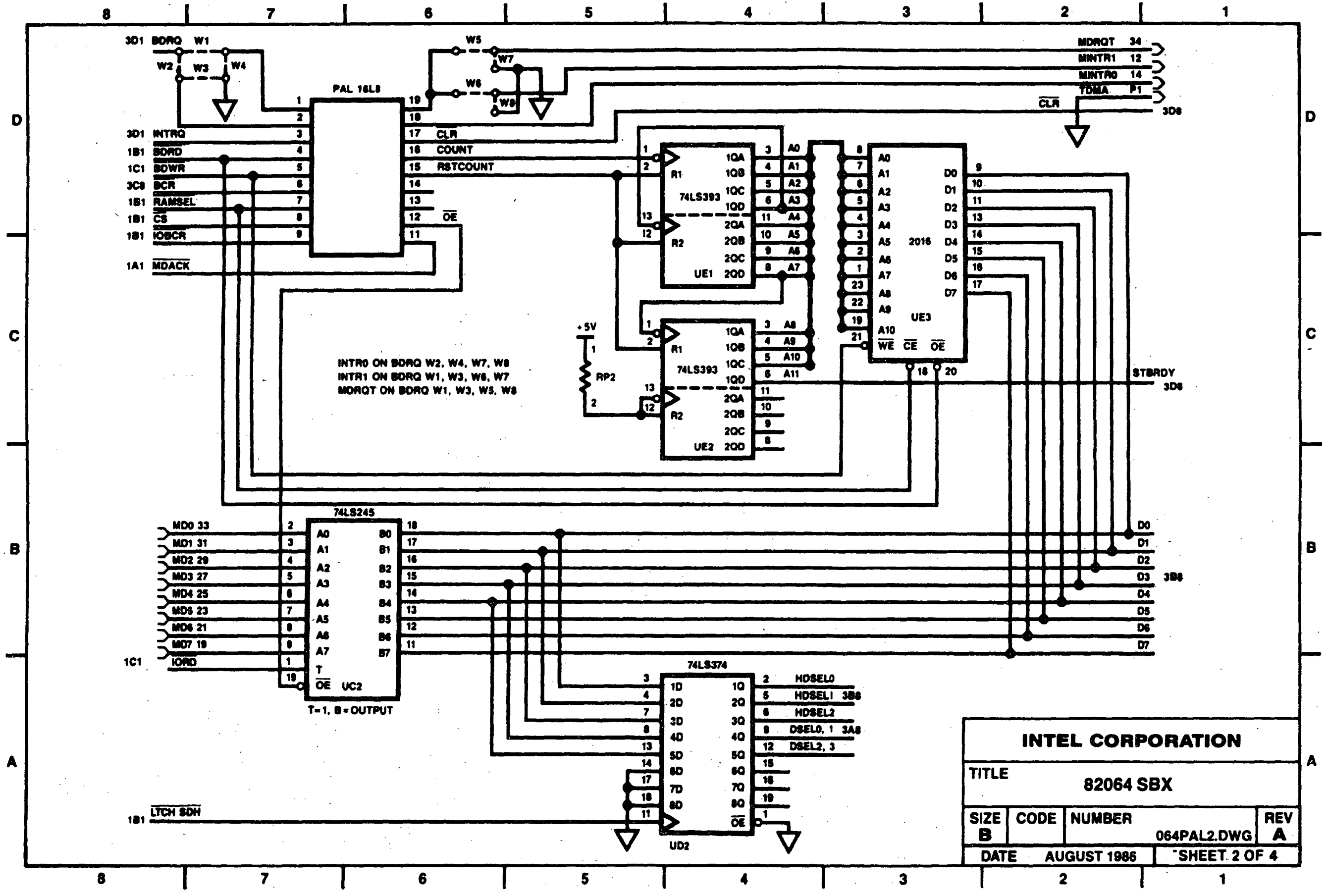
This appendix contains a schematic of the previous design using PAL's to replace the random logic. The previous design could not do DMA transfers and inserted a large delay when transferring data from buffer RAM to the system. The PAL version does do DMA transfers and buffer reads happen at full SBX bus speed. One other minor change was to replace the 500 ns delay line with a 74LS164, which is a more cost effective solution.

This schematic is only a paper design since only random logic was replaced with the PAL's.

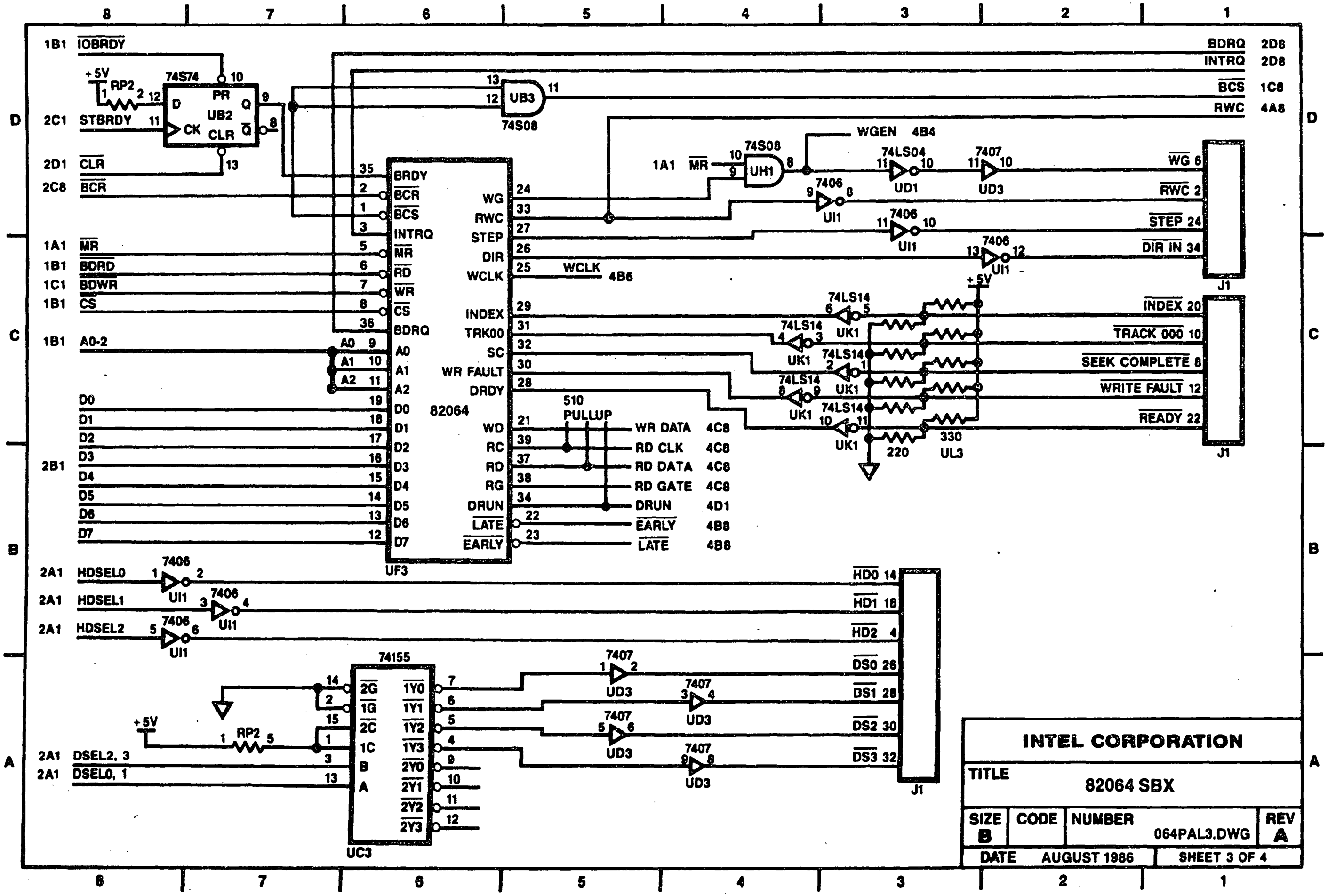
```
   PAL Equation's
PAL - Page 1:
   BDRD/ = (IORD/ * MDACK/) + (IORD/ * MCS0/ * MA0 * MA1 * MA2) +
(DELAYED-READ/ * CLK) IF BCS

   LTCHSDH/ = (MCS0/ * MA0/ * MA1 * MA2 * IOWR/)

   RAMSEL/ = (MCS0 * MA0 * MA1 * MA2) + (BCS/) + (MDACK/)

   IOBRDY/ = (MCS1/ * MA0/ * MA1 * MA2/ * IOWR/)

   IOBCR/ = (MCS1/ * MA0 * MA1/ * MA2/ * IOWR/)

   BDWR/ = (IOWR/) IF BCS

   CS/ = (MCS0/) IF BCS

   CLK = (MCS0/ * MA0 * MA1/ * MA2/) + (MCS0/ * MA0/ * MA1 * MA2/) + (MCS0/
* MA0 * MA1 * MA2/) + (MCS0/ * MA0/ * MA1/ * MA2) + (MCS0/ * MA0 * MA1/ *
MA2) + (MCS0/ * MA0/ * MA1 * MA2) + (MCS0/ * MA0 * MA1 * MA2)

PAL - Page 2:
   MINTR1/MDRQT = (PIN1)

   MINTR0 = (PIN2) + (INTRQ)

   COUNT = (BDWR/ + BDRD/) * (RAMSEL/)

   RSTCOUNT = (IOBCR/) + (BCR/)

   OE/ = (MDACK/) + (CS/)

   CLR/ = (IOBCR/) + (BCR/)
```


74S04
UD1
74LS164
A
B
CLR
QA
QB
QC
QD
QE
QF
QG
QH
15 IORD
P1
4A6
10 MHZ
74LS74
PR
D
Q
UB2
CLR
RP1
+5V
74S08
UB3
RP1
1K
DELAYED READ
PAL 16L8
BCS
1D1
22 MCS0
20 MCS1
13 IOWR
MA0
MA1
MA2
7 MA2
9 MA1
11 MA0
37 MDACK
5 RESET
8 MPST
19
18 BDWR
17 CLK
16 IOBCR
15 IOBRDY
14 CS
13 LTCH SDH
12 BDRD
11
RAMSEL
IORD 2C8
MWAIT 16
BDWR 2C8,3C8
RAMSEL 2C8
IOBCR 2C8
IOBRDY 3D8
CS 2C8,3C8
LTCH SDH 2A8
BDRD 2D8,3C8
MDACK 2C8
A0-2 3C8
MR 3C8
INTEL CORPORATION
TITLE 82064 SBX
SIZE B
CODE
NUMBER 064PAL.DWG
REV A
DATE AUGUST 1986
SHEET 1 OF 4

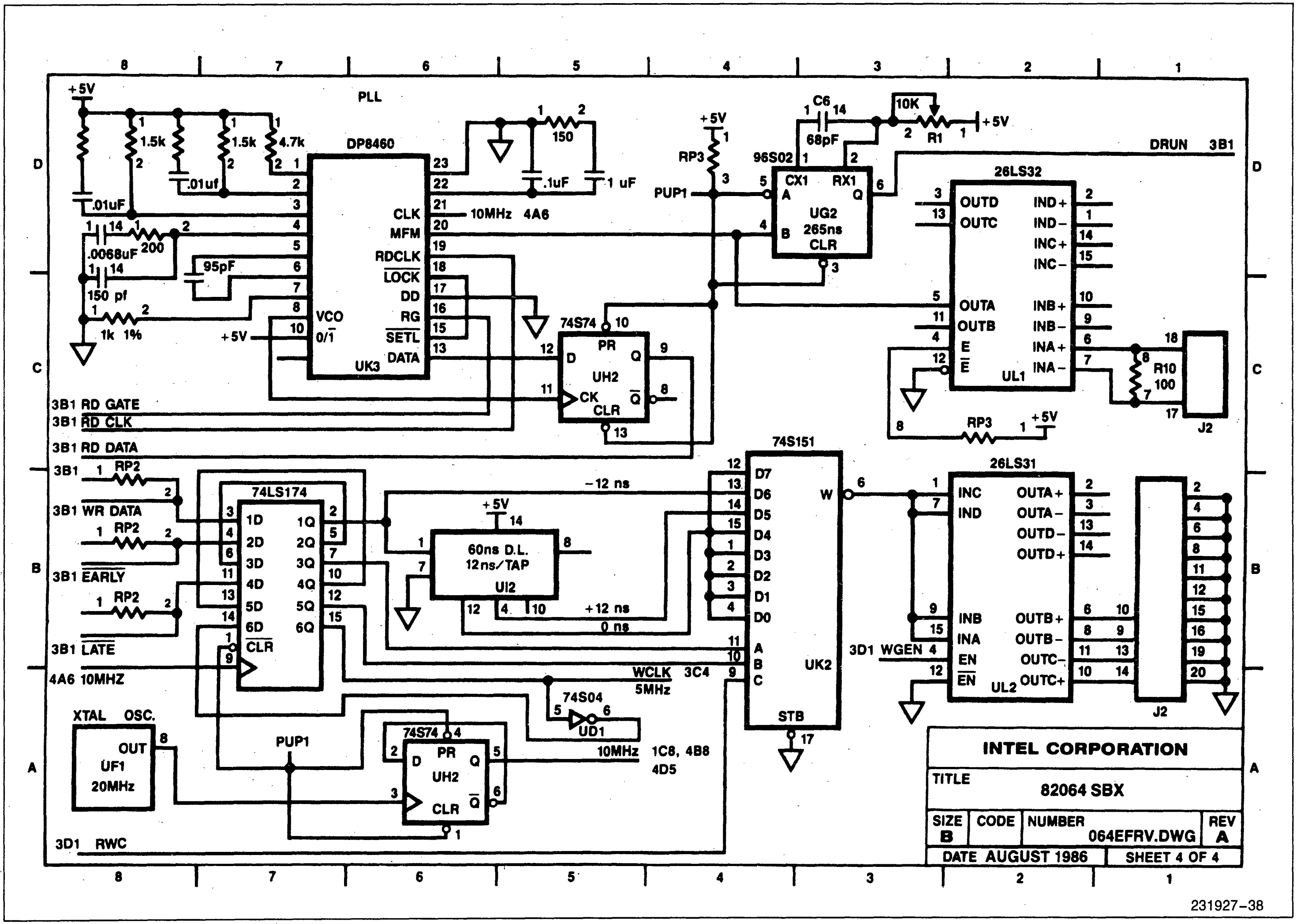
231927-35


3D1 BDRQ W1
W2 W3 W4
W5
W7
W6
W8
W9
PAL 16L8
3D1 INTRQ
1B1 BDRD
1C1 BDWR
3C8 BCR
1B1 RAMSEL
1B1 CS
1B1 IOBCR
1A1 MDACK
CLR
COUNT
RSTCOUNT
OE
MDRQT 34
MINTR1 12
MINTR0 14
TDMA F1
CLR
3D8
74LS393
1QA 1QB 1QC 1QD 2QA 2QB 2QC 2QD
R1 R2
UE1
UE2
+5V
RP2
2016
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11
WE CE OE
D0 D1 D2 D3 D4 D5 D6 D7
UE3
STBRDY
3D8
INTR0 ON BDRQ W2, W4, W7, W8
INTR1 ON BDRQ W1, W3, W6, W7
MDRQT ON BDRQ W1, W3, W5, W8
74LS245
MD0 33
MD1 31
MD2 29
MD3 27
MD4 25
MD5 23
MD6 21
MD7 19
1C1 IORD
A0 A1 A2 A3 A4 A5 A6 A7
B0 B1 B2 B3 B4 B5 B6 B7
T
OE UC2
T=1, B=OUTPUT
D0 D1 D2 D3 3B8 D4 D5 D6 D7
74LS374
1D 2D 3D 4D 5D 6D 7D 8D
1Q 2Q 3Q 4Q 5Q 6Q 7Q 8Q
OE
UD2
HDSEL0
HDSEL1 3B8
HDSEL2
DSEL0, 1 3A8
DSEL2, 3
1B1 LTCH SDH
INTEL CORPORATION
TITLE 82064 SBX
SIZE B
CODE
NUMBER 064PAL2.DWG
REV A
DATE AUGUST 1986
SHEET 2 OF 4
8 7 6 5 4 3 2 1
D C B A

231927-36


1B1 IOBRDY
+5V
RP2
74S74
D
CK
PR
UB2
CLR
Q
Q̄
2C1 STBRDY
2D1 CLR
2C8 BCR
1A1 MR
1B1 BDRD
1C1 BDWR
1B1 CS
1B1 A0-2
A0
A1
A2
D0
D1
D2
D3
D4
D5
D6
D7
2B1
UB3
74S08
BRDY
BCR
BCS
INTRQ
MR
RD
WR
CS
BDRQ
82064
UF3
WG
RWC
STEP
DIR
WCLK
INDEX
TRK00
SC
WR FAULT
DRDY
WD
RC
RD
RG
DRUN
LATE
EARLY
WCLK 4B6
510 PULLUP
WR DATA 4C8
RD CLK 4C8
RD DATA 4C8
RD GATE 4C8
DRUN 4D1
EARLY 4B8
LATE 4B8
BDRQ 2D8
INTRQ 2D8
BCS 1C8
RWC 4A8
WGEN 4B4
1A1 MR
74S08
UH1
74LS04
UD1
7407
UD3
7406
UI1
74LS14
UK1
+5V
220
330
UL3
WG 6
RWC 2
STEP 24
DIR IN 34
J1
INDEX 20
TRACK 000 10
SEEK COMPLETE 8
WRITE FAULT 12
READY 22
2A1 HDSEL0
2A1 HDSEL1
2A1 HDSEL2
HD0 14
HD1 18
HD2 4
74155
UC3
2G
1G
2C
1C
B
A
1Y0
1Y1
1Y2
1Y3
2Y0
2Y1
2Y2
2Y3
2A1 DSEL2, 3
2A1 DSEL0, 1
DS0 26
DS1 28
DS2 30
DS3 32
INTEL CORPORATION
TITLE 82064 SBX
SIZE B
CODE
NUMBER 064PAL3.DWG
REV A
DATE AUGUST 1986
SHEET 3 OF 4

231927-37



INTEL CORPORATION			
TITLE	82064 SBX		
SIZE B	CODE	NUMBER 064EFRV.DWG	REV A
DATE AUGUST 1986		SHEET 4 OF 4	

231927–38

Universal Peripheral Interface Slave Microcontrollers

9

UPI-452
CHMOS PROGRAMMABLE I/O PROCESSOR

83C452 - 8K × 8 Mask Programmable Internal ROM

80C452 - External ROM/EPROM

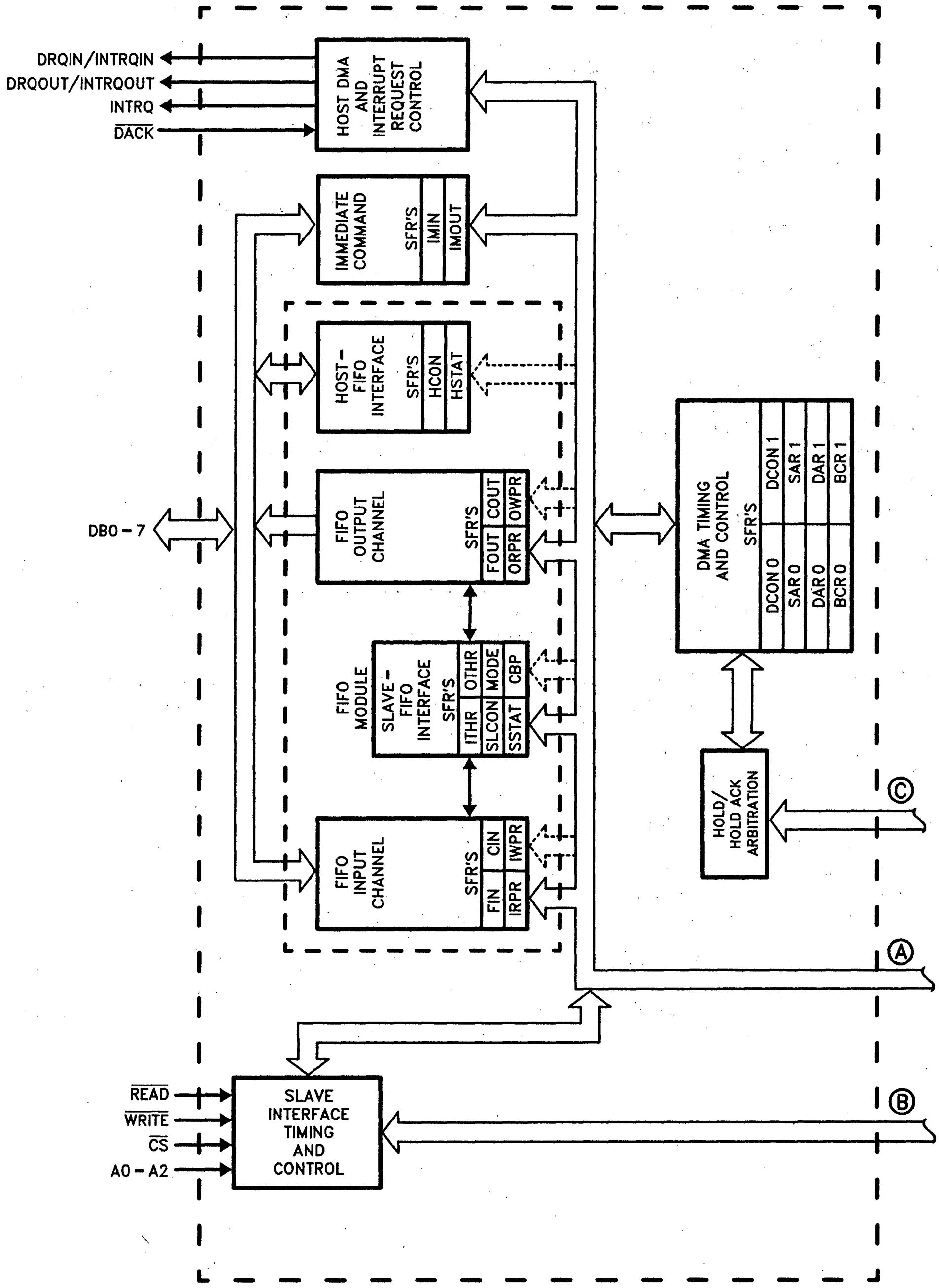
- **83C452/80C452:3.5 to 14 MHz Clock Rate**
- **Software Compatible with the MCS-51 Family**
- **128-Byte Bi-Directional FIFO Slave Interface**
- **Two DMA Channels**
- **256 × 8-Bit Internal RAM**
- **34 Additional Special Function Registers**
- **40 Programmable I/O Lines**
- **Two 16-Bit Timer/Counters**
- **Boolean Processor**
- **Bit Addressable RAM**
- **8 Interrupt Sources**
- **Programmable Full Duplex Serial Channel**
- **64K Program Memory Space**
- **64K Data Memory Space**
- **68-Pin PGA and PLCC**
  (See Packaging Spec., Order: #231369)

The Intel UPI-452 (Universal Peripheral Interface) is a 68 pin CHMOS Slave I/O Processor with a sophisticated bi-directional FIFO buffer interface on the slave bus and a two channel DMA processor on-chip. The UPI-452 is the newest member of Intel's UPI family of products. It is a general-purpose slave I/O Processor that allows the designer to grow a customized interface solution.

The UPI-452 contains a complete 80C51 with twice the on-chip data and program memory. The sophisticated slave FIFO module acts as a buffer between the UPI-452 internal CPU and the external host CPU. To both the external host and the internal CPU, the FIFO module looks like a bi-directional bottomless buffer that can both read and write data. The FIFO manages the transfer of data independent of the UPI-452 core CPU and generates an interrupt or DMA request to either CPU, host or internal, as a FIFO service request.

The FIFO consists of two channels:the Input FIFO and the Output FIFO. The division of the FIFO module array, 128 bytes, between Input channel and Output channel is programmable by the user. Each FIFO byte has an additional logical ninth bit to distinguish between a data byte and a Data Stream Command byte. Additionally, Immediate Commands allow direct, interrupt driven, bi-directional communication between the UPI-452 internal CPU and external host CPU, bypassing the FIFO.

The on-chip DMA processor allows high speed data transfers from one writeable memory space to another. As many as 64K bytes can be transferred in a single DMA operation. Three distinct memory spaces may be used in DMA operations; Internal Data Memory, External Data Memory, and the Special Function Registers (including the FIFO IN, FIFO OUT, and Serial Channel Special Functions Registers).



231428–1

Figure 1. Architectural Block Diagram

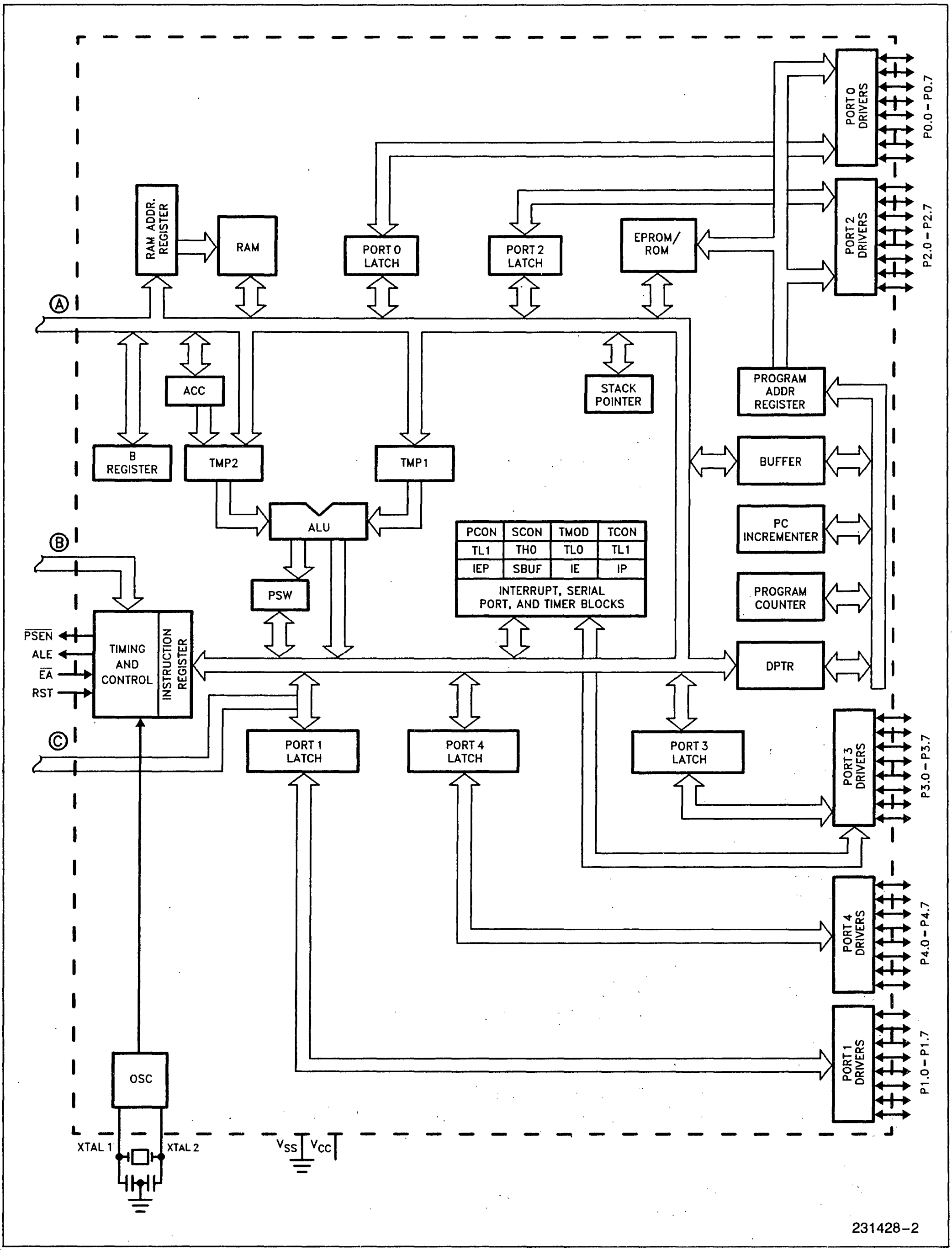


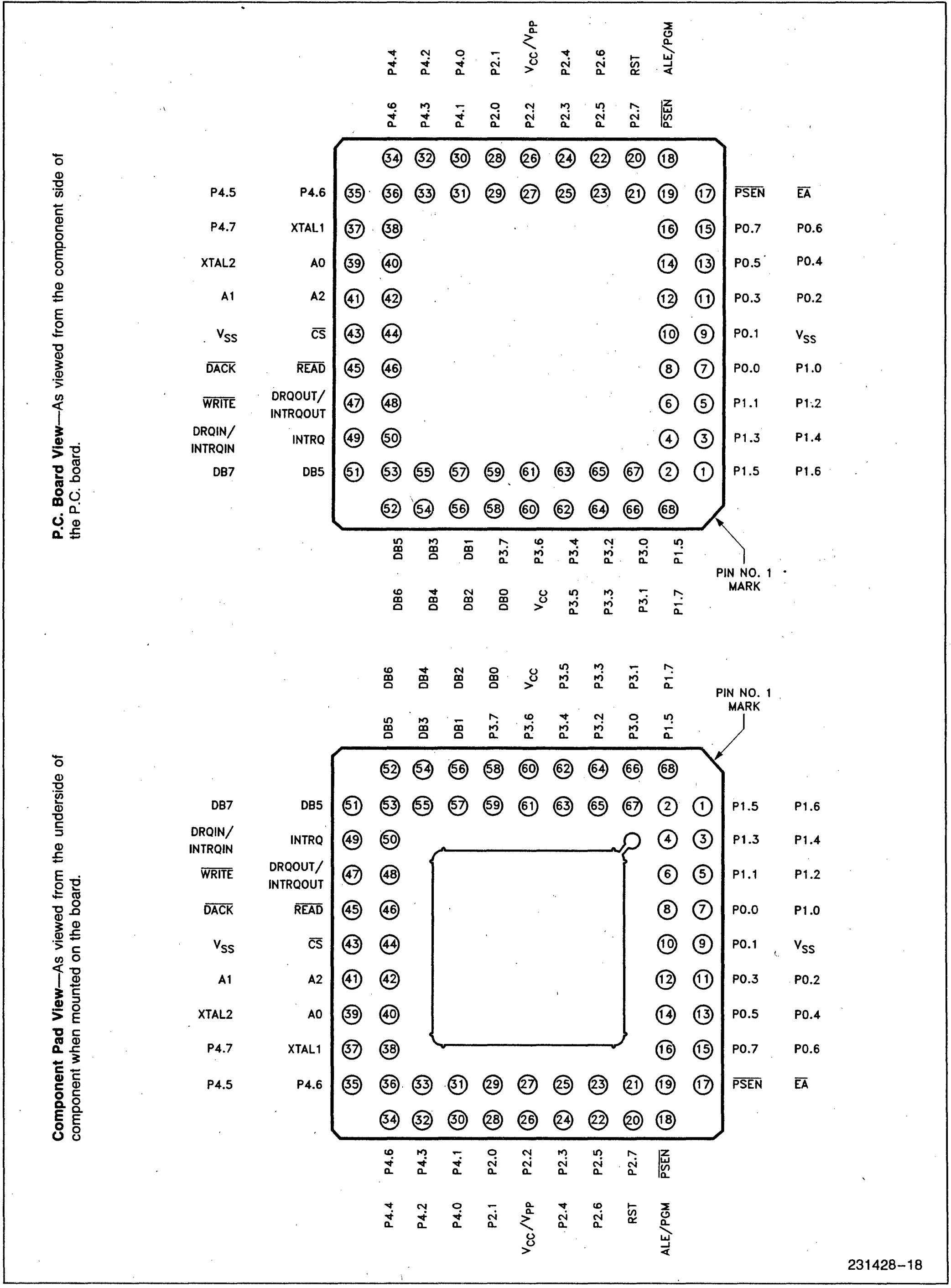
Figure 1. Architectural Block Diagram (Continued)

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

Figures:

Tables:

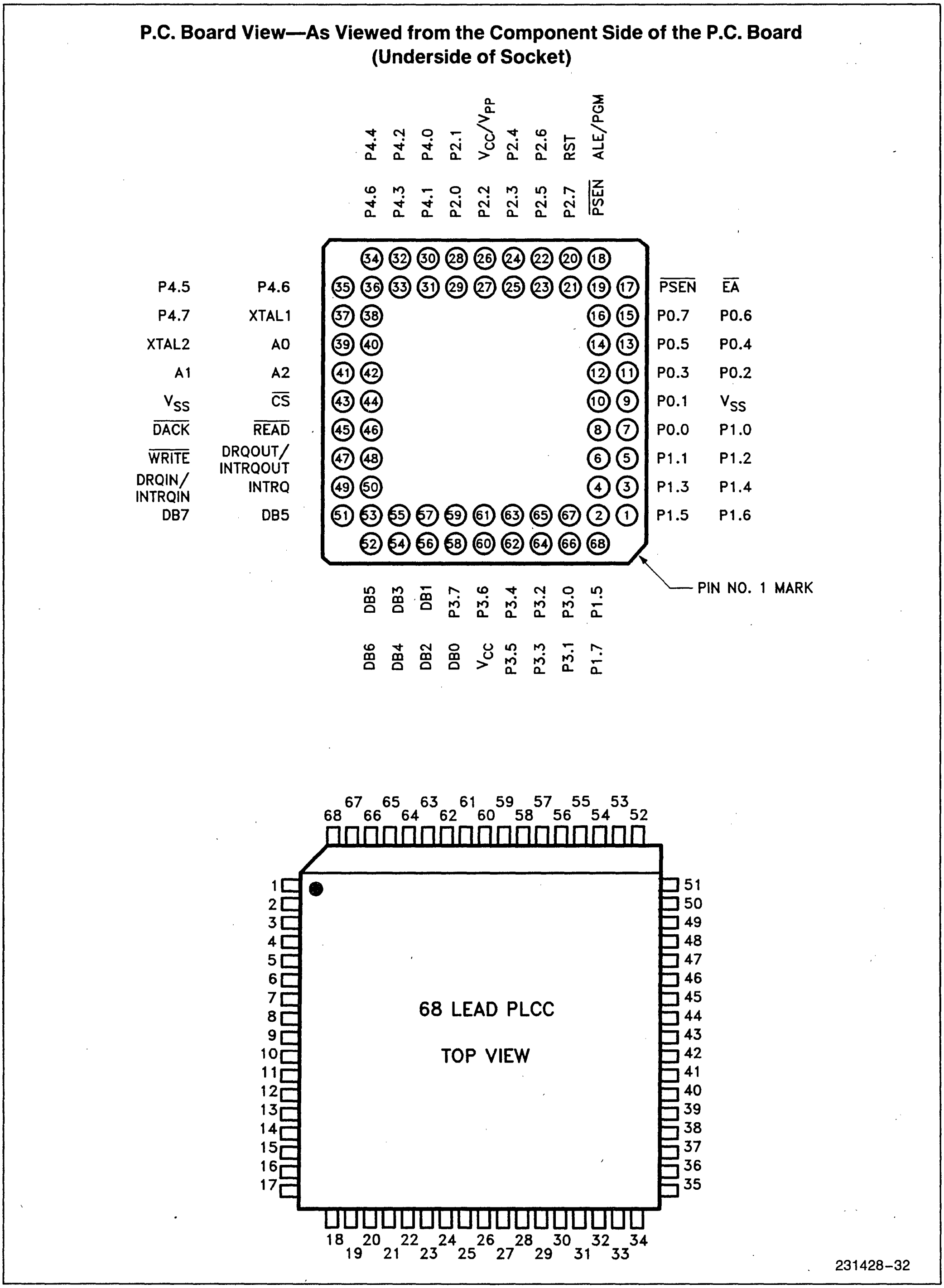


P.C. Board View—As viewed from the component side of the P.C. board.

Component Pad View—As viewed from the underside of component when mounted on the board.

Figure 2. UPI-452 68-Pin PGA Pinout Diagram

P.C. Board View—As Viewed from the Component Side of the P.C. Board (Underside of Socket)



231428-32

Figure 2A. UPI 452 68-Pin PLCC Pinout Diagram

UPI MICROCONTROLLER FAMILY

The UPI-452 joins the current members of the UPI microcontroller family. UPI's are derivatives of the MCS™ family of microcontrollers. Because of their on-chip system bus interface, UPI's are designed to be system bus "slaves", while their microcontroller counterparts are intended as system bus "masters".

These UPI Microcontrollers are fully supported by Intel's development tools (ICE, ASM and PLM).

Packaging

The 80C452/83C452 is available in either a 68-pin PGA (Pin Grid Array) or 68-pin PLCC package.

UPI Family (Slave Configuration)	MCS Family (Master Configuration)	Speed	RAM (Bytes)	ROM (Bytes)
80C452	80C51	12 MHz	256	—
83C452	80C51	12 MHz	256	8K
80C452-1	80C51	14 MHz	256	—
83C452-1	80C51	14 MHz	256	8K

UPI-452 PIN DESCRIPTIONS

Symbol	Pin #	Type	Name and Function
V_{SS}	9/43	I	Circuit Ground.
V_{CC}	60	I	+5V power supply during normal and idle mode operation. It is also the standby power pin for power down mode.
XTAL1	38	I	Input to the oscillator's high gain amplifier. A crystal or external source can be used.
XTAL2	39	O	Output from the high gain amplifier.
Port 0 (AD0–AD7) P0.0 .1 .2 .3 .4 .5 .6 P0.7	 8 10 11 12 13 14 15 16	I/O	Port 0 is an 8-bit open drain bi-directional I/O port. Port 0 can sink eight LS TTL inputs. It is also the multiplexed low-order address and data local expansion bus during accesses to external memory.

UPI-452 PIN DESCRIPTIONS (Continued)

Symbol	Pin #	Type	Name and Function
Port 1		I/O	Port 1 is an 8-bit quasi-bi-directional I/O port. Port 1 can sink four LS TTL inputs. The alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1. Otherwise, the port pin is stuck at 0. Pins P1.5 and P1.6 are multiplexed with $\overline{\text{HLD}}$ and $\overline{\text{HLDA}}$ respectively whose functions are defined as below: Port Pin — Alternate Function P1.5 — $\overline{\text{HLD}}$ —Local bus hold input/output signal P1.6 — $\overline{\text{HLDA}}$ —Local bus hold acknowledge input
(A0–A7)			
($\overline{\text{HLD}}$, $\overline{\text{HLDA}}$)			
P1.0	7		
.1	6		
.2	5		
.3	4		
.4	3		
.5	2		
.6	1		
P1.7	68		
Port 2		I/O	Port 2 is an 8-bit quasi-bi-directional I/O port. It also emits the high-order 8 bits of address when accessing local expansion bus external memory. Port 2 can sink four LS TTL inputs.
(A8–A15)			
P2.0	29		
.1	28		
.2	27		
.3	25		
.4	24		
.5	23		
.6	22		
.7	21		
Port 3		I/O	Port 3 is an 8-bit quasi-bi-directional I/O port. It is also multiplexed with the interrupt, timer, local serial channel, RD/ and WR/ functions that are used by various options. The alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1. Otherwise, the port pin is stuck at 0. Port 3 can sink four LS TTL inputs. The alternate functions assigned to the pins of Port 3 are as follows: Port Pin — Alternate Function P3.0 — RxD — Serial input port P3.1 — TxD — Serial output port P3.2 — INT0 — Interrupt 0 Input P3.3 — INT1 — Interrupt 1 Input P3.4 — T0 — Input to counter 0 P3.5 — T1 — Input to counter 1 P3.6 — WR/ — The write control signal latches the data from Port 0 outputs into the External Data Memory on the local bus. P3.7 — RD/ — The read control signal latches the data from Port 0 outputs on the local bus.
P3.0	67		
.1	66		
.2	65		
.3	64		
.4	63		
.5	62		
.6	61		
P3.7	59		

UPI-452 PIN DESCRIPTIONS (Continued)

Symbol	Pin #	Type	Name and Function
Port 4 P4.0 .1 .2 .3 .4 .5 .6 .7	 30 32 33 34 35 36 37	I/O	Port 4 is an 8-bit quasi-bi-directional I/O port. Port 4 can sink/source four TTL inputs.
RST	20	I	A high level on this pin for two machine cycles while the oscillator is running resets the device. An internal pulldown resistor permits Power-on reset using only a capacitor connected to V_{CC}. This pin does not receive the power down voltage as is the case for HMOS MCS-51 family members. This function has been transferred to the V_{CC} pin.
ALE	18	O	Provides Address Latch Enable output used for latching the address into external memory during normal operation. ALE can sink/source eight LS TTL inputs.
$\overline{\text{PSEN}}$	19	O	The Program Store Enable output is a control signal that enables the external Program Memory to the bus during normal fetch operation. $\overline{\text{PSEN}}$ can sink/source eight LS TTL inputs.
$\overline{\text{EA}}$	17	I	When held at TTL high level, the UPI-452 executes instructions from the internal ROM when the PC is less than 8192 (8K, 2000H). When held at a TTL low level, the UPI-452 fetches all instructions from external Program Memory.
DB0 DB1 DB2 DB3 DB4 DB5 DB6 DB7	58 57 56 55 54 53 52 51	I/O	Host Bus Interface is an 8-bit bi-directional bus. It is used to transfer data and commands between the UPI-452 and the host processor. This bus can sink/source eight LS TTL inputs.
$\overline{\text{CS}}$	44	I	This pin is the Chip Select of the UPI-452.
A0 A1 A2	40 41 42	I	These three address lines are used to interface with the host system. They define the UPI-452 operations. The interface is compatible with the Intel microprocessors and the MULTIBUS.
$\overline{\text{READ}}$	46	I	This pin is the read strobe from the host CPU. Activating this pin causes the UPI-452 to place the contents of the Output FIFO (either a command or data) or the Host Status/Control Special Function Register on the Slave Data Bus.
$\overline{\text{WRITE}}$	47	I	This pin is the write strobe from the host. Activating this pin will cause the value on the Slave Data Bus to be written into the register specified by A0–A2.
DRQIN/ INTRQIN	49	O	This pin requests an input transfer from the host system whenever the Input Channel requires data.
DRQOUT/ INTRQOUT	48	O	This output pin requests an output transfer whenever the Output Channel requires service. If the external host to UPI-452 DMA is enabled, and a Data Stream Command is at the Output FIFO, DRQOUT is deactivated and INTRQ is activated (see 'GENERAL PURPOSE DMA CHANNELS' section).

UPI-452 PIN DESCRIPTIONS (Continued)

Symbol	Pin #	Type	Name and Function
INTRQ	50	O	This output pin is used to interrupt the host processor when an Immediate Command Out or an error condition is encountered. It is also used to interrupt the host processor when the FIFO requests service if the DMA is disabled and INTRQIN and INTRQOUT are not used.
$\overline{\text{DACK}}$	45	I	This pin is the DMA acknowledge for the host bus interface Input and Output Channels. When activated, a write command will cause the data on the Slave Data Bus to be written as data to the Input Channel (to the Input FIFO). A read command will cause the Output Channel to output data (from the Output FIFO) on to the Slave Data Bus. This pin should be driven high (+5V) in systems which do not have a DMA controller (see Address Decoding).
V_{CC}	26	I	+5V power supply during operation.

ARCHITECTURAL OVERVIEW

Introduction

The UPI-452 slave microcontroller incorporates an 80C51 with double the program and data memory, a slave interface which allows it to be connected directly to the host system bus as a peripheral, a FIFO buffer module, a two channel DMA processor, and a fifth I/O port (Figure 3). The UPI-452 retains all of the 80C51 architecture, and is fully compatible with the MCS-51 instruction set.

The Special Function Register (SFR) interface concept introduced in the MCS-51 family of microcontrollers has been expanded in the UPI-452. To the 20 Special Function Registers of the MCS-51, the UPI-452 adds 34 more. These additional Special Function Registers, like those of the MCS-51, provide access to the UPI-452 functional elements including the FIFO, DMA and added interrupt capabilities. Several of the 80C51 core Special Function Registers have also been expanded to support added features of the UPI-452.

This data sheet describes the unique features of the UPI-452. Refer to the 80C51 data sheet for a description of the UPI-452's core CPU functional blocks including;

— Timers/Counters

— I/O Ports

— Interrupt timing and control (other than FIFO and DMA interrupts)

— Serial Channel

— Local Expansion Bus

— Program/Data Memory structure

— Power-Saving Modes of Operation

— CHMOS Features

— Instruction Set

Figure 3 contains a conceptual block diagram of the UPI-452. Figure 4 provides a functional block diagram.

FIFO Buffer Interface

A unique feature of the UPI-452 is the incorporation of a 128 byte FIFO array at the host-slave interface. The FIFO allows asynchronous bi-directional transfers between the host CPU and the internal CPU.

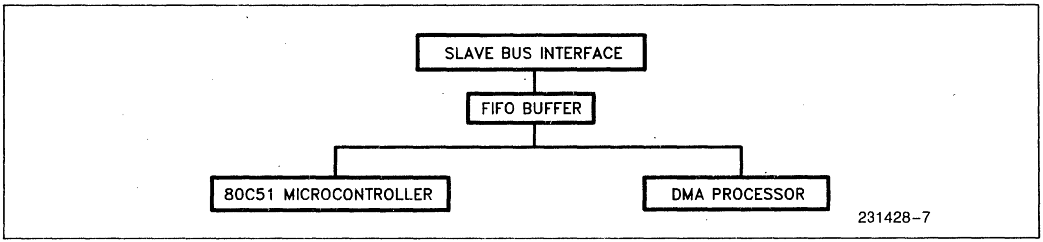


Figure 3. UPI-452 Conceptual Block Diagram

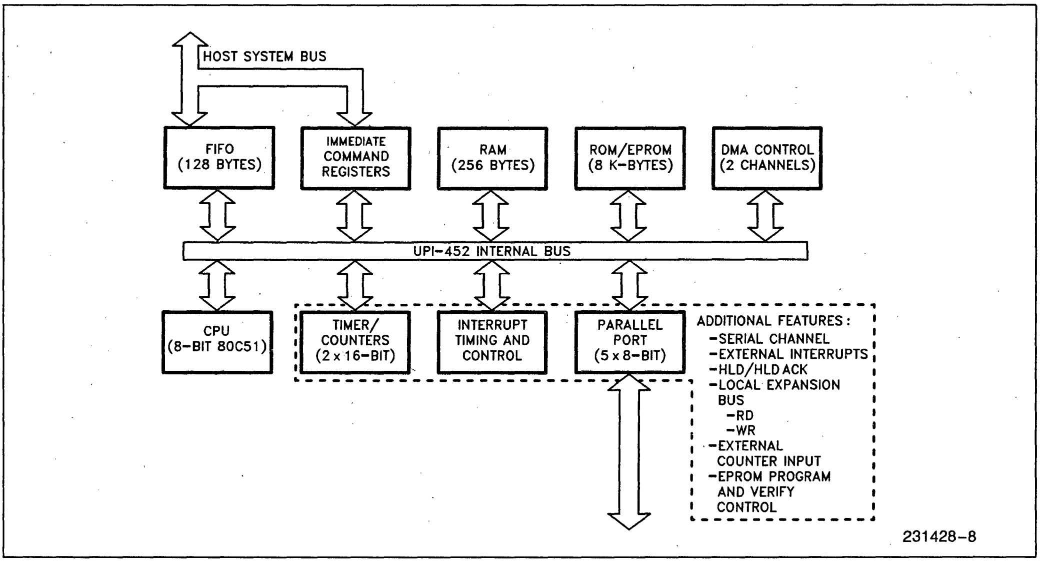


Figure 4. UPI-452 Functional Block Diagram

The division of the 128 bytes between Input and Output channels is user programmable allowing maximum flexibility. If the entire 128 byte FIFO is allocated to the Input channel, a high performance Host can transfer up to 128 bytes at one time, then dedicate its resources to other functions while the internal CPU processes the data in the FIFO. Various handshake signals allow the external Host to operate independently and without frequent monitoring of the UPI-452 internal CPU. The FIFO Buffer insures that the slave processor receives data in the same order that it was sent by the host without the need to keep track of addresses. Three slave bus interface handshake methods are supported by the UPI-452: DMA, Interrupt and Polled.

The FIFO is nine bits wide. The ninth bit acts as a command/data flag. Commands written to the FIFO by either the host or internal CPU are called Data Stream Commands or DSCs. DSCs are written to the input FIFO by the Host via a unique external address. DSCs are written to the output FIFO by the internal CPU via the COMMAND OUT Special Function Register (SFR). When encountered by the host or internal CPU a Data Stream Command can be used as an address vector to user defined service routines. DSCs provide synchronization of data and commands between the Host and internal CPU.

FIFO PROGRAMMABLE FEATURES

Size of Input/Output Channels

The 128 bytes of FIFO space can be allocated between the Input and Output channels via the Channel Boundary Pointer (CBP) SFR. This register contains the number of address locations assigned to the Input channel. The remaining address locations are automatically assigned to the Output FIFO. The CBP SFR can only be programmed by the internal CPU during FIFO DMA Freeze Mode (See FIFO-External Host Interface FIFO DMA Freeze Mode description). The CBP is initialized to 40H (64 bytes) upon reset.

The number in the Channel Boundary Pointer SFR is actually the first address location of the Output FIFO. Writing to the CBP SFR reassigns the Input and Output FIFO address space. Whenever the CBP is written, the Input FIFO pointers are reset to zero and the Output FIFO pointers are set to the value in the CBP SFR.

All of the FIFO space may be assigned to one channel. In such a situation the other channel's data path consists of a single SFR (FIFO IN/COMMAND IN or FIFO OUT/COMMAND OUT SFR) location.

CBP Register	Input FIFO Size	Output FIFO Size
0	1	128
1	1	128
2	2	126
3	3	125
4	4	124
•	•	•
7B	123	5
7C	124	4
7D	125	3
7E	128	1
7F	128	1

FIFO Read/Write Pointers

These normally operate in auto-increment (and auto-rollover) mode, but can be reassigned by the internal CPU during FIFO DMA Freeze Mode (See FIFO-External Host Interface FIFO DMA Freeze Mode description).

Threshold Register

The Input FIFO Threshold SFR contains the number of empty bytes that must be available in the Input FIFO to generate a Host interrupt. The Output FIFO Threshold SFR contains the number of bytes, data and/or DSC(s), that must be in the FIFO before an interrupt is generated. The Threshold feature prevents the Host from being interrupted each time the FIFO needs to load or unload one byte of data. The thresholds, therefore, allow the FIFO's operation to be adjusted to the speed of the Host, optimizing the overall interface performance.

Immediate Commands

The UPI-452 provides, in addition to data and DSCs, a third direct means of communication between the external Host and internal CPU called Immediate Commands. As the name implies, an Immediate Command is available to the receiving CPU immediately, via an interrupt, without being entered into the FIFO as are Data Stream Commands. Like Data Stream Commands, Immediate Commands are written either via a unique external address by the host CPU, or via dedicated SFR by the internal CPU.

The DSC and/or Immediate Command interface may be defined as either Interrupt or Polled under user program control via the Interrupt Enable (IE), Slave Control Register (SLCON), and Interrupt Enable Priority (IEP) Special Function Registers, for the internal CPU and via the Host Control SFR for the external Host CPU.

DMA

The UPI-452 contains a two channel internal DMA controller which allows transfer of data between any of the three writeable memory spaces: Internal Data Memory, External Load Expansion Bus Data Memory and the Special Function Register array. The Special Function Register array appears as a set of unique dedicated memory addresses which may be used as either the source or destination address of a DMA transfer. Each DMA channel is independently programmable via dedicated Special Function Registers for mode, source and destination addresses, and byte count to be transferred. Each DMA channel has four programmable modes:

— Alternate Cycle Mode

— Burst Mode

— FIFO or Serial Channel Demand Mode

— External Demand Mode

A complete description of each mode and DMA operation may be found in the section titled "General Purpose DMA Channels".

FIFO/SLAVE INTERFACE FUNCTIONAL DESCRIPTION

Overview

The FIFO is a 128 Byte RAM array with recirculating pointers to manage the read and write accesses. The FIFO consists of an Input and an Output channel. Access cycles to the FIFO by the internal CPU and external Host are interleaved and appear to be occurring concurrently to both the internal CPU and external Host. Interleaving access cycles ensures efficient use of this shared resource. The internal CPU accesses the FIFO in the same way it would access any of the Special Function Registers e.g., direct and register indirect addressing as well as arithmetic and logical instructions.

Input FIFO Channel

The Input FIFO Channel provides for data transfer from the external Host to the internal CPU (Figure 5). The registers associated with the Input Channel during normal operation are listed in Table 1*.

Table 1. Input FIFO Channel Registers*

	Register Name	Description
1)	Input Buffer Latch	Host CPU Write only
2)	FIFO IN SFR	Internal CPU Read only
3)	COMMAND IN SFR	Internal CPU Read only
4)	Input FIFO Read Pointer SFR	Internal CPU Read only
5)	Input FIFO Write Pointer SFR	Internal CPU Read only
6)	Input FIFO Threshold SFR	Internal CPU Read only

*See "FIFO-EXTERNAL HOST INTERFACE FIFO DMA FREEZE MODE" section for FIFO DMA Freeze Mode SFR characteristics description.

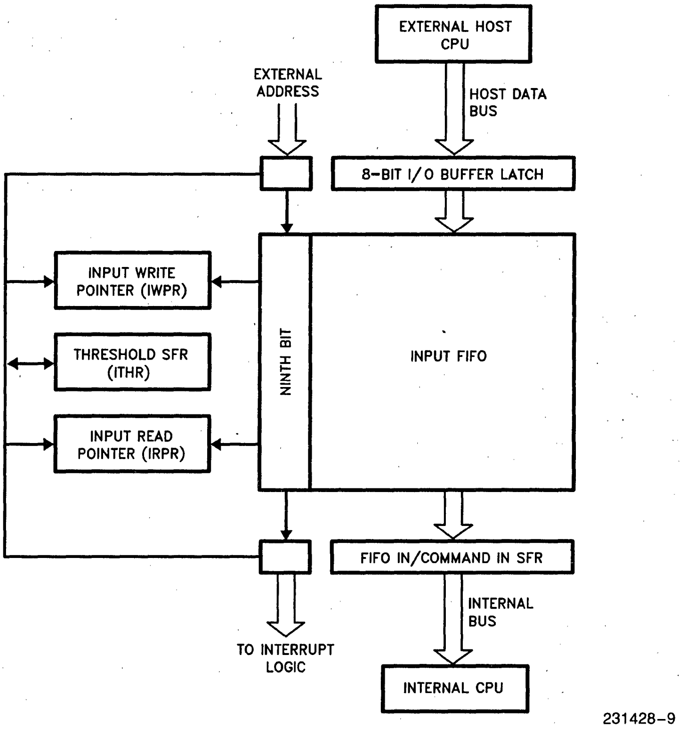


Figure 5. Input FIFO Channel Functional Block Diagram

The host CPU writes data and Data Stream Commands into the Input Buffer Latch on the rising edge of the external WR signal. External addressing determines whether the byte is a data byte or Data Stream Command and the FIFO logic sets the ninth bit of the FIFO accordingly as the byte is moved from the Input Buffer Latch into the FIFO. A "1" in the ninth bit indicates that the incoming byte is a Data Stream Command. The internal CPU reads data bytes via the FIFO IN SFR, and Data Stream Commands via the COMMAND IN SFR.

A Data Stream Command will generate an interrupt to the internal CPU prior to being read and after completion of the previous operation. The DSC can then be read via the COMMAND IN SFR. Data can only be read via the FIFO IN SFR and Data Stream Commands via the COMMAND IN SFR. Attempting to read Data Stream Commands as data by addressing the FIFO IN SFR will result in "0FFH" being read, and the Input FIFO Read Pointer will remain intact. (This prevents accidental misreading of Data Stream Commands.) Attempting to read data as Data Stream Commands will have the same consequence.

The Input FIFO Channel addressing is controlled by the Input FIFO Read and Write Pointer SFRs. These SFRs are read only registers during normal operation. However, during FIFO DMA Freeze Mode (See FIFO-External Host Interface FIFO DMA Freeze Mode description), the internal CPU has write access to them. Any write to these registers in normal mode will have no effect. The Input Write Pointer SFR contains the address location to which data/commands are written from the Input Buffer Latch. The write pointer is automatically incremented after each write and is reset to zero if equal to the CBP, as the Input FIFO operates as a circular buffer.

If a write is performed on an empty FIFO, the first byte is also written into the FIFO IN or COMMAND IN SFR. If the Host continues writing while the Input FIFO is full, an external interrupt, if enabled, is sent to the host to signal the overrun condition. The writes are ignored by the FIFO control logic. Similarly, an internal CPU read of an empty FIFO will cause an underrun error interrupt to be generated to the internal CPU and a value of "0FFH" will be read by the internal CPU.

The Read Pointer SFR holds the address of the next byte to be read from the Input FIFO. An Input FIFO read operation post-increments the Input Read Pointer SFR and loads a new data byte into the FIFO IN SFR or a Data Stream Command into the COMMAND IN SFR at the end of the read cycle.

An Input FIFO Request for Service (via DMA, Interrupt or a flag) is generated to the Host whenever more data can be written into the Input FIFO. For efficient utilization of the Host, a "threshold" value can be programmed into the Input FIFO Threshold SFR. The range of values of the Input FIFO Threshold SFR can be from 0 to (CBP-3). The Request for Service Interrupt is generated only after the Input FIFO has room to accommodate a threshold number of bytes or more. The threshold is equal to the total number of bytes assigned to the Input FIFO (CBP) minus the number of bytes programmed in the Input FIFO Threshold SFR. With this feature the Host is assured that it can write at least a threshold number of bytes to the Input FIFO channel without worrying about an overrun condition. Once the Request for Service is generated it remains active until the Input FIFO becomes full.

Output FIFO Channel

The Output FIFO Channel provides data transfer from the UPI-452 internal CPU to the external Host (Figure 6).

The registers associated with the Output Channel during normal operation are listed in Table 2*.

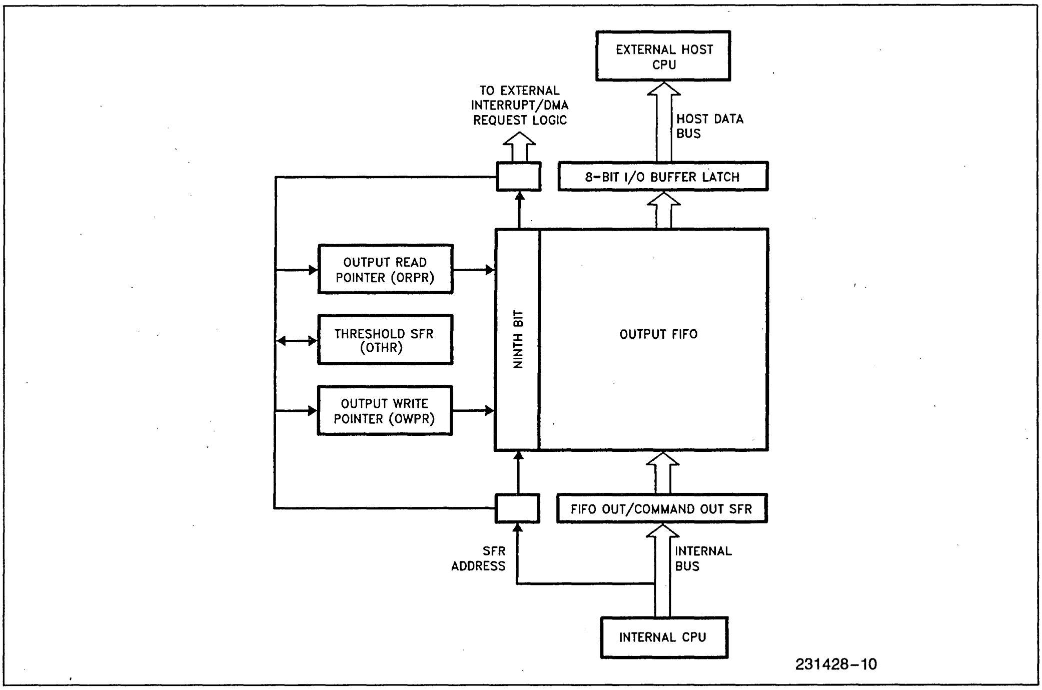


Figure 6. Output FIFO Channel Functional Block Diagram

Table 2. Output FIFO Channel Registers

	Register Name	Description
1)	Output Buffer Latch	Host CPU Read only
2)	FIFO OUT SFR	Internal CPU Read and Write
3)	COMMAND OUT SFR	Internal CPU Read and Write
4)	Output FIFO Read Pointer SFR	Internal CPU Read only
5)	Output FIFO Write Pointer SFR	Internal CPU Read only
6)	Output FIFO Threshold SFR	Internal CPU Read only

*See "FIFO-EXTERNAL HOST INTERFACE FIFO DMA FREEZE MODE" section for FIFO DMA Freeze Mode register characteristics description.

The UPI-452 internal CPU transfers data to the Output FIFO via the FIFO OUT SFR and commands via the COMMAND OUT SFR. If the byte is written to the COMMAND OUT SFR, the ninth bit is automatically set (=1) to indicate a Data Stream Command. If the byte is written to the FIFO OUT SFR the ninth bit is cleared (=0). Thus the FIFO OUT and COMMAND OUT SFRs are the same but the address determines whether the byte entered in the FIFO is a DSC or data byte.

The Output FIFO preloads a byte into the Output Buffer Latch. When the Host issues a RD/ signal, the data is immediately read from the Output Buffer Latch. The next data byte is then loaded into the Output Buffer Latch, a flag is set and an interrupt, if enabled, is generated if the byte is a DSC (ninth bit is set). The operation is carefully timed such that an interrupt can be generated in time for it to be recognized by the Host before its next read instruction. Internal CPU write and external Host read operations are interleaved at the FIFO so that they appear to be occurring concurrently.

The Output FIFO read and write pointer operation is the same as for the Input Channel. Writing to the FIFO OUT or COMMAND OUT SFRs will increment the Output Write Pointer SFR but reading from it will leave the write pointer unchanged. A rollover of the Output FIFO Write Pointer causes the pointer to be reset to the value in the Channel Boundary Pointer (CBP) SFR.

If the external host attempts to read a Data Stream Command as a data byte it will result in invalid data (0FFH) being read. The DSC is not lost because the invalid read does not increment the pointer. Similarly attempting to read a data byte as a Data Stream Command has the same result.

A Request for Service is generated to the external Host under the following two conditions:

1.) Whenever the internal CPU has written a threshold number of bytes or more into the Output FIFO (threshold = (OTHR) + 1). The threshold number should be chosen such that the bus latency time for the external Host does not result in a FIFO overrun error condition on the internal CPU side. The threshold limit should be large enough to make a bus request by the UPI-452 to the external host CPU worthwhile. Once a request for service is generated, the request remains active until the Output FIFO becomes empty. The range of values of the FIFO Output Threshold (OTHR) SFR is from 2 to {(80H-CBP)-1}. The threshold number can be programmed via the OTHR SFR.

2.) The second type of Request for Service is called "Flush Mode" and occurs when the internal CPU writes a Data Stream Command into the Output FIFO. Its purpose is to ensure that a data block entered into the Output FIFO, which is less than the programmed threshold, will generate a Request for Service interrupt, if enabled, and be read, or "Flushed" from the Output FIFO, by the external host CPU regardless of the status of the OTHR SFR.

Immediate Commands

Immediate Commands provide direct communication between the external Host and UPI-452. Unlike Data Stream Commands which are entered into the FIFO, the Immediate Command is available to the receiving CPU directly, bypassing the FIFO. The Immediate Command can serve as a program vector pointing into a jump table in the recipients software. Immediate Command Interrupts are generated, if enabled, and a bit in the appropriate Status Register is set when an Immediate Command is input or output. A similar bit is provided to acknowledge when an Immediate Command has been read and whether the register is available to receive another command. The bits are reset when the Immediate Commands are read. Two Special Function Registers are dedicated to the Immediate Command interface. External addressing determines whether the Host is accessing the Input FIFO or the Immediate Command IN (IMIN) SFR. The internal CPU writes Immediate Commands to the Immediate Command OUT (IMOUT) SFR.

Both processors have the ability to enable or disable Immediate Command Interrupts. By disabling the interrupt, the recipient of the Immediate Command can poll the status SFR and read the Immediate Command at its convenience. Immediate Commands should only be written when the appropriate Immediate Command SFR is empty (as indicated in the appropriate status SFR:HSTAT/SSTAT). Similarly, the Immediate Command SFR should only be read when there is data in the Register.

The flowcharts in Figure 7a and 7b illustrate the proper handshake mechanisms between the external Host and internal CPU when handling Immediate Commands.

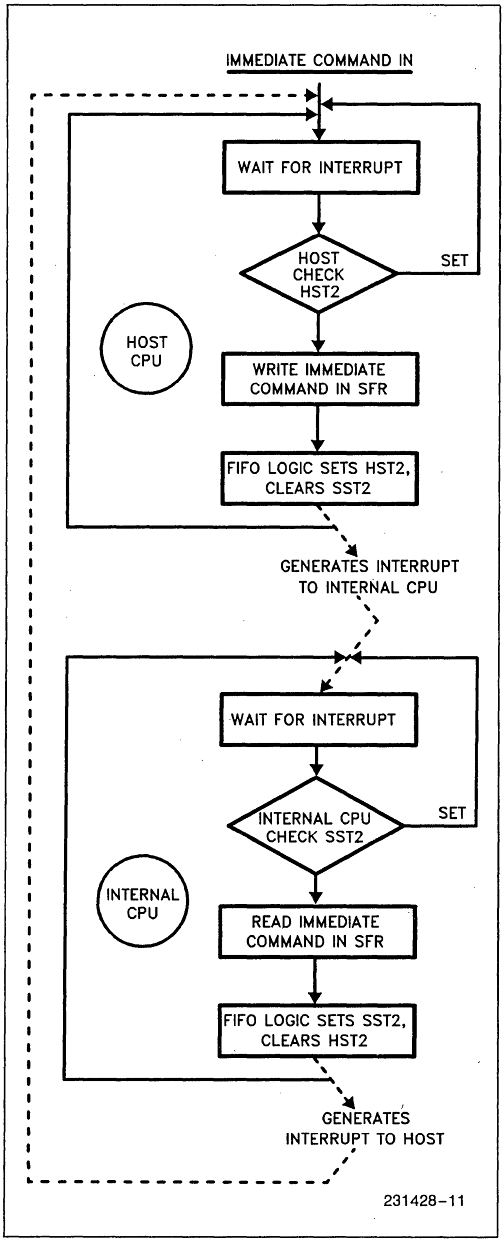

IMMEDIATE COMMAND IN
WAIT FOR INTERRUPT
HOST CHECK HST2
SET
HOST CPU
WRITE IMMEDIATE COMMAND IN SFR
FIFO LOGIC SETS HST2, CLEARS SST2
GENERATES INTERRUPT TO INTERNAL CPU
WAIT FOR INTERRUPT
INTERNAL CPU CHECK SST2
SET
INTERNAL CPU
READ IMMEDIATE COMMAND IN SFR
FIFO LOGIC SETS SST2, CLEARS HST2
GENERATES INTERRUPT TO HOST
231428-11

Figure 7a. Handshake Mechanisms for Handling Immediate Command IN Flowchart

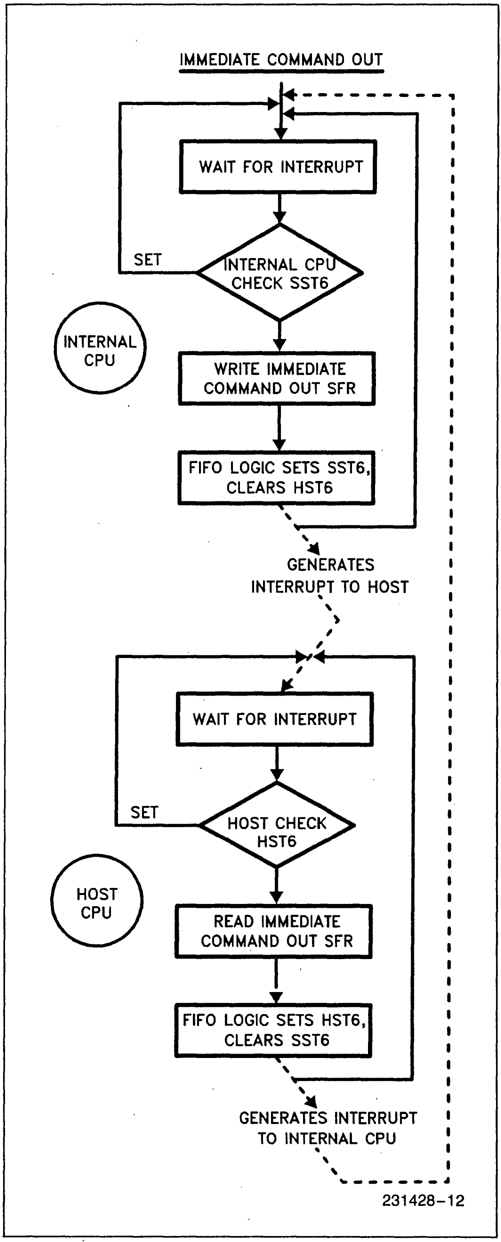

IMMEDIATE COMMAND OUT
WAIT FOR INTERRUPT
INTERNAL CPU CHECK SST6
SET
INTERNAL CPU
WRITE IMMEDIATE COMMAND OUT SFR
FIFO LOGIC SETS SST6, CLEARS HST6
GENERATES INTERRUPT TO HOST
WAIT FOR INTERRUPT
HOST CHECK HST6
SET
HOST CPU
READ IMMEDIATE COMMAND OUT SFR
FIFO LOGIC SETS HST6, CLEARS SST6
GENERATES INTERRUPT TO INTERNAL CPU
231428-12

Figure 7b. Handshake Mechanisms for Handling Immediate Command OUT Flowchart

HOST & SLAVE INTERFACE SPECIAL FUNCTION REGISTERS

Slave Interface Special Function Registers

The Internal CPU interfaces with the FIFO slave module via the following registers:

1) Mode Special Function Register (MODE)
2) Slave Control Special Function Register (SLCON)
3) Slave Status Special Function Register (SSTAT)

Each register resides in the SFR Array and is accessible via all direct addressing modes except bit. Only the Slave Control Register (SLCON) is bit addressable.

1) MODE Special Function Register (MODE)

The MODE SFR provides the primary control of the external host-FIFO interface. It is included in the SFR Array so that the internal CPU can configure the external host-FIFO interface should the user decide that the UPI-452 slave initialize itself independent of the external host CPU.

The MODE SFR can be directly modified by the internal CPU through direct address instructions. It can also be indirectly modified by the external host CPU by setting up a MODE SFR service routine in the UPI-452 program memory and having the host issue a Command, either Immediate or DSC, to vector to that routine.

Symbolic Address									Physical Address
MODE	—	MD6	MD5	MD4	—	—	—	—	0F9H
	(MSB)							(LSB)	
	Status On Reset:								
	1*	0	0	0	1*	1*	1*	1*	

MD7 (reserved)**

MD6 Request for Service to external CPU via;

1 = DMA (DRQIN/DRQOUT) request to external host when the Input or Output FIFO channel requests service

0 = Interrupt (INTRQIN/INTRQOUT or INTRQ) to external host when the Input or Output FIFO channel requests service or a DSC is encountered in the I/O Buffer Latch

MD5 Configure DRQIN/INTRQIN and DRQOUT/INTRQOUT to be either;

1 = Enable (Actively driven)

0 = Disable (Tri-state)

MD4 Configure INTRQ to be either;

1 = Enable (Actively driven)

0 = Disable (Tri-state)

MD3 (reserved) **

MD2 (reserved) **

MD1 (reserved) **

MD0 (reserved) **

2) Slave Control SFR (SLCON)

The Slave Control SFR is used to configure the FIFO-internal CPU interface. All interrupts are to the internal CPU.

Symbolic Address									Physical Address
SLCON	IFI	OFI	ICII	ICOI	FRZ	—	IFRS	OFRS	0E8H
	(MSB)							(LSB)	

Status On Reset:

0	0	0	0	0	1*	0	0

IFI Enable Input FIFO Interrupt (due to Underrun Error Condition, Data Stream Command or Request Service)

1 = Enable

0 = Disable

OFI Enable Output FIFO Interrupt (due to Overrun Error Condition or Request Service)

1 = Enable

0 = Disable

Note: If the DMA is configured to service a FIFO demand, then the Request for Service Interrupt is not generated.

ICII Generate Interrupt when a command is written to the Immediate Command in Register

1 = Enable

0 = Disable

ICOI Generate Interrupt when Immediate Command Out Register is Available

1 = Enable

0 = Disable

FRZ Enable FIFO DMA Freeze Mode

1 = Normal operation

0 = FIFO DMA Freeze Mode

SC2 (reserved) **

IFRS Input FIFO Channel Request for Service

1 = Request when Input FIFO not empty

0 = Request when Input FIFO full

OFRS Output FIFO Channel Request for Service

1 = Request when Output FIFO not full

0 = Channel Request when Output FIFO empty

NOTES:

*A '1' will be read from all SFR reserved locations except HCON SFR, HC0 and HC2.

**'reserved'—these locations are reserved for future use by Intel Corporation.

3) Slave Status SFR (SSTAT)

The bits in the Slave Status SFR reflect the status of the FIFO-internal CPU interface. It can be read during an internal interrupt service routine to determine the nature of the interrupt or read during a polling sequence to determine a course of action.

Symbolic Address									Physical Address
SSTAT	SST7	SST6	SST5	SST4	SST3	SST2	SST1	SST0	0E9H
	← Output FIFO Status →				← Input FIFO Status →				

Status On Reset:

1	0	0	0	1	1	1	1
(MSB)							(LSB)

SST7 Output FIFO Overrun Error Condition
1 = No Error
0 = Error (latched until Slave Status SFR is read)

SST6 Immediate Command Out Register Status
1 = Full (i.e. Host CPU has not read previous Immediate Command Out sent by internal CPU)
0 = Available

SST5 FIFO DMA Freeze Mode Status
1 = Normal Operation
0 = FIFO DMA Freeze Mode in Progress

SST4 Output FIFO Request for Service Flag
1 = Output FIFO does not request service
0 = Output FIFO requests service

SST3 Input FIFO Underrun Error Condition Flag
1 = No Underrun Error
0 = Underrun Error (latched until Slave Status SFR is read)

SST2 Immediate Command In SFR Status
1 = Empty
0 = Immediate Command received from host CPU

SST1 Data Stream Command/Data at Input FIFO Flag
1 = Data (not DSC)
0 = DSC (at COMMAND IN SFR)

SST0 Input FIFO Request For Service Flag
1 = Input FIFO Does Not Request Service
0 = Input FIFO Request for Service

EXTERNAL HOST INTERFACE SPECIAL FUNCTION REGISTERS

The external host CPU has direct access to the following SFRs:
1) Host Control Special Function Register
2) Host Status Special Function Register

It can also access other SFRs by commanding the internal CPU to change them accordingly via Data Stream Commands or Immediate Commands. The protocol for implementing this is entirely determined by the user.

1) Host Control SFR (HCON)

By writing to the Host Control SFR, the host can enable or disable FIFO interrupts and DMA requests and can reset the UPI-452.

Symbolic Address									**Physical Address**
HCON	HC7	HC6	HC5	HC4	HC3	—	HC1	—	0E7H
	(MSB)							(LSB)	
	Status On Reset:								
	0	0	0	0	0	0*	0	0*	

HC7 Enable Output FIFO Interrupt due to Underrun Error Condition, Data Stream Command or Service Request

1 = Enable

0 = Disable

HC6 Enable Input FIFO Interrupt due to Overrun Error Condition, or Service Request

1 = Enable

0 = Disable

HC5 Enable the generation of the Interrupt due to Immediate Command Out being present

1 = Enable

0 = Disable

HC4 Enable the Interrupt due to the Immediate Command In Register being Available for a new Immediate Command byte

1 = Enable

0 = Disable

HC3 Reset UPI-452

1 = Software RESET

0 = Normal Operation

HC2 (reserved) **

HC1 Select between INTRQ and INTRQIN/INTRQOUT as Request for Service interrupt signal when DMA is disabled

1 = INTRQ

0 = INTRQIN or INTRQOUT

HC0 (reserved) **

NOTES:

*A '1' will be read from all SFR reserved locations except HCON SFR, HC0 and HC2.

**'reserved'—these locations are reserved for future use by Intel Corporation.

2) Host Status SFR (HSTAT)

The Host Status SFR provides information on the FIFO-Host Interface and can be used to determine the source of an external interrupt during polling. Like the Slave Status SFR, the Host Status SFR reflects the current status of the FIFO-external host interface.

Symbolic Address									**Physical Address**
HSTAT	HST7	HST6	HST5	HST4	HST3	HST2	HST1	HST0	0E6H
	← Output FIFO Status →				← Input FIFO Status →				

Status On Reset:

1	1	1	1	1	1/0*	1	1
(MSB)							(LSB)

HST7 Output FIFO Underrun Error Condition
1 = No Underrun Error
0 = Underrun Error (latched until Host Status Register is read)

HST6 Immediate Command Out SFR Status
1 = Empty
0 = Immediate Command Present

HST5 Data Stream Command/Data at Output FIFO Status
1 = Data (not DSC)
0 = DSC (present at Output FIFO COMMAND OUT SFR)
(Note: Only if HST4 = 0, if HST4 = 1 then undetermined)

HST4 Output FIFO Request for Service Status
1 = No Request for Service
0 = Output FIFO Request for Service due to:

a. Output FIFO containing the threshold number of bytes or more

b. Internal CPU sending a block of data terminated by a DSC (DSC Flush Mode)

HST3 Input FIFO Overrun Error Condition
1 = No Overrun Error
0 = Overrun Error (latched until Host Status Register is read)

HST2 Immediate Command In SFR Status
1 = Full (i.e. Internal CPU has not read previous Immediate Command sent by Host)
0 = Empty

* Reset value;

'1' — if read by the external Host

'0' — if read by internal CPU (reads shadow latch - see FIFO DMA Freeze Mode description)

HST1 FIFO DMA Freeze Mode Status
1 = Freeze Mode in progress.
(In Freeze Mode, the bits of the Host Status SFR are forced to a '1' initially to prevent the external Host from attempting to access the FIFO. The definition of the Host Status SFR bits during FIFO DMA Freeze Mode can be found in FIFO DMA Freeze Mode description)
0 = Normal Operation

HST0 Input FIFO Request Service Status
1 = Input FIFO does not request service
0 = Input FIFO request service due to the Input FIFO containing enough space for the host to write the threshold number of bytes or more

FIFO MODULE - EXTERNAL HOST INTERFACE

Overview

The FIFO-external Host interface supports high speed asynchronous bi-directional 8-bit data transfers. The host interface is fully compatible with Intel microprocessor local busses and with MULTIBUS. The FIFO has two specialized DMA request pins for Input and Output FIFO channel DMA requests. These are multiplexed to provide a dedicated Request for Service interrupt (DRQIN/INTRQIN, DRQOUT/INTRQOUT).

The external Host can program, under user defined protocol, thresholds into the FIFO Input and Output Threshold SFRs which determine when the FIFO Request for Service interrupt is generated to the Host CPU. The FIFO module external Host interface is configured by the internal CPU via the MODE SFR. "The external Host can enable and disable Host interface interrupts via the Host Control SFR." Data Stream Commands in the Input FIFO channel allow the Host to influence the processing of data blocks and are sent with the data flow to maintain synchronization. Data Stream Commands in the Output FIFO Channel allow the internal CPU to perform the same function, and also to set the Output FIFO Request Service status logic to the host CPU regardless of the programmed value in the Threshold SFR.

Slave Interface Address Decoding

The UPI-452 determines the desired Host function through address decoding. The lower three bits of the address as well as the READ, $\overline{\text{WRITE}}$, Chip Select ($\overline{\text{CS}}$) and DMA Acknowledge ($\overline{\text{DACK}}$) are used for decoding. Table 3 shows the pin states and the Read or Write operations associated with each configuration.

Interrupts to the Host

The UPI-452 interrupts the external Host via the INTRQ pin. In addition, the DRQIN and DRQOUT pins can be multiplexed as interrupt request lines, INTRQIN and INTRQOUT respectively, when DMA is disabled. This provides two special FIFO "Request for Service" interrupts.

There are eight FIFO-related interrupt sources; two from The Input FIFO; three from The Output FIFO; one from the Immediate Command Out SFR; one from the Immediate Command IN SFR; and one due to FIFO DMA Freeze Mode.

INPUT FIFO: The Input FIFO interrupt is generated whenever:

a. The Input FIFO contains space for a threshold number of bytes.

Table 3. UPI-452 Address Decoding

DACK	CS	A2	A1	A0	Read	Write
1	1	X	X	X	No Operation	No Operation
1	0	0	0	0	Data or DMA from Output FIFO Channel	Data or DMA to Input FIFO Channel
1	0	0	0	1	Data Stream Command from Output FIFO Channel	Data Stream Command to Input FIFO Channel
1	0	0	1	0	Host Status SFR Read	Reserved
1	0	0	1	1	Host Control SFR Read	Host Control SFR Write
1	0	1	0	0	Immediate Command SFR Read	Immediate Command to SFR Write
1	0	1	1	X	Reserved	Reserved
0	X	X	X	X	DMA Data from Output FIFO Channel	DMA Data to Input FIFO Channel
1	0	1	0	1	Reserved	Reserved

NOTES:

1. Attempting to read a DSC as a data byte will result in invalid data being read. The read pointers are not incremented so that the DSC is not lost. Attempting to read a data byte as a DSC has the same result.
2. If DACK is active the UPI-452 will attempt a DMA operation when RD or WR becomes active regardless of the DMA enable bit (MD6) in the MODE SFR. Care should be taken when using DACK. For proper operation, DACK must be driven high (+5V) when not using DMA.

b. When an Input FIFO overrun error condition exists. The appropriate bits in the Host Status SFR are set and the interrupt is generated only if enabled.

OUTPUT FIFO: The Output FIFO Request for Service Interrupt operates in a similar manner as the Input FIFO interrupt:

a. When the FIFO contains the threshold number of bytes or more.

b. Output FIFO error condition interrupts are generated when the Output FIFO is underrun.

c. Data Stream Command present in the Output Buffer Latch.

A Data Stream Command interrupt is used to halt normal processing, using the command as a vector to a service routine. When DMA is disabled, the user may program (through HC1) INTRQ to include FIFO Request for Service Interrupts or use INTRQIN and INTRQOUT as Request for Service Interrupts.

IMMEDIATE COMMAND INTERRUPTS:

a. An Immediate Command Out Interrupt is generated, if enabled, to the Host and the corresponding Host Status SFR bit (HSTAT HST6) is cleared, when the internal CPU writes to the Immediate Command OUT (IMOUT) SFR. When the Host reads the Immediate Command OUT (IMOUT) SFR the corresponding bit in the Host Status (HSTAT) SFR is set. This causes the Slave Status Immediate Command OUT Status bit (SSTAT SST6) to be cleared indicating that the Immediate Command OUT (IMOUT) SFR is empty. If enabled, a FIFO-Slave Interface will also be generated to the internal CPU. (See Figure 7b, Immediate Command OUT Flowchart.)

b. An Immediate Command IN interrupt is generated, if enabled, to the Host when the internal CPU has read a byte from the Immediate Command IN (IMIN) SFR. The read operation clears the Host Status SFR Immediate Command IN Status bit (HSTAT HST2) indicating that the Immediate Command IN SFR is empty. The corresponding Slave Status (SSTAT) SFR bit is also set to indicate an empty status. Setting the Slave Status SFR bit generates a FIFO-Slave Interface interrupt, if enabled, to the internal CPU. (See Figure 7a, Immediate Command IN Flowchart.)

NOTE:

Immediate Command IN and OUT interrupts are actually specific Request For Service interrupts to the Host.

FIFO DMA FREEZE MODE: When the internal CPU invokes FIFO DMA Freeze Mode, for example at reset or to reconfigure the FIFO interface, INTRQ is activated. The INTRQ can only be deactivated by the external Host reading the Host Status SFR (HST1 remains active until FIFO DMA Freeze Mode is disabled by the internal CPU).

Once an interrupt is generated, INTRQ will remain high until no interrupt generating condition exists. For a FIFO underrun/overrun error interrupt, the interrupt condition is deactivated by the external Host reading the Host Status SFR. An interrupt is serviced by reading the Host Status SFR to determine the source of the interrupt and vectoring the appropriate service routine.

DMA Requests to the Host

The UPI-452 generates two DMA requests, DRQIN and DRQOUT, to facilitate data transfer between the Host and the Input and Output FIFO channels. A DMA acknowledge, $\overline{\text{DACK}}$, is used as a chip select and initiates a data transfer. The external $\overline{\text{READ}}$ and $\overline{\text{WRITE}}$ signals select the Input and Output FIFO respectively. The $\overline{\text{CS}}$ and address lines can also be used as a DMA acknowledge for processors with onboard DMA controllers which do not generate a $\overline{\text{DACK}}$ signal.

The internal CPU can configure the UPI-452 to request service from the external host via DMA or interrupts by programming Mode SFR MD6 bit. In addition the external Host enables DMA requests through bits 6 and 7 of the Host Control SFR. When a DMA request is invoked the number of bytes transferred to the Input FIFO is the total number of bytes in the Input FIFO (as determined by the CBP SFR) minus the value programmed in the Input FIFO Threshold SFR. The DMA request line is activated only when the Input FIFO has a threshold number of bytes that can be transferred.

The Output FIFO DMA request is activated when a DSC is written by the internal CPU at the end of a less than threshold size block of data (Flush Mode) or when the Output FIFO threshold is reached. The request remains active until the Input FIFO becomes full or the Output FIFO becomes empty. If a DSC is encountered during an Output FIFO DMA transfer, the DMA request is dropped until the DSC is read. The DMA request will be reactivated after the DSC is read and remains active until the Output FIFO becomes empty or another DSC is encountered.

FIFO MODULE - INTERNAL CPU INTERFACE

Overview

The Input and Output FIFOs are accessed by the internal CPU through direct addressing of the FIFO IN/COMMAND IN and FIFO OUT/COMMAND OUT Special Function Registers. All of the 80C51 instructions involving direct addressing may be used to access the FIFO's SFRs. The FIFO IN, COMMAND IN and Immediate Command In SFRs are actually read only registers, and their Output counterparts are write only. Internal DMA transfers data between Internal memory, External Memory and the Special Function Registers. The Special Function Registers appear as another group of dedicated memory addresses and are programmed as the source or destination via the DMA0/DMA1 Source Address or Destination Address Special Function Registers. The FIFO module manages the transfer of data between the external host and FIFO SFRs.

Internal CPU Access to FIFO Via Software Instructions

The internal CPU has access to the Input and Output FIFOs via the FIFO IN/COMMAND IN and FIFO OUT/COMMAND OUT SFRs which reside in the Special Function Register Array. At the end of every instruction that involves a read of the FIFO IN/COMMAND IN SFR, the SFR is written over by a new byte from the Input FIFO channel when available. At the end of every instruction that involves a write to the FIFO OUT/COMMAND OUT SFR, the new byte is written into the Output FIFO channel and the write pointer is incremented after the write operation (post incremented).

The internal CPU reads the Input FIFO by using the FIFO IN/COMMAND IN SFR as the source register in an instruction. Those instructions which read the Input FIFO are listed below:

```
ADD  A,FIFO IN/COMMAND IN
ADDC A,FIFO IN/COMMAND IN
PUSH FIFO IN/COMMAND IN
ANL  A,FIFO IN/COMMAND IN
ORL  A,FIFO IN/COMMAND IN
XRL  A,FIFO IN/COMMAND IN
CJNE A,FIFO IN/COMMAND IN, rel
SUBB A,FIFO IN/COMMAND IN
MOV  direct,FIFO IN/COMMAND IN
MOV  @Ri,FIFO IN/COMMAND IN
MOV  Rn,FIFO IN/COMMAND IN
MOV  A,FIFO IN/COMMAND IN
```

After each access to these registers, they are overwritten by a new byte from the FIFO.

NOTE:
Instructions which use the FIFO IN or COMMAND IN SFR as both a source and destination register will have the data destroyed as the next data byte is rewritten into the FIFO IN register at the end of the instruction. These instructions are not supported by the UPI-452 FIFO. Data can only be read through the FIFO IN SFR and DSCs through the COMMAND IN SFR. Data read through the COMMAND IN SFR will be read as 0FFH, and DSCs read through the FIFO IN SFR will be read as OFFH. The Immediate Command in SFR is read with the same instructions as the FIFO IN and COMMAND IN SFRs.

The FIFO IN, COMMAND IN and Immediate Command In SFRs are read only registers. Any write operation performed on these registers will be ignored and the FIFO pointers will remain intact.

The internal CPU uses the FIFO OUT SFR to write to the Output FIFO and any instruction which uses the FIFO OUT or COMMAND OUT SFR as a destination will invoke a FIFO write. DSCs are differentiated from data by writing to the COMMAND OUT SFR. In the FIFO, Data Stream Commands have the ninth bit associated with the command byte set to "1". The instructions used to write to the Output FIFO are listed below:

```
MOV FIFO OUT/COMMOUT, A
MOV FIFO OUT/COMMOUT, direct
MOV FIFO OUT/COMMOUT, Rn
POP FIFO OUT/COMMOUT
MOV FIFO OUT/COMMOUT, #data
MOV FIFO OUT/COMMOUNT, @Ri
```

NOTE:
Instructions which use the FIFO OUT/COMMAND OUT SFRs as both a source and destination register cause invalid data to be written into the Output FIFO. These instructions are not supported by the UPI-452 FIFO.

GENERAL PURPOSE DMA CHANNELS

Overview

There are two identical General Purpose DMA Channels on the UPI-452 which allow high speed data transfer from one writeable memory space to another. As many as 64K bytes can be transferred in a single DMA operation. The following memory spaces can be used with DMA channels:

- Internal Data Memory
- External Data Memory
- Special Function Registers

The Special Function Register array appears as a limited group of dedicated memory addresses. The Special Function Registers may be used in DMA transfer operations by specifying the SFR as the source or destination address. The Special Function Registers which may be used in DMA transfers are listed in Table 4. Table 4 also shows whether the SFR may be used as Source or Destination only, or both.

The FIFO can be accessed during DMA by using the FIFO IN SFR as the DMA Source Address Register (SAR) or the FIFO OUT SFR as the Destination Address Register (DAR). (Note: Since the FIFO IN SFR is a read only register, the DMA transfer will be ignored if it is used as a DMA DAR. This is also true if the FIFO OUT SFR is used as a DMA SAR.)

Each DMA channel is software programmable to operate in either Block Mode or Demand Mode. In the Block Mode, DMA transfers can be further programmed to take place in Burst Mode or Alternate Cycle mode. In Burst Mode, the processor halts its execution and dedicates its resources to the DMA transfer. In Alternate Cycle Mode, DMA cycles and instruction cycles occur alternately.

In Demand Mode, a DMA transfer occurs only when it is demanded. Demands can be accepted from an external device (through External Interrupt pins, EXT0/EXT1) or from either the Serial Channel or FIFO flags. In this way, a DMA transfer can be synchronized to an external device, the FIFO or the Serial Port. If the External Interrupt is configured in Edge Mode, a single byte transfer occurs per transition. The external interrupt itself will occur if enabled. If the External Interrupt is configured in Level Mode, DMA transfers continue until the External Interrupt request goes inactive or the byte count becomes zero. The following flags activate Demand Mode transfers of one byte to/from the FIFO or Serial Channel:

RI - Serial Channel Receiver Buffer Full

TI - Serial Channel Transmitter Buffer Empty

Architecture

There are three 16 bit and one 8 bit Special Function Registers associated with each DMA channel.

- The 16 bit Source Address SFR (SAR) points to the source byte.
- The 16 bit Destination Address SFR (DAR) points to the destination.
- The 16 bit Byte Count SFR (BCR) contains the number of bytes to be transferred and is decremented when a byte transfer is accomplished.
- The DMA Control SFR (DCON) is eight bits wide and specifies the source memory space, destination memory space and the mode of operation.

In Auto Increment mode, the Source Address and/or Destination Address is incremented when a byte is transferred. When a DMA transfer is complete (BCR = 0), the DONE bit is set and a maskable interrupt is generated. The GO bit must be set to start any DMA transfer (also, the Slave Control SFR FRZ bit must be set to disable FIFO DMA Freeze Mode). The two DMA channels are designated as DMA0 and DMA1, and their corresponding registers are suffixed by 0 or 1; e.g. SAR0, DAR1, etc.

Table 4. DMA Accessible Special Function Registers

SFR	Symbol	Address	Source Only	Destination Only	Either
Accumulator	A/ACC	0E0H			Y
B Register	B	0F0H			Y
FIFO IN	FIN	0EEH	Y		
COMMAND IN	CIN	0EFH	Y		
FIFO OUT	FOUT	0FEH		Y	
COMMAND OUT	COUT	0FFH		Y	
Serial Data Buffer	SBUF	099H			Y
Port 0	P0	080H			Y
Port 1	P1	090H			Y
Port 2	P2	0A0H			Y
Port 3	P3	0B0H			Y
Port 4	P4	0C0H			Y

DMA Special Function Registers

DMA Control SFR: DCON0, DCON1

Symbolic Address									Physical Address
DCON0	DAS	IDA	SAS	ISA	DM	TM	DONE	GO	092H
DCON1	DAS	IDA	SAS	ISA	DM	TM	DONE	GO	093H
	(MSB)							(LSB)	

Reset Status: DCON0 and DCON1 = 00H

Bit Definition:

DAS	IDA	Destination Address Space
0	0	External Data Memory without Auto-Increment
0	1	External Data Memory with Auto-Increment
1	0	Special Function Register
1	1	Internal Data Memory

SAS	ISA	Source Address Space
0	0	External Data Memory without Auto-Increment
0	1	External Data Memory with Auto-Increment
1	0	Special Function Register
1	1	Internal Data Memory

DM	TM	DMA Transfer Mode
0	0	Alternate-Cycle Transfer Mode
0	1	Burst Transfer Mode
1	0	FIFO or Serial Channel Demand Mode
1	1	External Demand Mode

DONE DMA transfer Flag:

0 DMA transfer is not completed.

1 DMA transfer is complete.

NOTE:
This flag is set when contents of the Byte Count SFR decrements to zero. It is reset automatically when the DMA vectors to its interrupt routine.

GO Enable DMA Transfer:

0 Disable DMA transfer (in all modes).

1 Enable DMA transfer. If the DMA is in the Block mode, start DMA transfer if possible. If it is in the Demand mode, enable the channel and wait for a demand.

NOTE:
The GO bit is reset when the BCR decrements to zero.

DMA Transfer Modes

The following four modes of DMA operation are possible in the UPI-452.

1. ALTERNATE-CYCLE MODE

General

Alternate cycle mode is useful when CPU processing must occur during the DMA transfers. In this mode, a DMA cycle and an instruction cycle occur alternately. The interrupt request is generated (if enabled) at the end of the process, i.e. when BCR decrements to zero. The transfer is initiated by setting the GO bit in the DCON SFR.

Alternate-Cycle FIFO Demand Mode

Alternate cycle demand mode is useful for FIFO transfers of a less urgent nature. As mentioned before, CPU instruction cycles are interleaved with DMA transfer cycles, allowing true parallel processing.

This mode differs from FIFO Demand Mode in that CPU instruction cycles must be interleaved with DMA transfers, even if the FIFO is demanding DMA. In FIFO Demand Mode, CPU cycles would never occur if the FIFO demand was present.

Input Channel

The DMA is configured as in FIFO Demand Mode and transfers are initiated whenever an Input FIFO service request is generated. DMA transfer cycles are alternated with instruction execution cycles. DMA transfers are terminated as in FIFO Demand Mode.

Output Channel

The DMA is configured as in FIFO Demand Mode and transfers are initiated whenever an Output FIFO requests service. DMA transfer cycles are alternated with instruction execution cycles. DMA transfers are terminated as in FIFO Demand Mode.

The FIFO logic resets the interrupt flag after transferring the byte, so the interrupt is never generated.

Once the DMA is programmed to service the FIFO, the request for service interrupt for the FIFO is inhibited until the DMA is done (BCR = 0).

2. BURST MODE

In BURST mode the DMA is initiated by setting the GO bit in the DCON SFR. The DMA operation continues until BCR decrements to zero (zero byte count), then an interrupt is generated (if enabled). No interrupts are recognized during this DMA operation once it has started.

Input Channel

The FIFO Input Channel can be used in burst mode by specifying the FIFO IN SFR as the DMA Source Address. DMA transfers begin when the GO bit in the DMA Control SFR is set. The number of bytes to be transferred must be specified in the Byte Count SFR (BCR) and auto-incrementing of the SAR must be disabled. Once the GO bit is set nothing can interrupt the transfer of data until the BCR is zero. In this mode, a Data Stream Command encountered in the FIFO will be held in the COMMAND IN SFR with the pointers frozen, and invalid data (FFH) will be read through the FIFO IN SFR. If the input FIFO becomes empty during the block transfer, an 0FFH will be read until BCR decrements to zero.

Output Channel

The Output FIFO Channel can be used in burst mode by specifying the FIFO OUT or COMMAND OUT SFR as the DMA Destination Address. DMA transfers begin when the GO bit is set. This mode can be used to send a block of data or a block of Data Stream Commands. If the FIFO becomes full during the block transfer, the remaining data will be lost.

NOTE:
All interrupts including FIFO interrupts are not recognized in Burst Mode. Burst Mode transfers should be used to service the FIFO only when the user is certain that no Data Stream Commands are in the block to be transferred (Input FIFO) and that the FIFO contains enough space to store the block to be transferred. In all other cases Alternate Cycle or Demand Mode should be used.

3. FIFO AND SERIAL CHANNEL DEMAND MODES

NOTES:
1. If the output FIFO is configured as a one byte buffer and the user program consists of two-cycle instructions only, then Alternate-Cycle Mode should be used.
2. In non-auto increment mode for internal to external, or external to internal transfers, the lower 8 bits of the external address should not correspond to the FIFO or Serial Port address.

FIFO Demand Mode

Although any DMA mode is possible using the FIFO buffer, only FIFO Demand and Alternate Cycle FIFO Demand Modes are recommended. FIFO Demand Mode DMA transfers using the input FIFO Channel are set-up by setting the GO bit and specifying the FIFO IN register as the DMA Source Address Register. The BCR should be set to the maximum number of expected transfers. The user must also program bit 1 of the Slave Control Register (SC1) to determine whether the Slave Status (SSTAT) SFR FIFO Request For Service Flag will be activated when the FIFO becomes not empty or full. Once the Request For Service Flag is activated by the FIFO, the DMA transfer begins, and continues until the request flag is deactivated. While the request is active, nothing can interrupt the DMA (i.e. it behaves like burst mode). The DMA Request is held active until one of the following occurs:

1) The FIFO becomes empty.

2) A Data Stream Command is encountered (this generates a FIFO interrupt and DMA operation resumes after the Data Stream Command is read).

3) BCR = 0 (this generates a DMA interrupt and sets the DONE bit).

DMA transfers to the Output FIFO Channel are similar. The FIFO OUT or COMMAND OUT SFR is the DMA Destination Address SFR and a transfer is started by setting the GO bit. The user programs bit 0 of the Slave Control SFR (SC0) to determine whether a demand occurs when the Output FIFO is not full or empty. DMA transfers begin when the Request For Service Flag is activated by the FIFO logic and continue as long as the flag is active. The Flag remains active until one of the following occurs:

1) The FIFO becomes full

2) BCR = 0 (this generates a DMA interrupt and sets the DONE bit).

As in Alternate Cycle FIFO Demand Mode, the FIFO logic resets the interrupt flag after transferring the byte, so the interrupt is never generated.

After the GO bit is set, the DMA is activated if one of the following conditions takes place:
SAR(0/1) = FIFO IN and HIFRS flag is set
DAR(0/1) = FIFO OUT and HOFRS flag is set

The HIFRS and HOFRS signals are internal flags which are not accessible by software. These flags are similar to the SST0 and SST4 flags in the Slave Status Register except that they are of the opposite polarity and once set they are not cleared until the Input FIFO becomes empty (HIFRS) or the Output FIFO becomes full (HOFRS).

Serial Channel Demand Mode

Serial Channel Demand Mode is the logical choice when using the Serial Port. The DMAs can be activated by one of the Serial Channel Flags. Receiver interrupt (RI) or Transmitter Interrupt (TI).

SAR(0/1) = SBUF and RI flag is set
DAR(0/1) = SBUF and TI flag is set

NOTE:
TI flag must be set by software to initiate the first transfer.

When the DMA transfer begins, only one byte is transferred at a time. The serial port hardware automatically resets the flag after completion of the transfer, so an interrupt will not be generated unless DMA servicing is held off due to the DMA being done (BCR = 0) or when the Hold/Hold Acknowledge logic is used and the DMA does not own the bus. In this case a Serial Port interrupt may be generated if enabled because of the status of the RI or TI flags.

In FIFO demand mode, Alternate cycle FIFO demand mode or Serial Port demand mode only one of the following registers (SBUF, FIN or FOUT) should be used as either the SAR or DAR registers to prevent undesired transfers. For example if SAR0 = FIN and DAR0 = SBUF in demand mode, the DMA transfer will start if either the HIFRS or TI flags are set.

4. EXTERNAL DEMAND MODE

The DMA can be initiated by an external device via External interrupt 0 and 1 (INT0/INT1) pins. The INT0 pin demands DMA0 (Channel 0) and INT1 demands DMA1 (Channel 1). If the interrupts are configured in edge mode, a single byte transfer is accomplished for every request. Interrupts also result (INT0 and INT1) after every byte transfer (if enabled). If the interrupts are configured in level mode, the DMA transfer continues until the request goes inactive or BCR = 0. In either case, a DMA interrupt is generated (if enabled) when BCR = 0. The GO bit must be set for the transfer to begin.

EXTERNAL MEMORY DMA

When transferring data to or from external memory via DMA, the HOLD (HLD) and HOLD-ACKNOWLEDGE (HLDA) signals are used for handshaking. The HOLD and HOLD-ACKNOWLEDGE are active low signals which arbitrate control of the local bus. The UPI-452 can be used in a system where multi-masters are connected to a single parallel Address/Data bus. The HLD/HLDA signals are used to share resources (memory, peripherals, etc.) among all the processors on the local bus. The UPI-452 can be configured in any of three different External Memory Modes controlled by bits 5 and 6 (REQ & ARB) in the PCON SFR (Table 5). Each mode is described below:

REQUESTER MODE: In this mode, the UPI-452 is not the bus master, but must request the bus from another device. The UPI-452 configures port pin P1.5 as a HLD output and pin P1.6 as a HLDA input. The UPI-452 issues a HLD signal when it needs external access for a DMA channel. It uses the local bus after receiving the HLDA signal from the bus master, and will not release the bus until its DMA operation is complete.

ARBITER MODE: In this mode, the UPI-452 is the bus master. It configures port pin P1.5 as HLD input and pin P1.6 as HLDA output. When a device asserts the HLD signal to use the local bus, the UPI-452 asserts the HLDA signal after current instruction execution is complete. If the UPI-452 needs an external access via a DMA channel, it waits until the requester releases the bus, HLD goes inactive.

DISABLE MODE: When external program memory is accessed by an instruction or by program counter overflow beyond the internal ROM address or external data memory is accessed by MOVX instructions, it is a local memory access and the HLD/HLDA logic is not initiated. When a DMA channel attempts data transfer to/from the external data memory, the HLD/HLDA logic is initiated as described below. DMA transfers from the internal memory space to the internal memory space does not initiate the HLD/HLDA logic.

The balance of the PCON SFR bits are described in the "80C51 Register Description: Power Control SFR" section below.

Latency

When the GO bit is set, the UPI-452 finishes the current instruction before starting the DMA operation. Thus the maximum latency is 3.5 microseconds (at 14 MHz).

DMA Interrupt Vectors

Each DMA channel has a unique vectored interrupt associated with it. There are two vectored interrupts associated with the two DMA channels. The DMA interrupts are enabled and priorities set via the Interrupt Enable and Priority SFR (see "Interrupts" section). The interrupt priority scheme is similar to the scheme in 80C51.

Table 5. DMA MODE CONTROL - PCON SFR

Symbolic Address									Physical Address
PCON	—*	ARB	REQ	—*	—*	—*	—*	—*	87H
	(MSB)							(LSB)	

*Defined as per MLS-51 Data Sheet
Reset Status: 00H

Definition:

ARB	REQ	
0	0	HLD/HLDA logic is disabled.
0	1	The UPI-452 is in the Requester Mode.
1	0	The UPI-452 is in the Arbiter Mode.
1	1	Invalid

When a DMA operation is complete (BCR decrements to zero), the DONE flag in the respective DCON (DCON0 or DCON1) SFR is set. If the DMA interrupt is enabled, the DONE flag is reset automatically upon vectoring to the interrupt routine.

Interrupts When DMA is Active

If a Burst Mode DMA transfer is in progress, the interrupts are not serviced until the DMA transfer is complete. This is also true for level activated External Demand DMA transfers. During Alternate Cycle DMA transfers, however, the interrupts are serviced at the end of the DMA cycle. After that, DMA cycles and instruction execution cycles occur alternately. In the case of edge activated External Demand Mode DMA transfers, the interrupt is serviced at the end of DMA transfer of that single byte.

DMA Arbitration

Only one of the two DMA channels is active at a time, except when both are configured in the Alternate Cycle mode. In this case, the DMA cycles and Instruction Execution cycles occur in the following order:

1. DMA Cycle 0.
2. Instruction execution.
3. DMA Cycle 1.
4. Instruction execution.

DMA0 has priority over DMA1 during simultaneous activation of the two DMA channels. If one DMA channel is active, the other DMA channel, if activated, waits until the first one is complete.

If DMA0 is already in the Alternate Cycle mode and DMA1 is activated in Alternate Cycle Mode, it will take two instruction cycles before DMA1 is activated (due to the priority of DMA0). Once DMA1 becomes active, the execution will follow the normal sequence.

If DMA0 is already in the Alternate Cycle mode and DMA1 is activated in Burst Mode, the DMA1 Burst transfer will follow the DMA0 Alternate Cycle transfer (after the completion of the next instruction).

If the UPI-452 (as a Requester) asserts a HLD signal to request a DMA transfer (see "External Memory DMA")and its other DMA Channel requests a transfer before the HLDA signal is received, the channel having higher priority is activated first. A Burst Mode transfer on channel 0 can not be interrupted since DMA0 has the highest priority. A Demand Mode transfer on channel 0 is the only type of activity that can interrupt a block transfer on DMA1.

If, while executing a DMA transfer, the Arbiter receives a HLD signal, and then before it can acknowledge, its other DMA Channel requests a transfer, it then completes the second DMA transfer before sending the HLDA signal to release the bus to the HLD request.

DMA transfers may be held off under the following conditions:

1. A write to any of the DMA registers inhibits the DMA for one instruction cycle.

NOTE:
An instruction cycle may be executed in 1, 2 or 4 machine cycles dependent on the instruction being executed. DMA transfers are only executed after the completion of an instruction cycle never between machine cycles of a single instruction cycle. Similarly instruction cycles are only executed upon completion of a DMA transfer whether it be a one machine cycle transfer or two machine cycles (for ext. to ext. memory transfers).

2. A single machine cycle DMA register read operation (i.e. MOV A, DCON0) will inhibit the DMA for one instruction cycle. However a two cycle DMA register read operation will not inhibit the DMA (i.e. MOV P1, DCON0).

If the HOLD/HOLD Acknowledge logic is enabled in requestor mode the hold request will go active once the go bit has been set (for burst mode) and once the demand flag is set (for demand mode) regardless of whether the DMA is held off by one of the above conditions.

The DMA Transfer waveforms are in Figures 8-11.

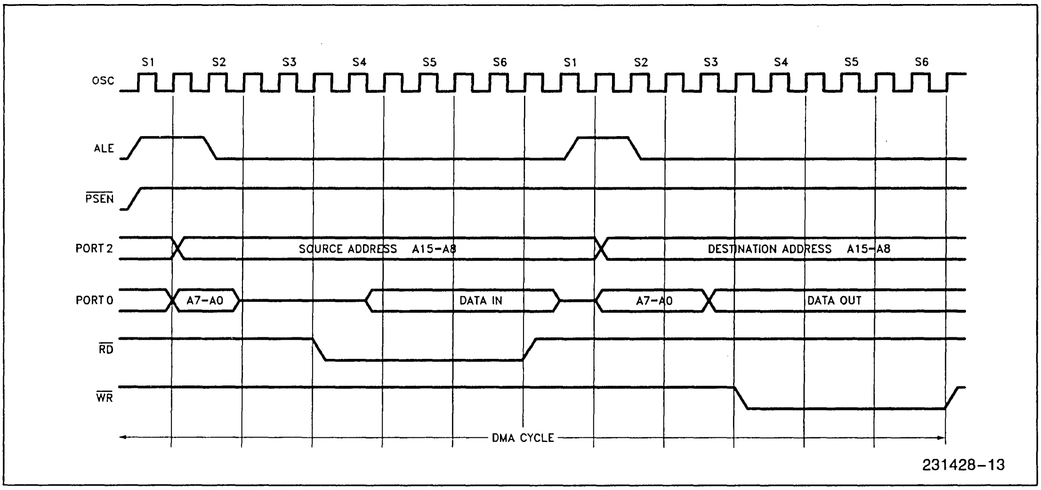


Figure 8. DMA Transfer from External Memory to External Memory

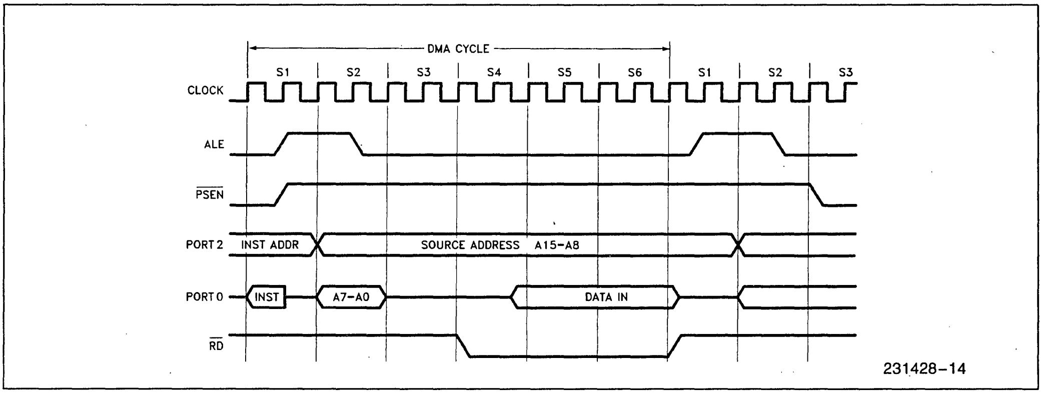


Figure 9. DMA Transfer from External Memory to Internal Memory

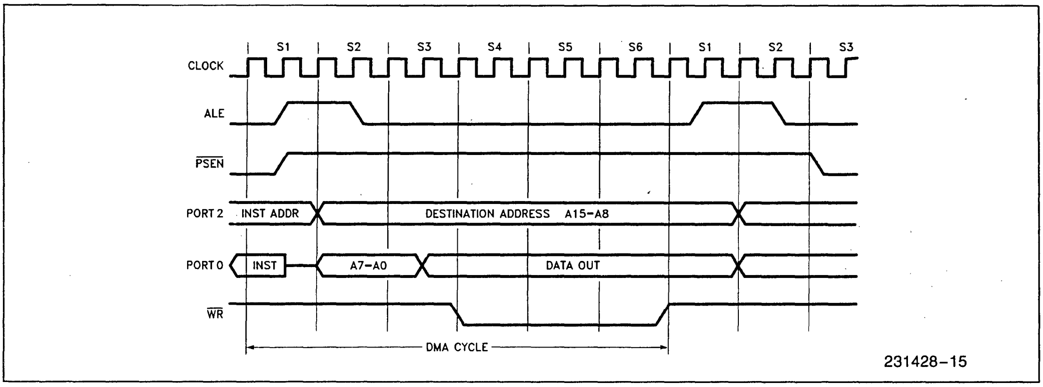


Figure 10. DMA Transfer from Internal Memory to External Memory

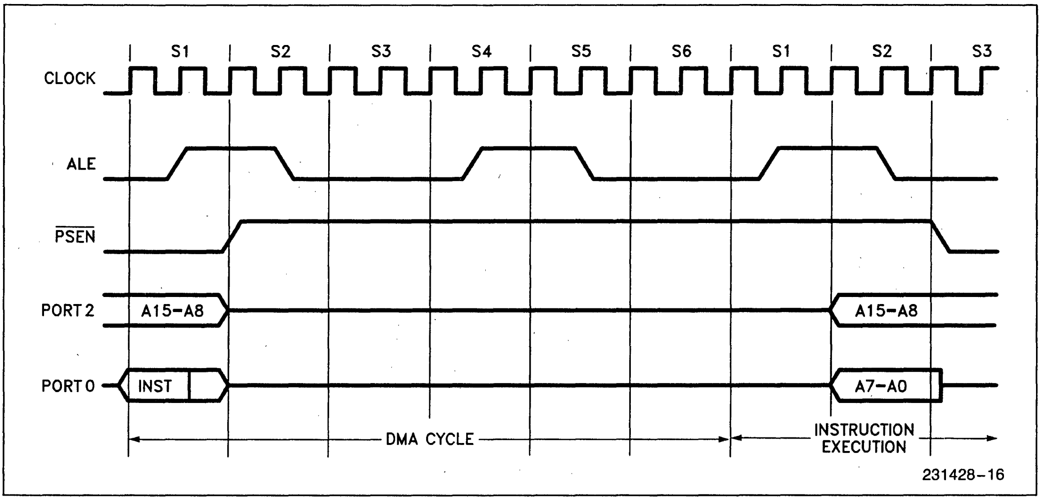


Figure 11. DMA Transfer from Internal Memory to Internal Memory

INTERNAL INTERRUPTS

Overview

The UPI-452 provides a total of eight interrupt sources (Table 6). Their operation is the same as in the 80C51, with the addition of three new interrupt sources for the UPI-452 FIFO and DMA features. These added interrupts have their enable and priority bits in the Interrupt Enable and Priority (IEP) SFR. The IEP SFR is in addition to the 80C51 Interrupt Enable (IE) and Interrupt Priority (IP) SFRs. The added interrupt sources are also globally enabled or disabled by the EA bit in the Interrupt Enable SFR. Table 6 lists the eight interrupt sources in order of priority. Table 7 lists the eight interrupt sources and their respective address vector location in program memory. (DMA interrupts are discussed in the "General Purpose DMA Channels" section. Additional interrupt information for Timer/Counter, Serial Channel, External Interrupt may be found in the Microcontroller Handbook for the 80C51.)

FIFO Module Interrupts to Internal CPU

The FIFO module generates interrupts to the internal CPU whenever the FIFO requests service or when a Data Stream Command is in the COMMAND IN SFR. The Input FIFO will request service whenever it becomes full or not empty depending on bit 1 of the Slave Control SFR (IFRS). Similarly, the Output FIFO requests service when it becomes empty or not full as determined by bit 0 of the Slave Control SFR (OFRS). Request for Service interrupts are generated only if enabled by the internal CPU via the Interrupt Enable SFR, and the Slave Control Register.

Table 6. Interrupt Priority

Interrupt Source	Priority Level
	(highest)
External Interrupt 0	0
Internal Timer/Counter 0	1
DMA Channel 0 Request	2
External Interrupt 1	3
DMA Channel 1 Request	4
Internal Timer/Counter 1	5
FIFO - Slave Bus Interface	6
Serial Channel	7
	(lowest)

Table 7. Interrupt Vector Addresses

Interrupt Source	Starting Address
External Interrupt 0	3 (003H)
Internal Timer/Counter 0	11 (00BH)
External Interrupt 1	19 (013H)
Internal Timer/Counter 1	27 (01BH)
Serial Channel	35 (023H)
FIFO - Slave Bus Interface	43 (02BH)
DMA Channel 0 Request	51 (033H)
DMA Channel 1 Request	59 (03BH)

A Data Stream Command Interrupt is generated whenever there is a Data Stream Command in the COMMAND IN SFR. The interrupt is generated to ensure that the internal interrupt is recognized before another instruction is executed.

Immediate Command Interrupts

a. An Immediate Command IN interrupt is generated, if enabled, to the internal CPU when the Host has written to the Immediate Command IN (IMIN) SFR. The write operation clears the Slave Status SFR bit (SSTAT SST2) and sets the Host Status SFR bit (HSTAT HST2) to indicate that a byte is present in the Immediate Command IN SFR. When the internal CPU reads the Immediate Command IN (IMIN) SFR the Slave Status SFR status bit is set, and the Host Status SFR status bit is cleared indicating the IMIN SFR is empty. Clearing the Host Status SFR bit will cause a Request For Service (INTRQ) interrupt, if enabled, to signal the Host that the IMIN SFR is empty. (See Figure 7a, Immediate Command IN Flowchart.)

b. An Immediate Command OUT interrupt is generated, if enabled, to the internal CPU when the Host has read the Immediate Command OUT SFR. The Host read causes the Slave Status Immediate Command OUT bit (SSTAT SST6) to be set and the corresponding Host Status bit (HSTAT HST6) to be cleared indicating the SFR is empty. When the internal CPU writes to the Immediate Command OUT SFR, the Host Status bit is set and Slave Status bit is cleared to indicate the SFR is full. (See Figure 7b, Immediate Command OUT Flowchart.)

NOTE:
Immediate Command IN and OUT interrupts are actually specific FIFO-Slave Interface interrupts to the internal CPU.

One instruction from the main program is executed between two consecutive interrupt service routines as in the 80C51. However, if the second interrupt service routine is due to a Data Stream Command Interrupt, the main program instruction is not executed (to prevent misreading of invalid data).

Interrupt Enabling and Priority

Each of the three interrupt special function registers (IE, IP and IEP) is listed below with its corresponding bit definitions.

Interrupt Enable SFR (IE)

Symbolic Address									Physical Address
IE	EA	—	—	ES	ET1	EX1	ET0	EX0	0A8H
	(MSB)							(LSB)	

Symbol	Position	Function
EA	IE.7	Enables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	(reserved)
—	IE.5	(reserved)
ES	IE.4	Serial Channel interrupt enable
ET1	IE.3	Internal Timer/Counter 1 Overflow Interrupt
EX1	IE.2	External Interrupt Request 1.
ET0	IE.1	Internal Timer/Counter 0 Overflow Interrupt
EX0	IE.0	External Interrupt Request 0.

Interrupt Priority SFR (IP)

A priority level of 0 or 1 may be assigned to each interrupt source, with 1 being higher priority level, through the IP and the IEP (Interrupt Enable and Priority) SFR. A priority level of 1 interrupt can interrupt a priority level 0 service routine to allow nesting of interrupts.

Symbolic Address									Physical Address
IP	—	—	—	PS	PT1	PX1	PT0	PX0	0B8H
	(MSB)							(LSB)	

Symbol	Position	Function	Priority Within A Level
			(lowest)
—	IP.7	(reserved)	—
—	IP.6	(reserved)	—
—	IP.5	(reserved)	—
PS	IP.4	Local Serial Channel	0.7
PT1	IP.3	Internal Timer/Counter 1	0.5
PX1	IP.2	External Interrupt Request 1	0.3
PT0	IP.1	Internal Timer/Counter 0	0.1
PX0	IP.0	External Interrupt Request 0	0.0
			(highest)

Interrupt Enable and Priority SFR (IEP)

The Interrupt Enable and Priority Register establishes the enabling and priority of those resources not covered in the Interrupt Enable and Interrupt Priority SFRs.

Symbolic Address									Physical Address
IEP	—	—	PFIFO	EDMA0	EDMA1	PDMA0	PDMA1	EFIFO	0F8H
	(MSB)							(LSB)	

Symbol	Position	Function	Priority Within a Level
—	IEP.7	(reserved)	
—	IEP.6	(reserved)	
PFIFO	IEP.5	FIFO Slave Bus Interface Interrupt Priority	0.6
EDMA0	IEP.4	DMA Channel 0 Interrupt Enable	
EDMA1	IEP.3	DMA Channel 1 Interrupt Enable	
PDMA0	IEP.2	DMA Channel 0 Priority	0.2
PDMA1	IEP.1	DMA Channel 1 Priority	0.4
EFIFO	IEP.0	FIFO Slave Bus Interface Interrupt Enable	

FIFO-EXTERNAL HOST INTERFACE FIFO DMA FREEZE MODE

Overview

During FIFO DMA Freeze Mode the internal CPU can reconfigure the FIFO interface. FIFO DMA Freeze Mode is provided to prevent the Host from accessing the FIFO during a reconfiguration sequence. The internal CPU invokes FIFO DMA Freeze Mode by clearing bit 3 of the Slave Control SFR (SC3). INTRQ becomes active whenever FIFO DMA Freeze Mode is invoked to indicate the freeze status. The interrupt can only be deactivated by the Host reading the Host Status SFR.

During FIFO DMA Freeze Mode only two operations are possible by the Host to the UPI-452 slave, the balance are disabled, as shown in Table 8. The internal DMA is disabled during FIFO DMA Freeze Mode, and the internal CPU has write access to all of the FIFO control SFRs (Table 9).

Initialization

At power on reset the FIFO Host interface is automatically frozen. The Slave Control Enable FIFO DMA Freeze Mode bit defaults to FIFO DMA Freeze Mode (SLCON FRZ=0). Below is a list of the FIFO Special Function Registers and their default power on reset values;

SFR Name	Label	Value
Channel Boundary Pointer	CBP	40H / 64D
Output Channel Read Pointers	ORPR	40H / 64D
Output Channel Write Pointers	OWPR	40H / 64D
Input Channel Read Pointers	IRPR	00H / 00D
Input Channel Write Pointers	IWPR	00H / 00D
Input Threshold	ITHR	80H / 128D
Output Threshold	OTHR	01H / 1D

The Input and Output FIFO channels can be reconfigured by programming any of these SFRs while the UPI-452 is in the Freeze Mode. The Host is notified when the Freeze Mode is active by a "1" in HST1 of the Host Status Register (HSTAT). The Host should interrogate HST1 to determine the status of the FIFO interface following reset before attempting to read from or write to the UPI-452 FIFO buffer.

NOTE:

During the initialization sequence of the UPI-452 FIFO SFRs, the OTHR should be changed from the default setting of 1 to a value between 2 and {(80H-CBP)-1}. Please refer to the section on Input and Output FIFO threshold SFRs for further information.

Table 8. Slave Bus Interface Status During FIFO DMA Freeze Mode

Interface Pins; $\overline{\text{DACK}}$	$\overline{\text{CS}}$	A2	A1	A0	$\overline{\text{READ}}$	$\overline{\text{WRITE}}$	Operation In Normal Mode	Status In FIFO DMA Freeze Mode
1	0	0	1	0	0	1	Read Host Status SFR	Operational
1	0	0	1	1	0	1	Read Host Control SFR	Operational
1	0	0	1	1	1	0	Write Host Control SFR	Disabled
1	0	0	0	0	0	1	Data or DMA Data from Output Channel	Disabled
1	0	0	0	0	1	0	Data or DMA Data to Input Channel	Disabled
1	0	0	0	1	0	1	Data Stream Command from Output Channel	Disabled
1	0	0	0	1	1	0	Data Stream Command to Input Channel	Disabled
1	0	1	0	0	0	1	Read Immediate Command Out from Output Channel	Disabled
1	0	1	0	0	1	0	Write Immediate Command In to Input Channel	Disabled
0	X	X	X	X	0	1	DMA Data from Output Channel	Disabled
0	X	X	X	X	1	0	DMA Data to Input Channel	Disabled

The UPI-452 can also be programmed to interrupt the Host following power on reset in order to indicate to the Host that FIFO DMA Freeze Mode is in progress. This is done by enabling the INTRQ interrupt output pin via the MODE SFR (MD4) before the Slave Control SFR Enable FIFO DMA Freeze Mode bit is set to Normal Mode. At power on reset the Mode SFR is forced to zero. This disables all interrupt and DMA output pins (INTRQ, DRQIN/INTRQIN and DRQOUT/INTRQOUT). Because the Host Status SFR FIFO DMA Freeze Mode In Progress bit is set, a Request For Service, INTRQ, interrupt is pending until the Host Status SFR is read. This is because the FIFO DMA Freeze Mode interrupt is always enabled. If the Slave Control FIFO DMA Freeze Mode bit (SLCON FRZ) is set to Normal Mode before the MODE SFR INTRQ bit is enabled, the INTRQ output will not go active when the MODE SFR INTRQ bit is enabled if the Host Status SFR has been read.

The default values for the FIFO and Slave Interface represents minimum UPI-452 internal initialization. No specific Special Function Register initialization is required to begin operation of the FIFO Slave Interface. The last initialization instruction must always set the UPI-452 to Normal Mode. This causes the UPI-452 to exit FIFO DMA Freeze Mode and enables Host read/write access of the FIFO.

Following reset, either hardware (via the RST pin) or software (via HCON SFR bit HC3) the UPI-452 requires 2 internal machine cycles (24 TCLCL) to update all internal registers.

Invoking FIFO DMA Freeze Mode During Normal Operation

When the UPI-452 is in normal operation, FIFO DMA Freeze Mode should not be arbitrarily invoked by clearing SC3 (SC3 = 0) because the external Host runs asynchronously to the internal CPU. Invoking FIFO DMA Freeze Mode without first stopping the external Host from accessing the UPI-452 will not guarantee a clean break with the external Host.

The proper way to invoke FIFO DMA Freeze Mode is by issuing an Immediate Command to the external host indicating that FIFO DMA Freeze Mode will be invoked. Upon receiving the Immediate Command, the external Host should complete servicing all pending interrupts and DMA requests, then send an Immediate Command back to the UPI-452 acknowledging the FIFO DMA Freeze Mode request. After issuing the first Immediate Command, the internal CPU should not perform any action on the FIFO until FIFO DMA Freeze Mode is invoked.

If FIFO DMA Freeze Mode is invoked without stopping the Host during Host transfers, only the last two bytes of data written into or read from the FIFO will be valid. The timing diagram for disabling the FIFO module to the external Host interface is illustrated in Figure 12. Due to this synchronization sequence, the UPI-452 might not go into FIFO DMA Freeze Mode immediately after SC3 is cleared. A special bit in the Slave Status Register (SST5) is provided to indicate the status of the FIFO DMA Freeze Mode. The FIFO DMA Freeze Mode operations described in this section are only valid after SST5 is cleared.

As FIFO DMA Freeze Mode is invoked, the DRQIN or DRQOUT will be deactivated (stopping the transferring of data), bit 1 of the Host Status SFR will be set (HST1 = 1), and SST5 will be cleared (SST5 = 0) to indicate to the external Host and internal CPU that the slave interface has been frozen. After the freeze becomes effective, any attempt by the external Host to access the FIFO will cause the overrun and underrun bits to be activated (bits HST7 (for reads) or HST3 (for writes)). These two bits, HST3 and HST7, will be set (deactivated) after the Host Status SFR has been read. If INTRQ is used to request service, the FIFO interface is frozen upon completion of any Host read or write operation in progress.

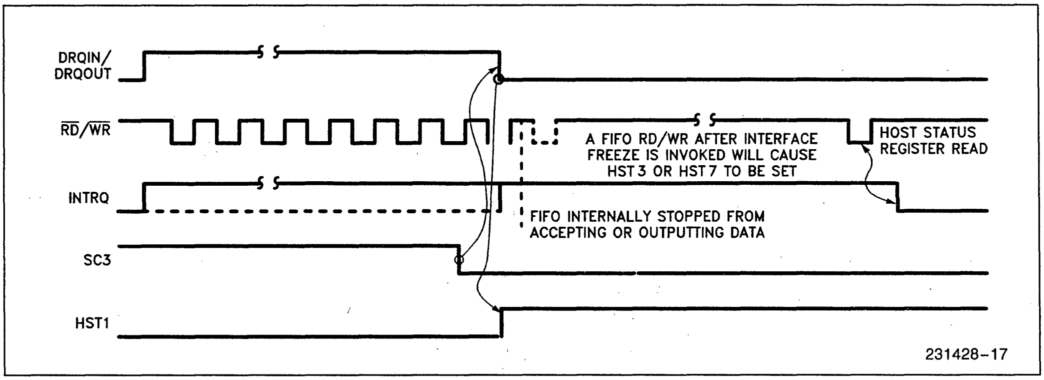

DRQIN/
DRQOUT
RD/WR
INTRQ
SC3
HST1
A FIFO RD/WR AFTER INTERFACE FREEZE IS INVOKED WILL CAUSE HST3 OR HST7 TO BE SET
HOST STATUS REGISTER READ
FIFO INTERNALLY STOPPED FROM ACCEPTING OR OUTPUTTING DATA
231428-17

Figure 12. Disabling FIFO to Host Slave Interface Timing Diagram

External Host writing to the Immediate Command In SFR and the Host Control SFR is also inhibited when the slave bus interface is frozen. Writing to these two registers after FIFO DMA Freeze Mode is invoked will also cause HST3 (overrun) to be activated (HST3 = 0). Similarly, reading the Immediate Command Out Register by the external Host is disabled during FIFO DMA Freeze Mode, and any attempt to do so will cause the clearing (deactivating, "0") of HST7 bit (underrun).

After the slave bus interface is frozen, the internal CPU can perform the following operations on the FIFO Special Function Registers (these operations are allowed only during FIFO DMA Freeze Mode).

For FIFO Reconfiguration	1. Changing the Channel Boundary Pointer SFR. 2. Changing the Input and Output Threshold SFR.
To Enhance the Testability	3. Writing to the read and write pointers of the Input and Output FIFO's. 4. Writing to and reading the Host Control SFRs. 5. Controlling some bits of Host and Slave Status SFRS. 6. Reading the Immediate Command Out SFR and Writing to the Immediate Comand In SFR.

Description of each of these special functions are as follows:

FIFO Module SFRs During FIFO DMA Freeze Mode

Table 9 summarizes the characteristics of all the FIFO Special Function Registers during normal and FIFO DMA Freeze Modes. The registers that require special treatment in FIFO DMA Freeze Mode are: HCON, IWPR, IRPR, OWPR, ORPR, HSTAT, SSTAT, MIN & MOUT SFRs. They can be described in detail as follows:

Host Control SFR (HCON)

During normal operation, this register is written to or read by the external Host. However, in FIFO DMA Freeze Mode (i.e. SST5 = 0) the UPI-452 internal CPU has write access to the Host Control SFR and write operations to this SFR by the external Host will not be accepted. If the Host attempts to write to HCON, the Input Channel error condition flag (HST3) will be cleared.

Input FIFO Pointer Registers (IRPR & IWPR)

Once the FIFO module is in FIFO DMA Freeze Mode, error flags due to overrun and underrun of the Input FIFO pointers will be disabled. Any attempt to create an overrun or underrun condition by changing the Input FIFO pointers would result in an inconsistency in performance between the status flag and the threshold counter.

To enhance the speed of the UPI-452, read operations on the Input FIFO will look ahead by two bytes. Hence, every time the IRPR is changed during FIFO DMA Freeze Mode, two NOPs need to be executed so that the two byte pipeline can be updated with the new data bytes pointed to by the new IRPR. The Threshold Counter SFR also needs to change by the same number of bytes as the IRPR (increase Threshold Counter if IRPR goes forward or decrease if IRPR goes backward). This will ensure that future interrupts will still be generated only after a threshold number of bytes are available. (See "Input and Output FIFO Threshold SFR" section below.)

In FIFO DMA Freeze Mode, the internal CPU can also change the content of IWPR, and each change of IWPR also requires an update of the Threshold Counter SFR.

Normally, the internal CPU cannot write into the Input FIFO. It can, however, during FIFO DMA Freeze Mode by first reconfiguring the FIFO as an Output FIFO (Refer to "Input and Output FIFO Threshold SFR" section below). Changing the IRPR to be equal to IWPR generates an empty condition while changing IWPR to be equal to IRPR generates a full condition. The order in which the pointers are written determines whether a full or empty condition is generated.

Output FIFO Pointer SFR (ORPR and OWPR)

In FIFO DMA Freeze Mode the contents of OWPR can be changed by the internal CPU, but each change of OWPR or ORPR requires the Threshold Counter SFR to be updated as described in the next section. A NOP must be executed whenever a new value is written into ORPR, as just described for changes to IRPR. As before, changing ORPR to be equal to OWPR will generate an empty condition, Output FIFO overrun or underrun condition cannot be generated though. The FIFO pointers should not be set to a value outside of its range.

Table 9. FIFO SFR's Characteristics During FIFO DMA Freeze Mode

Label	Name	Normal Operation (SST5 = 1)	FIFO DMA Freeze Mode Operation (SST5 = 0)
HCON	Host Control	Not Accessible	Read & Write
HSTAT	Host Status	Read Only	Read & Write 4
SLCON	Slave Control	Read & Write	Read & Write
SSTAT	Slave Status	Read Only	Read & Write 4
IEP	Interrupt Enable & Priority	Read & Write	Read & Write
MODE	Mode Register	Read & Write	Read & Write
IWPR	Input FIFO Write Pointer	Read Only	Read & Write 5
IRPR	Input FIFO Read Pointer	Read Only	Read & Write 1, 5
OWPR	Output FIFO Write Pointer	Read Only	Read & Write 6
ORPR	Output FIFO Read Pointer	Read Only	Read & Write 2, 6
CBP	Channel Boundary Pointer	Read Only	Read & Write 3
IMIN	Immediate Command In	Read Only	Read & Write
IMOUT	Immediate Command Out	Read & Write	Read & Write
FIN	FIFO IN	Read Only	Read Only
CIN	COMMAND IN	Read Only	Read Only
FOUT	FIFO OUT	Read & Write	Read & Write
COUT	COMMAND OUT	Read & Write	Read & Write
ITHR	Input FIFO Threshold	Read Only	Read & Write
OTHR	Output FIFO Threshold	Read Only	Read & Write

NOTES:

1. Writing of IRPR will automatically cause the FIFO IN SFR to load the contents of the Input FIFO from that location.
2. Writing to ORPR will automatically cause the IOBL SFR to load the contents of the Output FIFO at that ORPR address.
3. Writing to the CBP SFR will cause automatic reset of the four pointers of the Input and Output FIFO channels.
4. The internal CPU cannot directly change the status of these registers. However, by changing the status of the FIFO channels, the internal CPU can indirectly change the contents of the status registers.
5. Changing the Input FIFO Read/Write Pointers also requires that a consistent update of the Input FIFO Threshold Counter SFR.
6. Changing the Output FIFO Read/Write Pointers also requires that a consistent update of the Output FIFO Threshold Counter SFR.

Input and Output FIFO Threshold SFR (ITHR & OTHR)

The Input and Output FIFO Threshold SFRs are also programmable by the internal CPU during FIFO DMA Freeze Mode. For proper operation of the Threshold feature, the Threshold SFR should be changed only when the Input and Output FIFO channels are empty, since they reflect the current number of bytes available to read/write before an interrupt is generated.

Table 10 illustrates the Threshold SFRs range of values and the number of bytes to be transferred when the Request For Service Flag is activated:

Table 10. Threshold SFRs Range of Values and Number of Bytes to be Transferred

ITHR (lower seven bits)	No. of Bytes Available to be Written	OTHR (lower seven bits)	No. of Bytes Available to be Read
0	CBP	2	3
1	CBP-1	3	4
2	CBP-2	•	•
•	•	•	•
•	•	•	•
•	•	(80H-CBP)-3	(80H-CBP)-2
CBP-3	3	(80H-CBP)-2	(80H-CBP)-1
		(80H-CBP)-1	(80H-CBP)

The eighth bit of the Input and Output FIFO Threshold SFR indicates the status of the service requests regardless of the freeze condition. If the eighth bit is a "1", the FIFO is requesting service from the external Host. In other words, when the Threshold SFR value goes below zero (2's complement), a service request is generated*. *The 8th bit of the ITHR SFR must be set during initialization if the Host interrupt request is desired immediately upon leaving Freeze Mode. Normally the ITHR SFR is decremented after each external Host write to the Input FIFO and incremented after each internal CPU read of the Input FIFO. The OTHR SFR is decremented by internal CPU writes and incremented by external Host reads. Thus if the pointers are moved when the FIFO's are not empty, these relationships can be used to calculate the offset for the Threshold SFRs. It is best to change the Threshold SFRs only when the FIFO's are empty to avoid this complication. The threshold registers should also be updated after the pointers have been manipulated.

NOTE:
The ITHR should only be programmed in the range from 0 to (CBP-3). An ITHR value of (CBP-2) could result in a failure to set the Input FIFO service request signal after the Input FIFO has been emptied. Correspondingly, the OTHR should be programmed in the range from 2 to {(80H-CBP)-1}. An OTHR value of 1 could result in a failure to set the Output FIFO service request after subsequent writes by the UPI-452 have filled the Output FIFO.

NOTE:
When programming the ITHR SFR, the eighth bit should be set to 1 (OR'd with 80H). This causes HSTAT SFR HST0 = 0, Input FIFO Request For Service. If ITHR bit 7 = 0 then HSTAT HST0 = 1, Input FIFO Does Not Request Service, and no interrupt will be generated.

Host Status SFR (HSTAT)

When in FIFO DMA Freeze Mode, some bits in the Host Status SFR are forced high and will not reflect the new status until the system returns to normal operation. The definition of the register in FIFO DMA Freeze Mode is as follows:

NOTE:
The internal CPU reads this shadow latch value when reading the Host Status SFR. The shadow latch will keep the information for these bits so normal operation can be resumed with the right status. The following bits are set (= 1) when FIFO DMA Freeze Mode is invoked;

HST7 Output FIFO Error Condition Flag

1 = No error.

0 = An invalid read has been done on the output FIFO or the Immediate Command Out Register by the host CPU.

NOTE:
The normal underrun error condition status is disabled. If an Immediate Command Out (IMOUT) SFR read is attempted during FIFO DMA Freeze Mode, the contents of the IMOUT SFR is output on the Data Buffer and the error status is cleared (= 0).

HST6 Immediate Command Out SFR Status

During normal operation, this bit is cleared (=0) when the IMOUT SFR is written by the UPI-452 internal CPU and set (= 1) when the IMOUT SFR is read by the external Host. Once the host-slave interface is frozen (i.e. SST5 = 0), this bit will be read as a 1 by the host CPU. A shadow latch will keep the information for this bit so normal operation can be resumed with the correct status.

Shadow latch:

1 = Internal CPU reads the IMOUT SFR

0 = Internal CPU writes to the IMOUT SFR

HST5 Data Stream Command at Output FIFO

This bit is forced to a "1" during FIFO DMA Freeze Mode to prevent the external host CPU from trying to read the DSC. Once normal operation is resumed, HST5 will reflect the Data/Command status of the current byte in the Output FIFO.

Shadow Latch (read by the internal CPU):

1 = No Data Stream Command (DSC)

0 = Data Stream Command at Output FIFO

HST4 Output FIFO Service Request Status

When FIFO DMA Freeze Mode is invoked, this bit no longer reflects the Output FIFO Request Service Status. This bit wll be forced to a "1".

HST3 Input FIFO Error Condition Flag

1 = No error.

0 = One of the following operations has been attempted by the external host and is invalid:

1) Write into the Input FIFO

2) Write into the Host Control SFR

3) Write into the Immediate Command In SFR

NOTE:
The normal Input FIFO overrun condition is disabled.

HST2 Immediate Command In SFR Status

This bit is normally cleared when the internal CPU reads the IMIN SFR and set when the external host CPU writes into the IMIN SFR. When the host-slave interface is frozen, reading and writing of the IMIN by the internal CPU will change the shadow latch of this bit. This bit will be read as a "1" by the external Host.

Shadow latch.

1 = Internal CPU writes into IMIN SFR

0 = Internal CPU reads the IMIN SFR

HST1 FIFO DMA Freeze Mode Status

1 = FIFO DMA Freeze Mode.

0 = Normal Operation (non-FIFO DMA Freeze Mode).

NOTE:
This bit is used to indicate to the external Host that the host-slave interface has been frozen and hence the external Host functions are now reduced as shown in Table 8.

HST0 Input FIFO Request Service Satus

When slave interface is frozen this bit no longer reflects the Input FIFO Request Service Status. This bit will be forced to a "1".

Slave Status SFR (SSTAT)

The Slave Status SFR is a read-only SFR. However, once the slave interface is frozen, most of the bits of this SFR can be changed by the internal CPU by reconfiguring the FIFO and accessing the FIFO Special Function Registers.

SST7 Output FIFO Overrun Error Flag

Inoperative in FIFO DMA Freeze Mode.

SST6 Immediate Command Out SFR Status

In FIFO DMA Freeze Mode, this bit will be cleared when the internal CPU reads the Immediate Command Out SFR and set when the internal CPU writes to the Immediate Command Out Register.

SST5 FIFO-External Interface FIFO DMA Freeze Mode Status

This bit indicates to the internal CPU that FIFO DMA Freeze Mode is in progress and that it has write access to the FIFO Control, Host control and Immediate Command SFRs.

SST4 Output FIFO Request Service Status

During normal operation, this bit indicates to the internal CPU that the Output FIFO is ready for more data. The status of this bit reflects the position of the Output FIFO read and write pointers. Hence, in FIFO DMA Freeze Mode, this flag can be changed by the internal CPU indirectly as the read and write pointers change.

SST3 Input FIFO Underrun Flag

Inoperative during FIFO DMA Freeze Mode.

During normal operation, a read operation clears (=0) this bit when there are no data bytes in the Input FIFO and deactivated (=1) when the Slave Status SFR is read. In FIFO DMA Freeze Mode, this bit will not be cleared by an Input FIFO read underrun error condition, nor will it be reset by the reading of the Slave Status SFR.

SST2 Immediate Command In SFR Status

This bit is normally activated (=0) when the external host CPU writes into the Immediate Command In SFR and deactivated (=1) when it is read by the internal CPU. In FIFO DMA Freeze Mode, this bit will not be activated (=0) by the external Host's writing of the Immediate Command IN SFR since this function is disabled. However, this bit will be cleared (=0) if the internal CPU writes to the Immediate Command In SFR and it will be set =1) if it reads from the register.

SST1 Data Stream Command at Input FIFO Flag

In FIFO DMA Freeze Mode, this bit operates normally. It indicates whether the next byte of data from the Input FIFO is a DSC or data byte. If it is a DSC byte, reading from the FIFO IN SFR will result in reading invalid data (FFH) and vice versa. In FIFO DMA Freeze Mode, this bit still reflects the type of data byte available from the Input FIFO.

SST0 Input FIFO Service Request Flag

During normal operation, this bit is activated (=0) when the Input FIFO contains bytes that can be read by the internal CPU and deactivated (=1) when the Input FIFO does not need any service from the internal CPU. In FIFO DMA Freeze Mode, the status of this bit should not change unless the pointers of the Input FIFO are changed. In this mode, the internal CPU can indirectly change this bit by changing the read and write pointers of the Input FIFO but cannot change it directly.

Immediate Command In/Out SFR (IMIN/IMOUT)

If FIFO DMA Freeze Mode is in progress, writing to the Immediate Command In SFR by the external host will be disabled, and any such attempt will cause HST3 to be cleared (=0). Similarly, the Immediate Command Out SFR read operation (by the host) will be disabled internally and read attempts will cause HST7 to be cleared (=0).

Internal CPU Read and Write of the FIFO During FIFO DMA Freeze Mode

In normal operation, the Input FIFO can only be read by the internal CPU and similarly, the Output FIFO can only be written by the internal CPU. During FIFO DMA Freeze Mode, the internal CPU can read the entire contents of the Input FIFO by programming the CBP SFR to 7FH, setting the IRPR SFR to zero, and then the IWPR SFR to zero. Programming the pointer registers in this order generates a FIFO full signal to the FIFO logic and enables internal CPU read operations. If the IWPR and IRPR are already zero, the write pointer should be changed to a non-zero value to clear the empty status then the pointers can be set to zero. Writing to the IRDR SFR automatically updates the look ahead registers.

In a similar manner, the internal CPU can write to all 128 bytes of the FIFO by setting the CBP SFR to zero, setting OWPR SFR to zero, and then setting ORPR SFR to zero. This generates a FIFO empty signal and allows internal CPU write operations to all 128 bytes of the FIFO. The Threshold registers also need to be adjusted when the pointers are changed. (See "Input and Output FIFO Threshold SFR" section below.)

MEMORY ORGANIZATION

The UPI-452 has separate address spaces for Program Memory and Data Memory like the 80C51. The Program Memory can be up to 64K bytes. The lower 8K of Program Memory may reside on-chip. The Data Memory consists of 256 bytes of on-chip RAM, up to 64K bytes of off-chip RAM and a number of "SFRs" (Special Function Registers) which appear as yet another set of unique memory addresses.

Table 11a. Internal Memory Addressing

Memory Space	Addressing Method
Lower 128 Bytes of Internal RAM	Direct or Indirect
Upper 128 Bytes of Internal RAM	Indirect Only
UPI-452 SFR's	Direct Only

The 80C51 Special Function Registers are listed in Table 11a, and the additional UPI-452 SFRs are listed in Table 11b. A brief description of the 80C51 core SFRs is also provided below.

Accessing External Memory

As in the 80C51, accesses to external memory are of two types: Accesses to external Program Memory and accesses to external Data Memory.

External Program Memory is accessed under two conditions:

1) Whenever signal $\overline{EA}$ = 0; or
2) Whenever the program counter (PC) contains a number that is larger than 1FFFH.

This requires that the ROMless versions have $\overline{EA}$ wired low to enable the lower 8K program bytes to be fetched from external memory.

External Data Memory is accessed using either the MOVX @DPTR (16 bit address) or the MOVX @Ri (8 bit address) instructions, or during external data memory transfers.

Table 11b. 80C51 Special Function Registers

Symbol	Name	Address	Contents
*ACC	Accumulator	0E0H	00H
*B	B Register	0F0H	00H
*PSW	Program Status Word	0D0H	00H
SP	Stack Pointer	81H	07H
DPTR	Data Pointer (consisting of DPH and DPL)	82H	0000H
*P0	Port 0	80H	0FFH
*P1	Port 1	90H	0FFH
*P2	Port 2	0A0H	0FFH
*P3	Port 3	0B0H	0FFH
*IP	Interrupt Priority Control	0B8H	0E0H
*IE	Interrupt Enable Control	0A8H	60H
TMOD	Timer/Counter Mode Control	89H	00H
*TCON	Timer/Counter Control	88H	00H
TH0	Timer/Counter 0 (high byte)	8CH	00H
TL0	Timer/Counter 0 (low byte)	8AH	00H
TH1	Timer/Counter 1 (high byte)	8DH	00H
TL1	Timer/Counter 1 (low byte)	8BH	00H
*SCON	Serial Control	98H	00H
SBUF	Serial Data Buff	99H	I
PCON	Power Control	87H	I0H

I = Indeterminate
The SFRs marked with an asterisk (*) are both bit- and byte- addressable. The functions of the SFRs are as follows:

Table 11c. UPI-452 Additional Special Function Registers

Symbol	Name	Address	Contents
BCRL0	DMA Byte Count Low Byte/	0E2H	I
BCRH0	High Byte/ Channel 0	0E3H	I
BCRL1	Low Byte/	0F2H	I
BCRH1	Hi Byte/ Channel 1	0F3H	I
CBP	Channel Boundary Pointer	0ECH	40H
CIN	COMMAND IN	0EFH	I
COUT	COMMAND OUT DMA Destination Address	0FFH	I

Table 11c. UPI-452 Additional Special Function Registers (Continued)

Symbol	Name	Address	Contents
DARL0	Low Byte/	0C2H	I
DARH0	Hi Byte/ Channel 0	0C3H	I
DARL1	Low Byte/	0D2H	I
DARH1	Hi Byte/ Channel 1	0D3H	I
DCON0	DMA0 Control	92H	00H
DCON1	DMA1 Control	93H	00H
FIN	FIFO IN	0EEH	I
FOUT	FIFO OUT	0FEH	I
HCON	Host Control	0E7H	00H
HSTAT	Host Status	0E6H	0FBH
*IEP	Interrupt Enable and Priority	0F8H	0C0H
IMIN	Immediate Command In	0FCH	I
IMOUT	Immediate Command Out	0FDH	I
IRPR	Input Read Pointer	0EBH	00H
ITHR	Input FIFO Threshold	0F6H	80H
IWPR	Input Write Pointer	0EAH	00H
MODE	Mode Register	0F9H	8FH
ORPR	Output Read Pointer	0FAH	40H
OTHR	Output FIFO Threshold	0F7H	01H
OWPR	Output Write Threshold	0FBH	40H
*P4	Port 4 DMA Source Address	0C0H	0FFH
SARL0	Low Byte/	0A2H	I
SARH0	Hi Byte/ Channel 0	0A3H	I
SARL1	Low Byte/	0B2H	I
SARH1	Hi Byte/ Channel 1	0B3H	I
*SLCON	Slave Control	0E8H	04H
SSTAT	Slave Status	0E9H	08FH

I = Indeterminate
The SFRs marked with an asterisk (*) are both bit- and byte- addressable. The functions of the SFRs are as follows:

Miscellaneous Special Function Register Description

80C51 SFRs

ACCUMULATOR

ACC is the Accumuator SFR. The mnemonics for accumulator-specific instructions, however, refer to the accumulator simply as A.

B REGISTER

The B SFR is used during multiply and divide operations. For other instructions it can be treated as another scratch pad regster.

PROGRAM STATUS WORD

The PSW SFR contains program status information as detailed in Table 12.

STACK POINTER

The Stack Pointer register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on-chip RAM, the Stack Pointer is initialized to 07H after a reset. This causes the stack to begin at location 08H.

DATA POINTER

The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address. It may be manipulated as a 16-bit register or as two independent 8-bit registers.

PORTS 0 TO 4

P0, P1, P2, P3 and P4 are the SFR latches of Ports 0, 1, 2, 3 and 4, respectively.

SERIAL DATA BUFFER

The Serial Data Buffer is actually two separate registers, a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to the transmit buffer where it is held for serial transmission. (Moving a byte to SBUF is what initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

TIMER/COUNTER SFR

Register pairs (TH0, TL0), and (TH1, TL1) are the 16-bit counting registers for Timer/Counters 0 and 2.

POWER CONTROL SFR (PCON)

The PCON Register (Table 13) controls the power down and idle modes in the UPI-452, as well as providing the ability to double the Serial Channel baud rate. There are also two general purpose flag bits available to the user. Bits 5 and 6 are used to set the HOLD/HOLD Acknowledge mode (see "General Purpose DMA Channels" section), and bit 4 is not used.

Table 12. Program Status Word

Symbolic Address									Physical Address
PSW	CY	AC	FO	RS1	RS0	OV	—	P	0D0H
	(MSB)							(LSB)	

Symbol	Position	Name
CY	PSW.7	Carry Flag
AC	PSW.6	Auxiliary Carry (For BCD operations)
F0	PSW.5	Flag 0 (user assignable)
RS1	PSW.4	Register Bank Select bit 1*
RS0	PSW.3	Register Bank Select bit 0*
OV	PSW.2	Overflow Flag
—	PSW.1	(reserved)
P	PSW.0	Parity Flag

*(RS1, RS0) enable internal RAM register banks as follows:

RS1	RS0	Internal RAM Register Bank
0	0	Bank 0
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

Table 13. PCON Special Function Register

Symbolic Address									Physical Address
PCON	SMOD	ARB	REQ	—	GF1	GF0	PD	IDL	087H
	(MSB)							(LSB)	

Symbol	Position	Function
SMOD	PCON7	Double Baud rate bit. When set to a 1, the baud rate is doubled when the serial port is being used in either Mode 1, 2 or 3.
ARB	PCON6	HLD/HLDA Arbiter control bit *
REQ	PCON5	HLD/HLDA Requestor control bit *
—	PCON4	(reserved)
GF1	PCON3	General-purpose flag bit
GF0	PCON2	General-purpose flag bit
PD	PCON1	Power Down bit. Setting this bit activates power down operation.
IDL	PCON0	Idle Mode bit. Setting this bit activates idle mode operation.

*See "Ext. Memory DMA" description.

NOTE:
If 1's are written to PD and IDL at the same time, PD takes precedence. The reset value of PCON is (000X0000).

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ..... 0°C to 70°C†
Storage Temperature .... −65°C to +150°C
Voltage on Any
  Pin to V_{SS} .... −0.5V to V_{CC} + 0.5V
Voltage on V_{CC} to V_{SS} .... −0.5V to +6.5V
Power Dissipation 1.0W**

**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_A = 0°C to 70°C; V_{CC} = 5V ±10%; V_{SS} = 0V

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage (except XTAL1, RST)	2.0	V_{CC} + 0.5	V	
V_{IH1}	Input High Voltage (XTAL1, RST)	3.9	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage (Ports 1, 2, 3, 4)		0.45	V	I_{OL} = 1.6 mA (Note 1)
V_{OL1}	Output Low Voltage (except Ports 1, 2, 3, 4)		0.45	V	I_{OL} = 3.2 mA (Note 1)
V_{OH}	Output High Voltage (Ports 1, 2, 3, 4)	2.4		V	I_{OH} = −60 μA, V_{CC} = 5V ±10%
		0.9 V_{CC}		V	I_{OH} = −10 μA
V_{OH1}	Output High Voltage (except Ports 1, 2, 3, 4 and Host Interface (Slave) Port)	2.4		V	I_{OH} = −400 μA, V_{CC} = 5V ±10%
		0.9 V_{CC}		V	I_{OH} = −40 μA (Note 2)
V_{OH2}	Output High Voltage (Host Interface (Slave) Port)	2.4		V	I_{OH} = −400 μA, V_{CC} = 5V ±10%
		V_{CC} − 0.4		V	I_{OH} = −10 μA
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3, 4)		−50	μA	V_{IN} = 0.45V
I_{TL}	Logical 1 to 0 Transition Current (Ports 1, 2, 3, 4)		−650	μA	V_{IN} = 2V

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

Symbol	Parameter	Min	Max	Units	Test Conditions
I_{LI}	Input Leakage Current (except Ports 1, 2, 3, 4)		±10	μA	$0.45V < V_{IN} < V_{CC}$
I_{OZ}	Output Leakage Current (except Ports 1, 2, 3, 4)		±10	μA	$0.45V < V_{OUT} < V_{CC}$
I_{CC}	Operating Current		50	mA	V_{CC} = 5.5V, 14 MHz (Note 4)
I_{CCI}	Idle Mode Current		25	mA	V_{CC} = 5.5V, 14 MHz (Note 5)
I_{PD}	Power Down Current		100	μA	V_{CC} = 2V (Note 3)
RRST	Reset Pulldown Resistor	50	150	KΩ	
CIO	Pin Capacitance		20	pF	1 MHz, T_A = 25°C (sampled, not tested on all parts)

NOTES:

1. Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OLS} of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100 pF), the noise pulse on the ALE line may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.
2. Capacitive loading on Ports 0 and 2 may cause the V_{OH} on ALE and PSEN to momentarily fall before the 0.9 V_{CC} specification when the address bits are stabilizing.
3. Power DOWN I_{CC} is measured with all output pins disconnected; EA = Port 0 = V_{CC}; XTAL2 N.C.; RST = V_{SS}; DB = V_{CC}; $\overline{WR}$ = $\overline{RD}$ = $\overline{DACK}$ = $\overline{CS}$ = A0 = A1 = A2 = V_{CC}. Power Down Mode is not supported on the 87C452P.
4. I_{CC} is measured with all output pins disconnected; XTAL1 driven with TCLCH, TCHCL = 5 ns, V_{IL} = V_{SS} + 0.5V, V_{IH} = V_{CC} − 0.5V; XTAL2 N.C.; EA = RST = Port 0 = V_{CC}; $\overline{WR}$ = $\overline{RD}$ = $\overline{DACK}$ = $\overline{CS}$ = A0 = A1 = A2 = V_{CC}. I_{CC} would be slightly higher if a crystal oscillator is used.
5. Idle I_{CC} is measured with all output pins disconnected; XTAL1 driven with TCLCH, TCHCL = 5 ns, V_{IL} = V_{SS} + 0.5V, V_{IH} = V_{CC} − 0.5V; XTAL2 N.C.; Port 0 = V_{CC}; EA = RST = V_{SS}; $\overline{WR}$ = $\overline{RD}$ = $\overline{DACK}$ = $\overline{CS}$ = A0 = A1 = A2 = V_{CC}.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has 5 characters. The first character is always a 'T' (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for:

A:	Address.
C:	Clock.
D:	Input data.
H:	Logic level HIGH.
I:	Instruction (program memory contents).
L:	Logic level LOW, or ALE.
P:	PSEN.
Q:	Output data.
R:	READ signal.
T:	Time.
V:	Valid.
W:	WRITE signal.
X:	No longer a valid logic level.
Z:	Float.

EXAMPLE

TAVLL = Time for Address Valid to ALE Low.
TLLPL = Time for ALE Low to $\overline{PSEN}$ Low.

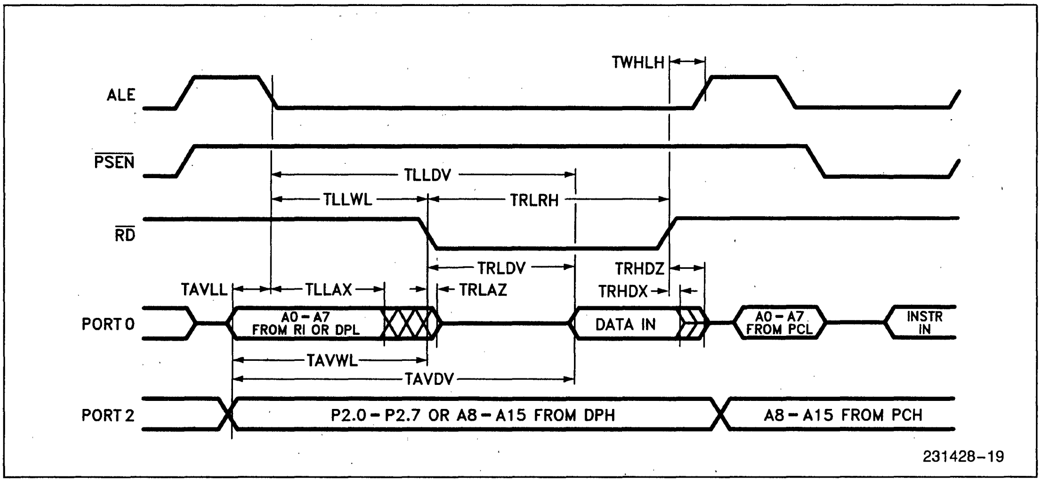
A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±10%, V_{SS} = 0V, Load Capacitance for Port 0, ALE, and PSEN = 100 pF, Load Capacitance for All Other Outputs = 80 pF

EXTERNAL PROGRAM AND DATA MEMORY CHARACTERISTICS

Symbol	Parameter	14 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency	3.5	14			MHz
TLHLL	ALE Pulse Width	103		2TCLCL−40		ns
TAVLL	Address Valid to ALE Low (Note 1)	25		TCLCL−55		ns
TLLAX	Address Hold after ALE Low	36		TCLCL−35		ns
TLLIV	ALE Low to Valid Instr In		185		4TCLCL−100	ns
TLLPL	ALE Low to $\overline{\text{PSEN}}$ Low	31		TCLCL−40		ns
TPLPH	$\overline{\text{PSEN}}$ Pulse Width	169		3TCLCL−45		ns
TPLIV	$\overline{\text{PSEN}}$ Low to Valid Instr In		110		3TCLCL−105	ns
TPXIX	Input Instr Hold after $\overline{\text{PSEN}}$	0		0		ns
TPXIZ	Input Instr Float after $\overline{\text{PSEN}}$ (Note 1)		57		TCLCL−25	ns
TAVIV	Address to Valid Instr In		252		5TCLCL−105	ns
TPLAZ	$\overline{\text{PSEN}}$ Low to Address Float		10		10	ns
TRLRH	$\overline{\text{RD}}$ Pulse Width	329		6TCLCL−100		ns
TWLWH	$\overline{\text{WR}}$ Pulse Width	329		6TCLCL−100		ns
TRLDV	$\overline{\text{RD}}$ Low to Valid Data In		192		5TCLCL−165	ns
TRHDX	Data Hold after $\overline{\text{RD}}$	0		0		ns
TRHDZ	Data Float after $\overline{\text{RD}}$		73		2TCLCL−70	ns
TLLDV	ALE Low to Valid Data In		422		8TCLCL−150	ns
TAVDV	Address to Valid Data In		478		9TCLCL−165	ns
TLLWL	ALE Low to $\overline{\text{RD}}$ or $\overline{\text{WR}}$ Low	164	264	3TCLCL−50	3TCLCL+50	ns
TAVWL	Address Valid to $\overline{\text{RD}}$ or $\overline{\text{WR}}$ Low	156		4TCLCL−130		ns
TQVWX	Data Valid to $\overline{\text{WR}}$ Transition	11		TCLCL−60		ns
TWHQX	Data Hold after $\overline{\text{WR}}$	21		TCLCL−50		ns
TRLAZ	$\overline{\text{RD}}$ Low to Address Float		0		0	ns
TWHLH	$\overline{\text{RD}}$ or $\overline{\text{WR}}$ High to ALE High	31	111	TCLCL−40	TCLCL+40	ns
TQVWH	Data Valid to $\overline{\text{WR}}$ (Setup Time)	350		7TCLCL−150		ns

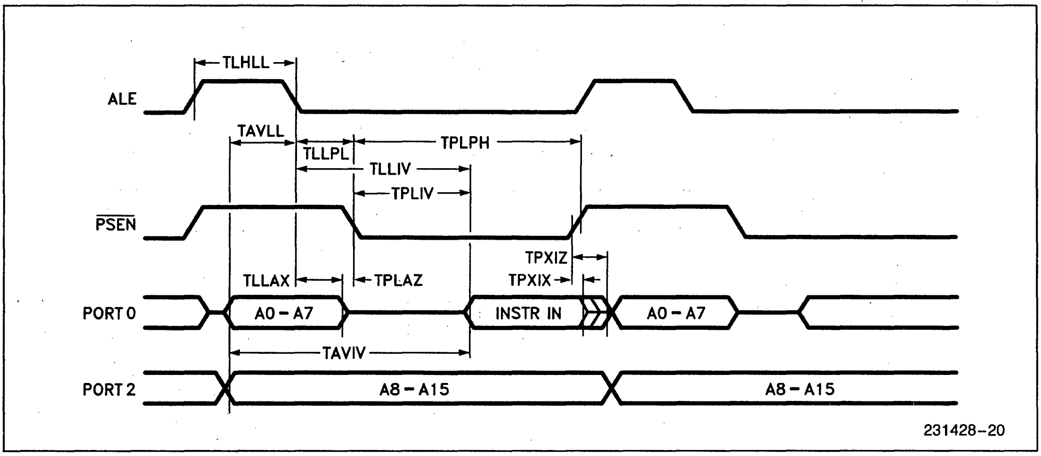
NOTE:
1. Use the value of 14 MHz specification or variable oscillator specification, whichever is greater.

EXTERNAL DATA MEMORY READ CYCLE



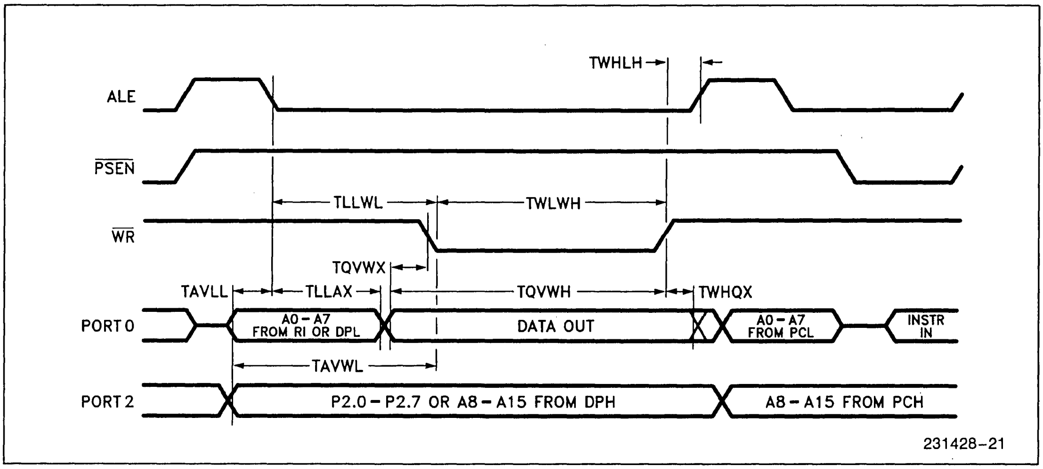
231428–19

EXTERNAL PROGRAM MEMORY READ CYCLE

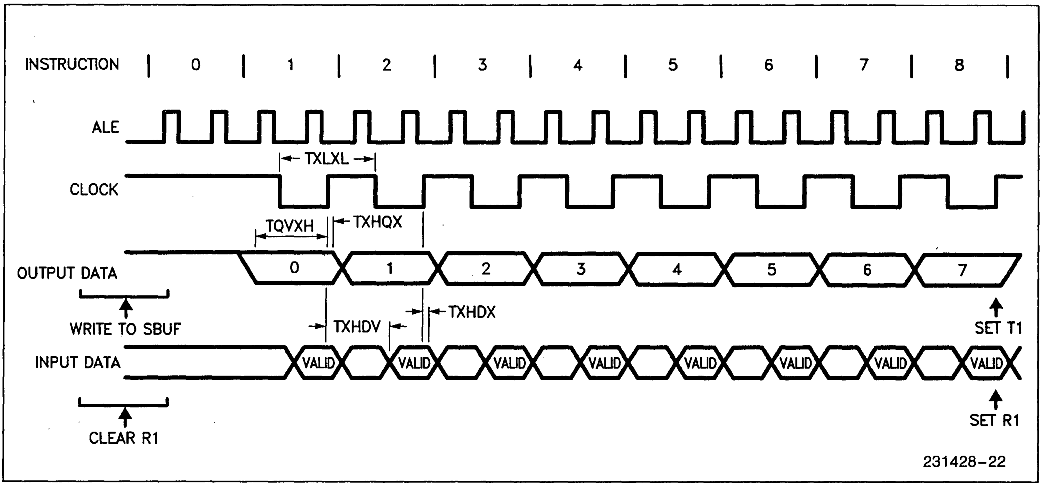


231428–20

EXTERNAL DATA MEMORY WRITE CYCLE



SHIFT REGISTER MODE TIMING WAVEFORMS



EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	14	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

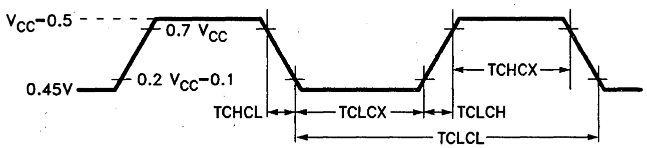
NOTE:
External clock timings are sampled, not tested on all parts.

SERIAL PORT TIMING—SHIFT REGISTER MODE

Test Conditions: T_A = 0°C to 70°C; V_{CC} = 5V ±10%; V_{SS} = 0V; Load Capacitance = 80 pF

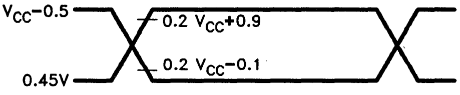
Symbol	Parameter	14 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	857		12TCLCL		ns
TQVXH	Output Data Setup to Clock Rising Edge	581		10TCLCL − 133		ns
TXHQX	Output Data Hold after Clock Rising Edge	26		2TCLCL − 117		ns
TXHDX	Input Data Hold after Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		581		10TCLCL − 133	ns

EXTERNAL CLOCK DRIVE WAVEFORM



231428-23

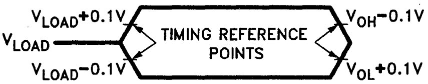
AC TESTING INPUT, OUTPUT WAVEFORMS



231428-24

AC inputs during testing are driven at V_{CC} −0.5V for a logic "1" and 0.45V for a logic "0". Timing measurements are made at V_{IH} min. for a logic "1" and V_{IL} max. for a logic "0".

FLOAT WAVEFORMS

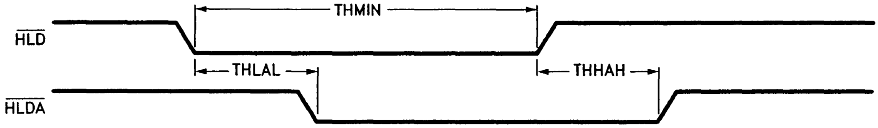


231428-25

For timing purposes a port pin is no longer floating when a 100 mV change from load voltage occurs, and begins to float when a 100 mV change from the loaded V_{OH}/V_{OL} level occurs. I_{OL}/I_{OH} ≥ ±20 mA.

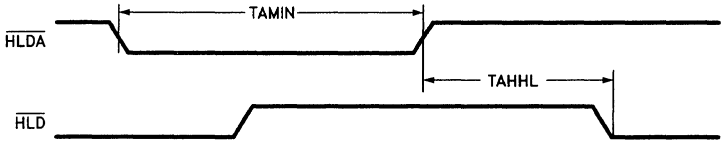
HLD/HLDA WAVEFORMS

Arbiter Mode



231428-26

Requestor Mode



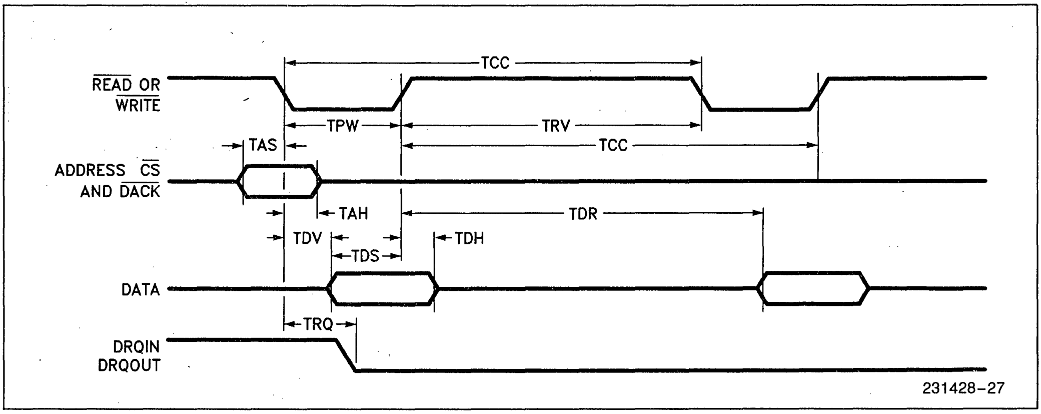
231428-31

HLD/HLDA TIMINGS

Test Conditions: T_A = 0°C to +70°C; V_{CC} = 5V ±10%, V_{SS} = 0V; Load Capacitance = 80 pF

Symbol	Parameter	14 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
THMIN	$\overline{\text{HLD}}$ Pulse Width	386		4TCLCL + 100		ns
THLAL	$\overline{\text{HLD}}$ to $\overline{\text{HLDA}}$ Delay if $\overline{\text{HLDA}}$ is Granted	186	672	4TCLCL − 100	8TCLCL + 100	ns
THHAH	HLD to HLDA Delay	186	672	4TCLCL − 100	8TCLCL + 100	ns
TAMIN	$\overline{\text{HLDA}}$ Pulse Width	386		4TCLCL + 100		ns
TAHHL	$\overline{\text{HLDA}}$ Inactive to $\overline{\text{HLD}}$ Active	186		4TCLCL − 100		ns

HOST PORT WAVEFORMS



HOST PORT TIMINGS

Test Conditions: T_A = 0°C to 70°C; V_{CC} = 5V ±10%; V_{SS} = 0V; Load Capacitance = 80 pF

Symbol	Parameter	14 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
TCC	Cycle Time	429		6TCLCL		ns
TPW	Command Pulse Width	100		100		ns
TRV	Recovery Time	60		60		ns
TAS	Address Setup Time	5		5		ns
TAH	Address Hold Time	30		30		ns
TDS	WRITE Data Setup Time	30		30		ns
TDHW	WRITE Data Hold Time	5		5		ns
TDHR	READ Data Hold Time	5	40	5	40	ns
TDV	READ Active to Read Data Valid Delay		92		92	ns
TDR	WRITE Inactive to Read Data Valid Delay (Applies only to Host Control SFR)		343		4.8TCLCL	ns
TRQ	READ or WRITE Active to DRQIN or DRQOUT Inactive Delay		150		150	ns

REVISION HISTORY

DOCUMENT:	UPI-452 Data Sheet
OLD REVISION NUMBER:	231428-004
NEW REVISION NUMBER:	231428-005

1. Maximum Clock Rate was changed from 16 MHz to 14 MHz. This change is reflected in all Maximum Timing specifications.
2. The proper range of values for ITHR has been changed from [0 to (CBP-2)] to [0 to (CBP-3)] to ensure proper setting of the Input FIFO request for service bit. See the following sections: INPUT FIFO CHANNEL, and INPUT AND OUTPUT FIFO THRESHOLD SFR (ITHR & OTHR).
3. The proper range of values for OTHR has been changed from [1 to {(80H-CBP)-1}] to [2 to {(80-CBP)-1}] to ensure proper setting of the Output FIFO request for service bit. See the following sections: OUTPUT FIFO CHANNEL, FIFO-EXTERNAL HOST INTERFACE FIFO DMA FREEZE MODE, and INPUT AND OUTPUT FIFO THRESHOLD SFR (ITHR & OTHR).
4. The following D.C. Characteristics were deleted from the data sheet:

   $V_{OH} = 0.75^* V_{CC}$ @ $I_{OH} = -25\ \mu A$,

   $V_{OH1} = 0.75^* V_{CC}$ @ $I_{OH} = 150\ \mu A$,

   $V_{OH2} = 3.0V$ @ $I_{OH} = 1$ mA, and

   $I_{CC1} = 15$ mA @ $V_{CC} = 5.5V$ (87C452P).

   See D.C. CHARACTERISTICS TABLE.
5. The parameter descriptions for THHAH and THLAL has been reversed and their maximum specification for clock rates less than 14 MHz has been changed from [4TCLC + 100 ns] to [8TCLC + 100 ns]. See HLD/HLDA TIMINGS.
6. TAMIN specification has been removed from the Arbiter Mode waveform diagram and added to the Requestor Mode waveform diagram. See HLD/HLDA WAVEFORMS.

PRELIMINARY

UPI™-41, 42: 8041AH/8042AH/8741AH/8742AH UNIVERSAL PERIPHERAL INTERFACE 8-BIT SLAVE MICROCONTROLLER

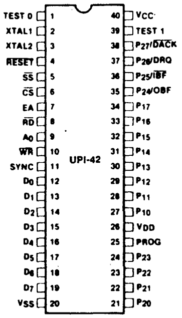
- UPI-41: 6 MHz; UPI-42: 12 MHz
- Pin, Software and Architecturally Compatible with all UPI-41 and UPI-42 Products
- 8-Bit CPU plus ROM/EPROM, RAM, I/O, Timer/Counter and Clock in a Single Package
- 2048 x 8 ROM/EPROM, 256 x 8 RAM on UPI-42, 1024 x 8 ROM/EPROM, 128 x 8 RAM on UPI-41, 8-Bit Timer/Counter, 18 Programmable I/O Pins
- One 8-Bit Status and Two Data Registers for Asynchronous Slave-to-Master Interface
- DMA, Interrupt, or Polled Operation Supported
- Fully Compatible with all Intel and Most Other Microprocessor Families
- Interchangeable ROM and EPROM Versions
- Expandable I/O
- Sync Mode Available
- Over 90 Instructions: 70% Single Byte
- Available in EXPRESS
  — Standard Temperature Range
- int$_e$ligent Programming™ Algorithm
  — Fast EPROM Programming
- Available in 40-Lead Cerdip, 40-Lead Plastic and 44-Lead Plastic Leaded Chip Carrier Packages
  (See Packaging Spec., Order #231369)

The Intel UPI-41 and UPI-42 are general-purpose Universal Peripheral Interfaces that allow the designer to develop customized solutions for peripheral device control.

They are essentially "slave" microcontrollers, or microcontrollers with a slave interface included on the chip. Interface registers are included to enable the UPI device to function as a slave peripheral controller in the MCS™ Modules and iAPX family, as well as other 8-, 16-bit systems.

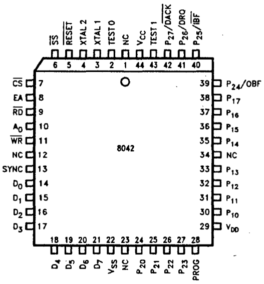
To allow full user flexibility, the program memory is available in ROM, One-Time Programmable EPROM (OTP) and UV-erasable EPROM. All UPI-41 and UPI-42 devices are fully pin compatible for easy transition from prototype to production level designs. These are the memory configurations available.

UPI Device	ROM	EPROM	RAM	Programming Voltage
8042AH	2K	—	256	—
8742AH	—	2K	256	12.5V
8041AH	1K	—	128	—
8741AH	—	1K	128	12.5V



210393-2

Figure 1. DIP Pin Configuration



210393-3

Figure 2. PLCC Pin Configuration

October 1987
Order Number: 210393-004

Table 1. Pin Description

Symbol	DIP Pin No.	PLCC Pin No.	Type	Name and Function
TEST 0, TEST 1	1 39	2 43	I	**TEST INPUTS:** Input pins which can be directly tested using conditional branch instructions. **FREQUENCY REFERENCE:** TEST 1 (T_1) also functions as the event timer input (under software control). TEST 0 (T_0) is used during PROM programming and ROM/EPROM verification. It is also used during Sync Mode to reset the instruction state to S1 and synchronize the internal clock to PH1. See the Sync Mode Section.
XTAL 1, XTAL 2	2 3	3 4	I	**INPUTS:** Inputs for a crystal, LC or an external timing signal to determine the internal oscillator frequency.
$\overline{\text{RESET}}$	4	5	I	**RESET:** Input used to reset status flip-flops and to set the program counter to zero. $\overline{\text{RESET}}$ is also used during EPROM programming and verification.
$\overline{\text{SS}}$	5	6	I	**SINGLE STEP:** Single step input used in conjunction with the SYNC output to step the program through each instruction (EPROM). This should be tied to +5V when not used. This pin is also used to put the device in Sync Mode by applying 12.5V to it.
$\overline{\text{CS}}$	6	7	I	**CHIP SELECT:** Chip select input used to select one UPI microcomputer out of several connected to a common data bus.
EA	7	8	I	**EXTERNAL ACCESS:** External access input which allows emulation, testing and ROM/EPROM verification. This pin should be tied low if unused.
$\overline{\text{RD}}$	8	9	I	**READ:** I/O read input which enables the master CPU to read data and status words from the OUTPUT DATA BUS BUFFER or status register.
A_0	9	10	I	**COMMAND/DATA SELECT:** Address Input used by the master processor to indicate whether byte transfer is data ($A_0 = 0$, F1 is reset) or command ($A_0 = 1$, F1 is set). $A_0 = 0$ during program and verify operations.
$\overline{\text{WR}}$	10	11	I	**WRITE:** I/O write input which enables the master CPU to write data and command words to the UPI INPUT DATA BUS BUFFER.
SYNC	11	13	O	**OUTPUT CLOCK:** Output signal which occurs once per UPI instruction cycle. SYNC can be used as a strobe for external circuitry; it is also used to synchronize single step operation.
D_0–D_7 (BUS)	12–19	14–21	I/O	**DATA BUS:** Three-state, bidirectional DATA BUS BUFFER lines used to interface the UPI microcomputer to an 8-bit master system data bus.
P_{10}–P_{17}	27–34	30–33 35–38	I/O	**PORT 1:** 8-bit, PORT 1 quasi-bidirectional I/O lines. P_{10}–P_{17} access the signature row and security bit.
P_{20}–P_{27}	21–24 35–38	24–27 39–42	I/O	**PORT 2:** 8-bit, PORT 2 quasi-bidirectional I/O lines. The lower 4 bits (P_{20}–P_{23}) interface directly to the 8243 I/O expander device and contain address and data information during PORT 4–7 access. The upper 4 bits (P_{24}–P_{27}) can be programmed to provide interrupt Request and DMA Handshake capability. Software control can configure P_{24} as Output Buffer Full (OBF) interrupt, P_{25} as Input Buffer Full ($\overline{\text{IBF}}$) interrupt, P_{26} as DMA Request (DRQ), and P_{27} as DMA ACKnowledge ($\overline{\text{DACK}}$).
PROG	25	28	I/O	**PROGRAM:** Multifunction pin used as the program pulse input during PROM programming. During I/O expander access the PROG pin acts as an address/data strobe to the 8243. This pin should be tied high if unused.
V_{CC}	40	44		**POWER:** +5V main power supply pin.
V_{DD}	26	29		**POWER:** +5V during normal operation. +12.5V during programming operation. Low power standby supply pin.
V_{SS}	20	22		**GROUND:** Circuit ground potential.

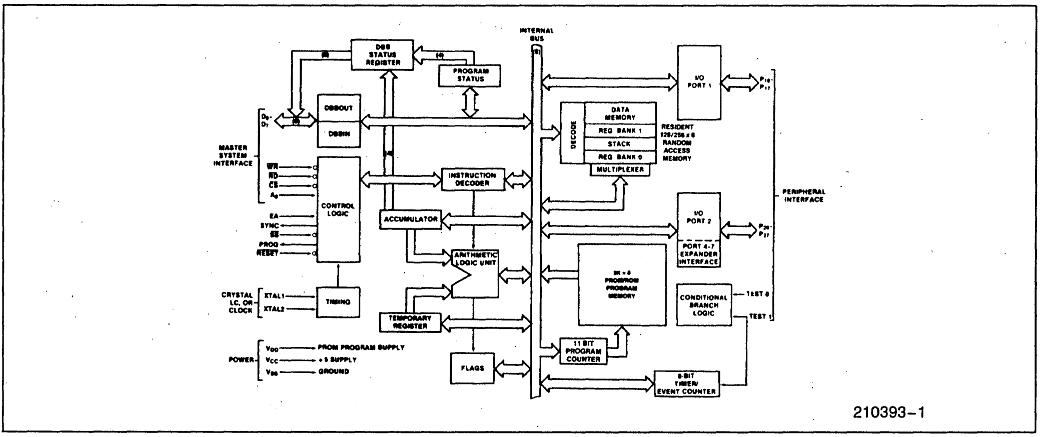
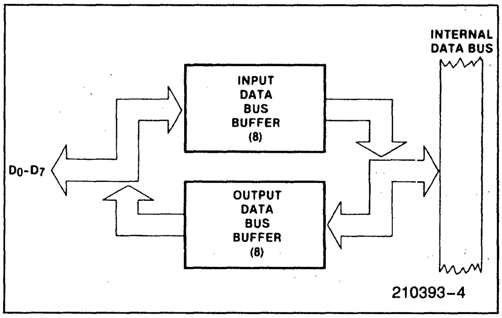


Figure 3. Block Diagram

UPI-41 and UPI-42 FEATURES

1. Two Data Bus Buffers, one for input and one for output. This allows a much cleaner Master/Slave protocol.



2. 8 Bits of Status

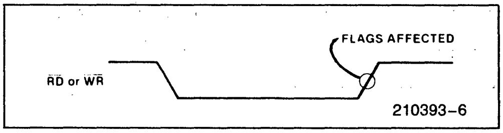
ST_7	ST_6	ST_5	ST_4	F_1	F_0	IBF	OBF
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

ST_4–ST_7 are user definable status bits. These bits are defined by the "MOV STS, A" single byte, single cycle instruction. Bits 4–7 of the acccumulator are moved to bits 4–7 of the status register. Bits 0–3 of the status register are not affected.

MOV STS, A Op Code: 90H

1	0	0	1	0	0	0	0
D_7							D_0

3. $\overline{RD}$ and $\overline{WR}$ are edge triggered. IBF, OBF, F_1 and INT change internally after the trailing edge of $\overline{RD}$ or $\overline{WR}$.

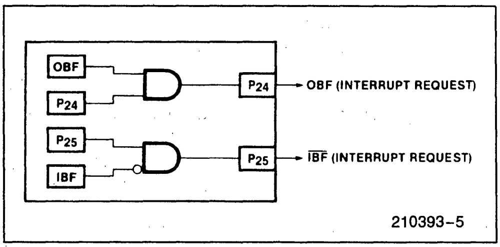


During the time that the host CPU is reading the status register, the UPI is prevented from updating this register or is 'locked out.'

4. P_{24} and P_{25} are port pins or Buffer Flag pins which can be used to interrupt a master processor. These pins default to port pins on Reset.

If the "EN FLAGS" instruction has been executed, P_{24} becomes the OBF (Output Buffer Full) pin. A "1" written to P_{24} enables the OBF pin (the pin outputs the OBF Status Bit). A "0" written to P_{24} disables the OBF pin (the pin remains low). This pin can be used to indicate that valid data is available from the UPI (in Output Data Bus Buffer).

If "EN FLAGS" has been executed, P_{25} becomes the $\overline{IBF}$ (Input Buffer Full) pin. A "1" written to P_{25} enables the $\overline{IBF}$ pin (the pin outputs the inverse of the IBF Status Bit. A "0" written to P_{25} disables the $\overline{IBF}$ pin (the pin remains low). This pin can be used to indicate that the UPI is ready for data.



Data Bus Buffer Interrupt Capability

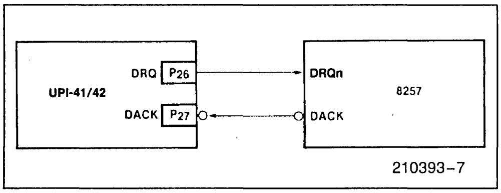
EN FLAGS Op Code: 0F5H

1	1	1	1	0	1	0	1
D_7							D_0

5. P_{26} and P_{27} are port pins or DMA handshake pins for use with a DMA controller. These pins default to port pins on Reset.

   If the "EN DMA" instruction has been executed, P_{26} becomes the DRQ (DMA Request) pin. A "1" written to P_{26} causes a DMA request (DRQ is activated). DRQ is deactivated by DACK•RD, DACK•WR, or execution of the "EN DMA" instruction.

   If "EN DMA" has been executed, P_{27} becomes the $\overline{\text{DACK}}$ (DMA ACKnowledge) pin. This pin acts as a chip select input for the Data Bus Buffer registers during DMA transfers.



DMA Handshake Capability

EN DMA Op Code: 0E5H

D7							D0
1	1	1	0	0	1	0	1

6. When EA is enabled on the UPI, the program counter is placed on Port 1 and the lower three bits of Port 2 (MSB = P_{22}, LSB = P_{10}). On the UPI this information is multiplexed with PORT DATA (see port timing diagrams at end of this data sheet).

7. The 8741AH and 8742AH support the int_eligent Programming Algorithm. (See the Programming Section.)

APPLICATIONS

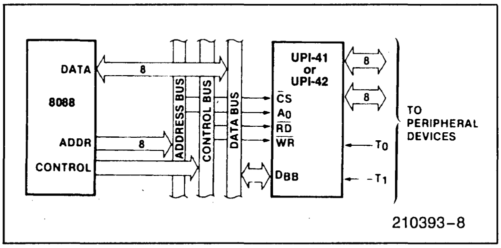


Figure 3. 8088-UPI-41/42 Interface

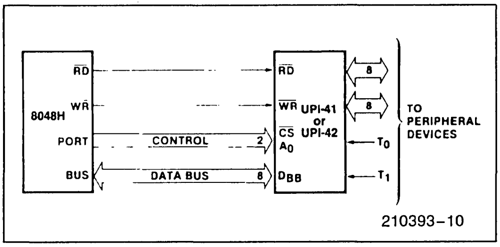


Figure 4. 8048H-UPI-41/42 Interface

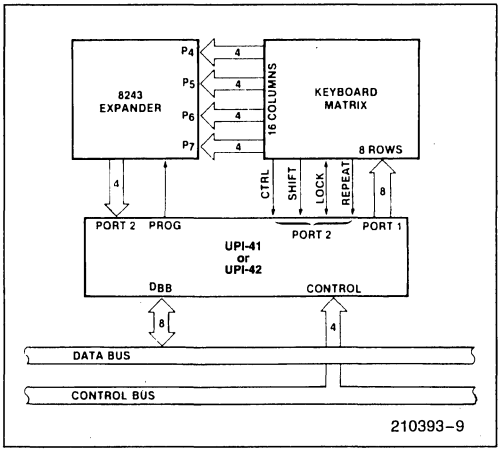


Figure 5. UPI-41/42-8243 Keyboard Scanner

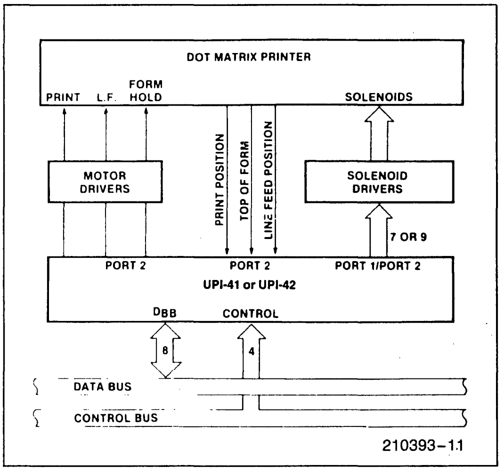


Figure 6. UPI-41/42 80-Column Matrix Printer Interface

PROGRAMMING, VERIFYING, AND ERASING THE 8741AH and 8742AH EPROM

Programming Verification

In brief, the programming process consists of: activating the program mode, applying an address, latching the address, applying data, and applying a programming pulse. Each word is programmed completely before moving on to the next and is followed by a verification step. The following is a list of the pins used for programming and a description of their functions:

Pin	Function
XTAL 1	2 Clock Inputs
$\overline{\text{Reset}}$	Initialization and Address Latching
Test 0	Selection of Program or Verify Mode
EA	Activation of Program/Verify Signature Row/Security Bit Modes
BUS	Address and Data Input Data Output During Verify
P_{20-22}	Address Input
V_{DD}	Programming Power Supply
PROG	Program Pulse Input

WARNING
An attempt to program a missocketed 8741AH or 8742AH will result in severe damage to the part. An indication of a properly socketed part is the appearance of the SYNC clock output. The lack of this clock may be used to disable the programmer.

The Program/Verify sequence is:

1. CS = 5V, V_{CC} = 5V, V_{DD} = 5V, RESET = 0V, A_0 = 0V, TEST 0 = 5V, clock applied or internal oscillator operating, BUS floating, PROG = 5V.
2. Insert 8741AH or 8742AH in programming socket
3. TEST 0 = 0V (select program mode)
4. EA = 12.5V (active program mode)
5. V_{CC} = 6V (programming supply)
6. V_{DD} = 12.5V (programming power)
7. Address applied to BUS and P_{20-22}
8. $\overline{\text{RESET}}$ = 5V (latch address)
9. Data applied to BUS
10. PROG = 5V followed by one 1 ms pulse to 0V
11. TEST 0 = 5V (verify mode)
12. Read and verify data on BUS
13. TEST 0 = 0V
14. Apply overprogram pulse
15. $\overline{\text{RESET}}$ = 0V and repeat from step 6
16. Programmer should be at conditions of step 1 when 8741AH or 8742AH is removed from socket

Please follow the int$_e$ligent Programming flow chart for proper programming procedure.

8741AH/8742AH Erasure Characteristics

The erasure characteristics of the 8741AH/8742AH are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000-4000Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical 8741AH/8742AH in approximately 3 years while it would take approximately one week to cause erasure when exposed to direct sunlight. If the 8741AH/8742AH is to be exposed to these types of lighting conditions for extended periods of time, opaque labels are available from Intel which should be placed over the 8741AH/8742AH window to prevent unintentional erasure.

The recommended erasure procedure for the 8741AH/8742AH is exposure to shortwave ultraviolet light which has a wavelength of 2537Å. The integrated dose (i.e., UV intensity × exposure time) for erasure should be a minimum of 15 w-sec/cm^2. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12,000 μW/cm^2 power rating. The 8741AH/8742AH should be placed within one inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure. Exposure of the 8741AH/8742AH to high intensity UV light for long periods may cause permanent damage.

int$_e$ligent Programming™ Algorithm

The int$_e$ligent Programming Algorithm rapidly programs Intel 8741AH/8742AH EPROMs using an efficient and reliable method particularly suited to the production programming environment. Typical programming time for individual devices is on the order of 10 seconds. Programming reliability is also ensured as the incremental program margin of each byte is continually monitored to determine when it has been successfully programmed. A flowchart of the 8741AH/8742AH int$_e$ligent Programming Algorithm is shown in Figure 7.

The int$_e$ligent Programming Algorithm utilizes two different pulse types: initial and overprogram. The duration of the initial PROG pulse(s) is one millisecond, which will then be followed by a longer overprogram pulse of length 3X msec. X is an iteration counter and is equal to the number of the initial one millisecond pulses applied to a particular 8741AH/8742AH location, before a correct verify occurs. Up to 25 one-millisecond pulses per byte are provided for before the overprogram pulse is applied.

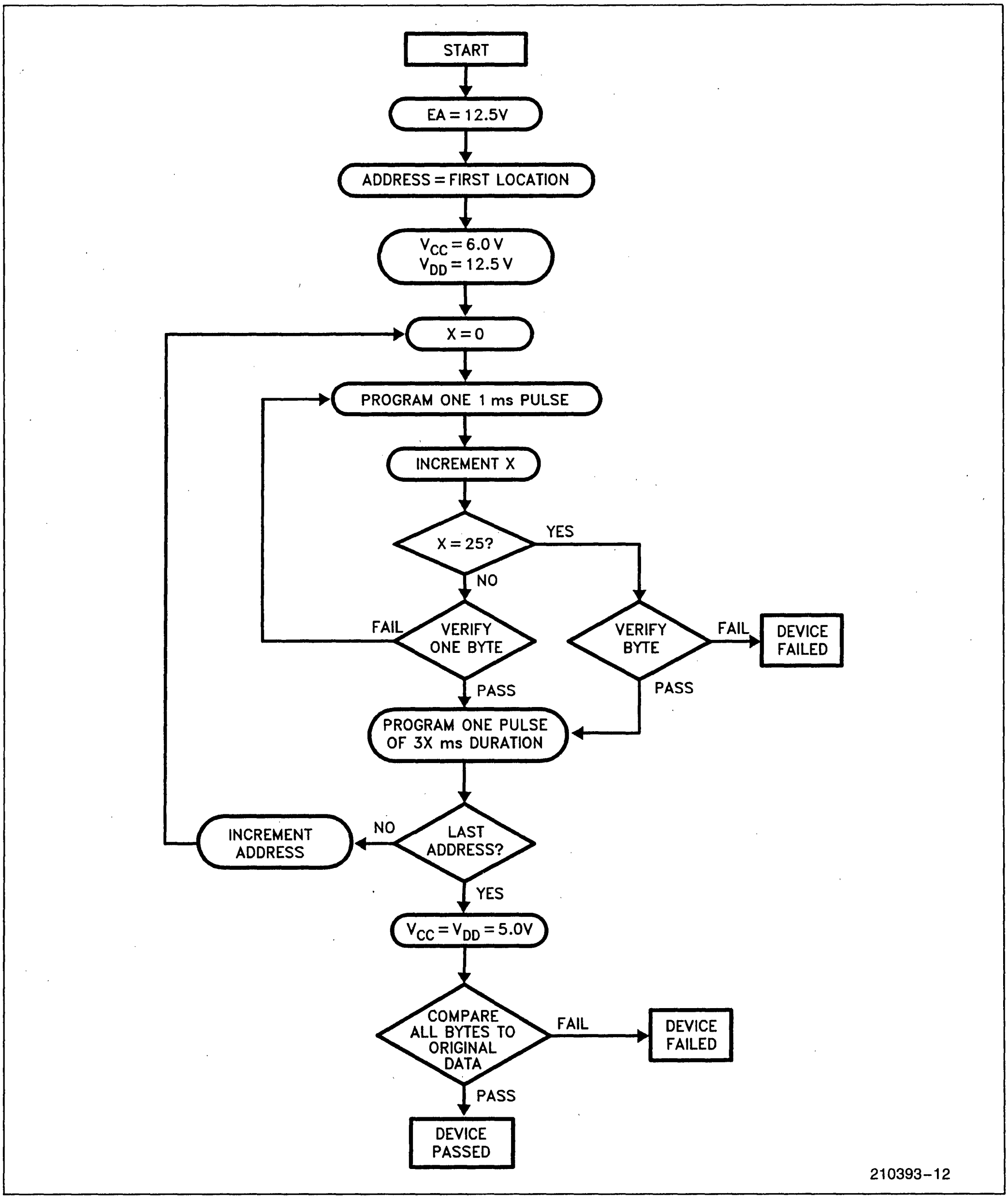

START
EA = 12.5V
ADDRESS = FIRST LOCATION
V_{CC} = 6.0 V
V_{DD} = 12.5 V
X = 0
PROGRAM ONE 1 ms PULSE
INCREMENT X
X = 25?
YES
NO
FAIL
VERIFY ONE BYTE
VERIFY BYTE
FAIL
DEVICE FAILED
PASS
PASS
PROGRAM ONE PULSE OF 3X ms DURATION
INCREMENT ADDRESS
NO
LAST ADDRESS?
YES
V_{CC} = V_{DD} = 5.0V
COMPARE ALL BYTES TO ORIGINAL DATA
FAIL
DEVICE FAILED
PASS
DEVICE PASSED
210393-12

Figure 7

The entire sequence of program pulses and byte verifications is performed at $V_{CC} = 6.0V$ and $V_{DD} = 12.5V$. When the int$_e$ligent Programming cycle has been completed, all bytes should be compared to the original data with $V_{CC} = 5.0$, $V_{DD} = 5V$.

Verify

A verify should be performed on the programmed bits to determine that they have been correctly programmed. The verify is performed with T0 = 5V, $V_{DD} = 5V$, EA = 12.5V, $\overline{SS}$ = 5V, PROG = 5V, A0 = 0V, and $\overline{CS}$ = 5V.

SECURITY BIT

The security bit is a single EPROM cell outside the EPROM array. The user can program this bit with the appropriate access code and the normal programming procedure, to inhibit any external access to the EPROM contents. Thus the user's resident program is protected. There is no direct external access to this bit. However, the security byte in the signature mode has the same address and can be used to check indirectly whether the security bit has been programmed or not. The security bit has no effect on the signature mode, so the security byte can always be examined.

SECURITY BIT PROGRAMMING/ VERIFICATION

Programming

a. Read the security byte of the signature mode. Make sure it is 00H.

b. Apply access code to appropriate inputs to put the device into security mode.

c. Apply high voltage to EA and V_{DD} pins.

d. Follow the programming procedure as per the int$_e$ligent Programming Algorithm with known data on the databus. Not only the security bit, but also the security byte of the signature mode is programmed.

e. Verify that the security byte of the signature mode contains the same data as appeared on the data bus. (If DB0-DB7 = high, the security byte will contain FFH.)

f. Read two consecutive known bytes from the EPROM array and verify that the wrong data are retrieved in at least one verification. If the EPROM can still be read, the security bit may have not been fully programmed though the security byte in the signature mode has.

Verification

Since the security bit address overlaps the address of the security byte of the signature mode, it can be used to check indirectly whether the security bit has been programmed or not. Therefore, the security bit verification is a mere read operation of the security byte of the signature row (1 = security bit programmed; 0 = security bit unprogrammed). Note that during the security bit programming, the reading of the security byte does not necessarily indicate that the security bit has been successfully programmed. Thus, it is recommended that two consecutive known bytes in the EPROM array be read and the wrong data should be read at least once, because it is highly improbable that random data coincides with the correct ones twice.

SIGNATURE MODE

The UPI-41/42 has an additional 32 bytes of EPROM available for Intel and user signatures and miscellaneous purposes. The 32 bytes are partitioned as follows:

A. **Test code/checksum—**This can accommodate up to 25 bytes of code for testing the internal nodes that are not testable by executing from the external memory. The test code/checksum is present on ROMs, and OTPs.

B. **Intel signature**—This allows the programmer to read from the UPI-41/42 the manufacturer of the device and the exact product name. It facilitates automatic device identification and will be present in the ROM and OTP versions. Location 10H contains the manufacturer code. For Intel, it is 89H. Location 11H contains the device code. The code is 43H and 42H for the 8042AH and OTP 8742AH, and 41H and 40H for the 8041AH and OTP 8741AH, respectively.

C. **User signature**—The user signature memory is implemented in the EPROM and consists of 2 bytes for the customer to program his own signature code (for identification purposes and quick sorting of previously programmed materials).

D. **Test signature**—This memory is used to store testing information such as: test data, bin number, etc. (for use in quality and manufacturing control).

E. **Security byte**—This byte is used to check whether the security bit has been programmed (see the security bit section).

The signature mode can be accessed by setting P10 = 0, P11-P17 = 1, and then following the programming and/or verification procedures. The location of the various address partitions are as follows:

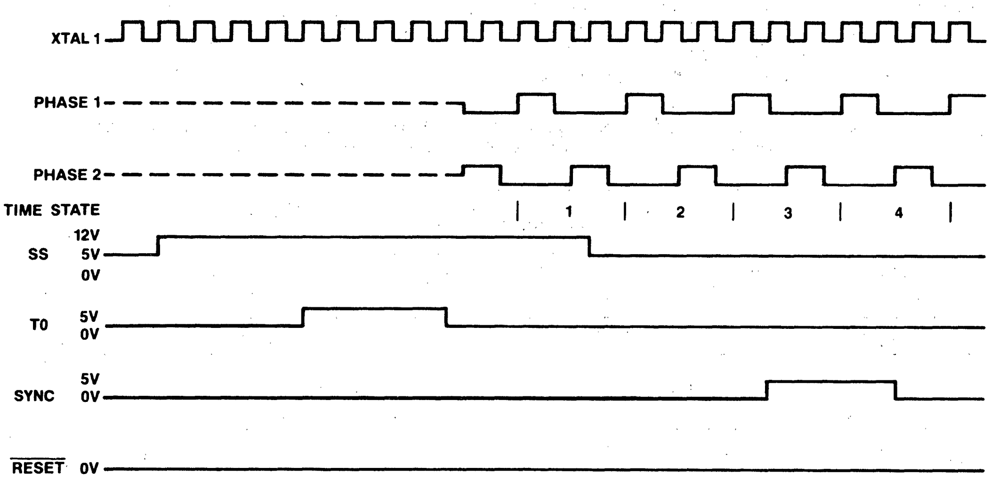
	Address		Device Type	No. of Bytes
Test Code/Checksum	0 16H	0FH 1EH	ROM/OTP	25
Intel Signature	10H	11H	ROM/OTP	2
User Signature	12H	13H	EPROM/OTP	2
Test Signature	14H	15H	ROM/OTP	2
Security Byte	1FH		EPROM/OTP	1

SYNC MODE

The UPI-41/42 has incorporated a new mode. The Sync Mode is provided to ease the design of multiple controller circuits by allowing the designer to force the device into known phase and state time. The Sync Mode may also be utilized by automatic test equipment (ATE) for quick, easy, and efficient synchronizing between the tester and the DUT (device under test).

Sync Mode is enabled when $\overline{SS}$ pin is raised to high voltage level of +12 volts. To begin synchronization, T0 is raised to 5 volts at least four clock cycles after $\overline{SS}$. T0 must be high for at least four X1 clock cycles to fully reset the prescaler and time state generators. T0 may then be brought down during low state of X1. Two clock cycles later, with the rising edge of X1, the device enters into Time State 1, Phase 1. $\overline{SS}$ is then brought down to 5 volts 4 clocks later after T0. RESET is allowed to go high 5 tCY (75 clocks) later for normal execution of code.

SYNC MODE TIMING DIAGRAMS



210393-28

Minimum Specifications
SYNC Operation Time, t_{SYNC} = 3.5 XTAL 1 Clock cycles. Reset Time, t_{RS} = 4 t_{CY}.

NOTE:
The rising and falling edges of T0 should occur during low state of XTAL1 clock.

ACCESS CODE

The following table summarizes the access codes required to invoke the Sync Mode, Signature Mode, and the Security Bit, respectively. Also, the programming and verification modes are included for comparison.

Modes		Control Signals							Data Bus		Access Code
										Port 2	Port 1
		T0	RST	SS	EA	PROG	V_{DD}	V_{CC}	0 1 2 3 4 5 6 7	0 1 2	0 1 2 3 4 5 6 7
Programming Mode		0	0	1	HV	1	V_{DDH}	V_{CC}	Address	Addr	a_0 a_1 X X X X X X
		0	1	1	HV	STB	V_{DDH}	V_{CC}	Data In	Addr	
Verification Mode		0	0	1	HV	1	V_{CC}	V_{CC}	Address	Addr	a_0 a_1 X X X X X X
		1	1	1	HV	1	V_{CC}	V_{CC}	Data Out	Addr	
Sync Mode		STB High	0	HV	0	X	V_{CC}	V_{CC}	X X X X X X X X	X X X	X X X X X X X X
Signature Mode	Prog	0	0	1	HV	1	V_{DDH}	V_{CC}	Addr. (see Sig Mode Table)	0 0 0	0 1 1 1 1 X X 1
		0	1	1	HV	STB	V_{DDH}	V_{CC}	Data In	0 0 0	
	Verify	0	0	1	HV	1	V_{CC}	V_{CC}	Addr. (see Sig Mode Table)	0 0 0	
		1	1	1	HV	1	V_{CC}	V_{CC}	Data Out	0 0 0	
Security Bit/Byte	Prog	0	0	1	HV	1	V_{DDH}	V_{CC}	Address	0 0 0	
		0	1	1	HV	STB	V_{DDH}	V_{CC}	Data In	0 0 0	
	Verify	0	0	1	HV	1	V_{CC}	V_{CC}	Address	0 0 0	
		1	1	1	HV	1	V_{CC}	V_{CC}	Data Out	0 0 0	

NOTES:
1. a_0 = 0 or 1; a_1 = 0 or 1. a_0 must = a_1.

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 . . . . 1.5 W

**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_A = 0°C to +70°C, $V_{CC} = V_{DD}$ = +5V ±10%

Symbol	Parameter	8042/8742		Units	Notes
		Min	Max		
V_{IL}	Input Low Voltage (Except XTAL1, XTAL2, RESET)	−0.5	0.8	V	
V_{IL1}	Input Low Voltage (XTAL1, XTAL2, RESET)	−0.5	0.6	V	
V_{IH}	Input High Voltage (Except XTAL1, XTAL2, RESET)	2.0	V_{CC}	V	
V_{IH1}	Input High Voltage (XTAL1, RESET)	3.5	V_{CC}	V	
V_{IH2}	Input High Voltage (XTAL2)	2.2	V_{CC}	V	
V_{OL}	Output Low Voltage (D_0–D_7)		0.45	V	I_{OL} = 2.0 mA

D.C. CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = V_{DD} = +5V \pm 10\%$ (Continued)

Symbol	Parameter	8042/8742		Units	Notes
		Min	Max		
V_{OL1}	Output Low Voltage ($P_{10}P_{17}$, $P_{20}P_{27}$, Sync)		0.45	V	$I_{OL} = 1.6$ mA
V_{OL2}	Output Low Voltage (PROG)		0.45	V	$I_{OL} = 1.0$ mA
V_{OH}	Output High Voltage (D_0–D_7)	2.4		V	$I_{OH} = -400\ \mu A$
V_{OH1}	Output High Voltage (All Other Outputs)	2.4			$I_{OH} = -50\ \mu A$
I_{IL}	Input Leakage Current (T_0, T_1, RD, WR, CS, A_0, EA)		±10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{OFL}	Output Leakage Current (D_0–D_7, High Z State)		±10	μA	$V_{SS} + 0.45$ $\leq V_{OUT} \leq V_{CC}$
I_{LI}	Low Input Load Current ($P_{10}P_{17}$, $P_{20}P_{27}$)		0.3	mA	$V_{IL} = 0.8V$
I_{LI1}	Low Input Load Current (RESET, SS)		0.2	mA	$V_{IL} = 0.8V$
I_{DD}	V_{DD} Supply Current		20	mA	Typical = 8 mA
$I_{CC} + I_{DD}$	Total Supply Current		135	mA	Typical = 80 mA
I_{DD} Standby	Power Down Supply Current		20	mA	Typical = 8 mA
I_{IH}	Input Leakage Current (P_{10}–P_{17}, P_{20}–P_{27})		100	μA	$V_{IN} = V_{CC}$
C_{IN}	Input Capacitance		10	pF	$T_A = 25°C$ (1)
C_{IO}	I/O Capacitance		20	pF	$T_A = 25°C$ (1)

NOTE:
1. Sampled, not 100% tested.

D.C. CHARACTERISTICS—PROGRAMMING

$T_A = 25°C \pm 5°C$, $V_{CC} = 6V \pm 0.25V$, $V_{DD} = 12.5V \pm 0.5V$

Symbol	Parameter	Min	Max	Units
V_{DDH}	V_{DD} Program Voltage High Level	12	13	V(1)
V_{DDL}	V_{DD} Voltage Low Level	4.75	5.25	V
V_{PH}	PROG Program Voltage High Level	2.0	5.5	V
V_{PL}	PROG Voltage Low Level	−0.5	0.8	V
V_{EAH}	Input High Voltage for EA	12.0	13.0	V(2)
V_{EAL}	EA Voltage Low Level	−0.5	5.25	V
I_{DD}	V_{DD} High Voltage Supply Current		50.0	mA
I_{EA}	EA High Voltage Supply Current		1.0	mA

NOTES:
1. Voltages over 13V applied to pin V_{DD} will permanently damage the device.
2. V_{EAH} must be applied to EA before V_{DDH} and removed after V_{DDL}.
3. V_{CC} must be applied simultaneously or before V_{DD} and must be removed simultaneously or after V_{DD}.

A.C. CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{SS} = 0V$, $V_{CC} = V_{DD} = +5V \pm 10\%$

DBB READ

Symbol	Parameter	Min	Max	Units
t_{AR}	CS, A_0 Setup to RD ↓	0		ns
t_{RA}	CS, A_0 Hold After RD ↑	0		ns
t_{RR}	RD Pulse Width	160		ns
t_{AD}	CS, A_0 to Data Out Delay		130	ns
t_{RD}	RD ↓ to Data Out Delay	0	130	ns
t_{DF}	RD ↑ to Data Float Delay		85	ns

DBB WRITE

Symbol	Parameter	Min	Max	Units
t_{AW}	CS, A_0 Setup to WR ↓	0		ns
t_{WA}	CS, A_0 Hold After WR ↑	0		ns
t_{WW}	WR Pulse Width	160		ns
t_{DW}	Data Setup to WR ↑	130		ns
t_{WD}	Data Hold After WR ↑	0		ns

CLOCK

Symbol	Parameter	Min	Max	Units
t_{CY} (UPI-41)	Cycle Time	2.5	9.20	μs(1)
t_{CYC} (UPI-41)	Clock Period	167	613	ns
t_{CY} (UPI-42)	Cycle Time	1.25	9.20	μs(1)
t_{CYC} (UPI-42)	Clock Period	83.3	613	ns
t_{PWH}	Clock High Time	33		ns
t_{PWL}	Clock Low Time	33		ns
t_R	Clock Rise Time		10	ns
t_F	Clock Fall Time		10	ns

NOTE:
1. $t_{CY} = 15/f(XTAL)$

A.C. CHARACTERISTICS DMA

Symbol	Parameter	Min	Max	Units
t_{ACC}	DACK to WR or RD	0		ns
t_{CAC}	RD or WR to DACK	0		ns
t_{ACD}	DACK to Data Valid	0	130	ns
t_{CRQ}	RD or WR to DRQ Cleared		110	ns(1)

NOTE:
1. $C_L = 150$ pF.

A.C. CHARACTERISTICS—PROGRAMMING

$T_A = 25°C \pm 5°C, V_{CC} = 6V \pm 0.25V, V_{DDL} = +5V \pm 0.25V, V_{DDH} = 12.5V \pm 0.5V$
(8741AH/8742AH ONLY)

Symbol	Parameter	Min	Max	Units
t_{AW}	Address Setup Time to RESET ↑	$4t_{CY}$		
t_{WA}	Address Hold Time After RESET ↑	$4t_{CY}$		
t_{DW}	Data in Setup Time to PROG ↓	$4t_{CY}$		
t_{WD}	Data in Hold Time After PROG ↑	$4t_{CY}$		
t_{PW}	Initial Program Pulse Width	0.95	1.05	ms(1)
t_{TW}	Test 0 Setup Time for Program Mode	$4t_{CY}$		
t_{WT}	Test 0 Hold Time After Program Mode	$4t_{CY}$		
t_{DO}	Test 0 to Data Out Delay		$4t_{CY}$	
t_{WW}	RESET Pulse Width to Latch Address	$4t_{CY}$		
t_r, t_f	PROG Rise and Fall Times	0.5	100	μs
t_{CY}	CPU Operation Cycle Time	2.5	3.75	μs
t_{RE}	RESET Setup Time Before EA ↑	$4t_{CY}$		
t_{OPW}	Overprogram Pulse Width	2.85	78.75	ms(2)
t_{DE}	EA High to V_{DD} High	$1t_{CY}$		

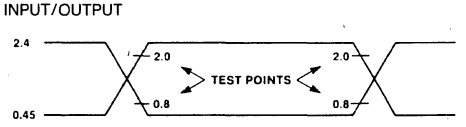
NOTES:
1. Typical Initial Program Pulse width tolerance = 1 ms ±5%.
2. This variation is a function of the iteration counter value, X.
3. If TEST 0 is high, t_{DO} can be triggered by RESET ↑.

A.C. CHARACTERISTICS PORT 2 $T_A = 0°C$ to $+70°C$, $V_{CC} = +5V \pm 10\%$

Symbol	Parameter	$f(t_{CY})$(3)	Min	Max	Units
t_{CP}	Port Control Setup Before Falling Edge of PROG	$1/15\ t_{CY} - 28$	55		ns(1)
t_{PC}	Port Control Hold After Falling Edge of PROG	$1/10\ t_{CY}$	125		ns(2)
t_{PR}	PROG to Time P2 Input Must Be Valid	$8/15\ t_{CY} - 16$		650	ns(1)
t_{PF}	Input Data Hold Time		0	150	ns(2)
t_{DP}	Output Data Setup Time	$2/10\ t_{CY}$	250		ns(1)
t_{PD}	Output Data Hold Time	$1/10\ t_{CY} - 80$	45		ns(2)
t_{PP}	PROG Pulse Width	$6/10\ t_{CY}$	750		ns

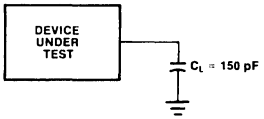
NOTES:
1. $C_L = 80$ pF.
2. $C_L = 20$ pF.
3. $t_{CY} = 1.25$ μs.

A.C. TESTING INPUT/OUTPUT WAVEFORM



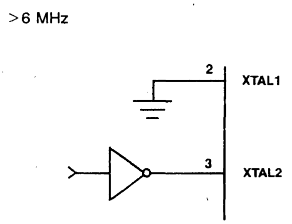
210393-14

A.C. TESTING LOAD CIRCUIT

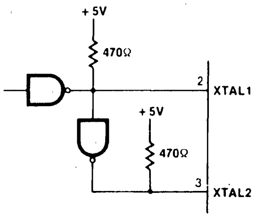


210393-15

DRIVING FROM EXTERNAL SOURCE-TWO OPTIONS



210393-16

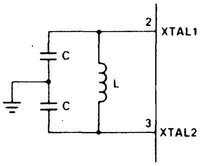


210393-17

Rise and Fall Times Should Not Exceed 10 ns. Resistors to V_{CC} are Needed to Ensure V_{IH} = 3.5V if TTL Circuitry is Used.

LC OSCILLATOR MODE

L	C	NOMINAL
45 H	20 pF	5.2 MHz
120 H	20 pF	3.2 MHz



$$f = \frac{1}{2\pi\sqrt{LC'}}$$

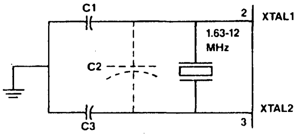
$$C' = \frac{C + 3Cpp}{2}$$

Cpp ≅ 5–10 pF Pin-to-Pin Capacitance

210393-18

Each C Should be Approximately 20 pF, including Stray Capacitance.

CRYSTAL OSCILLATOR MODE



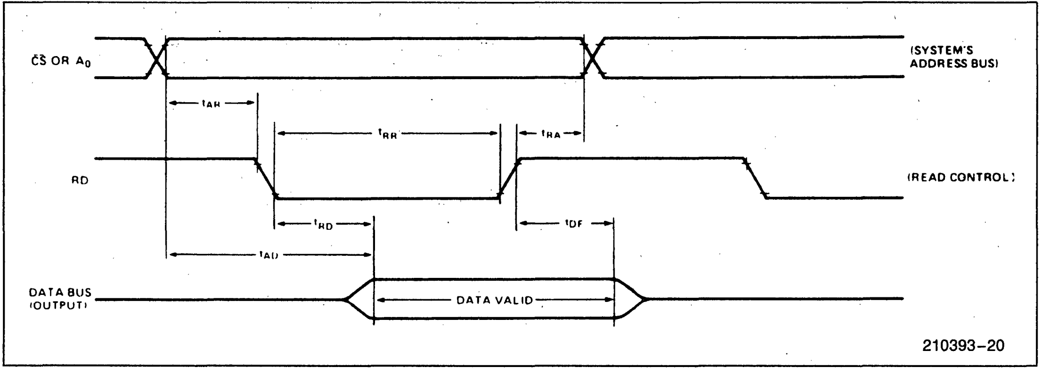
210393-19

C1	5 pF (STRAY 5 pF)
C2	(CRYSTAL + STRAY) 8 pF
C3	20–30 pF INCLUDING STRAY

Crystal Series Resistance Should be Less Than 30Ω at 12 MHz; Less Than 75Ω at 6 MHz; Less Than 180Ω at 3.6 MHz.

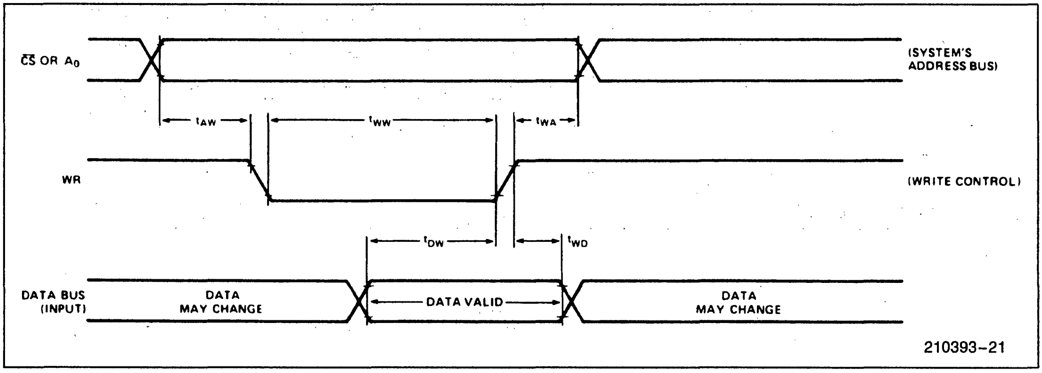
WAVEFORMS

READ OPERATION—DATA BUS BUFFER REGISTER



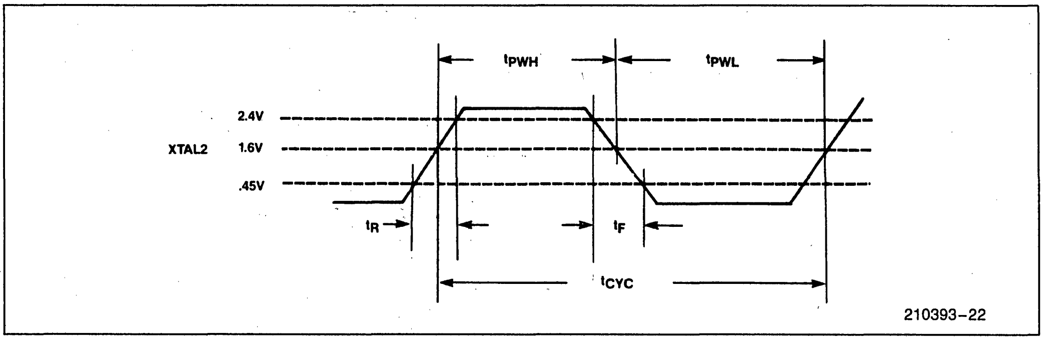
210393-20

WRITE OPERATION—DATA BUS BUFFER REGISTER



210393-21

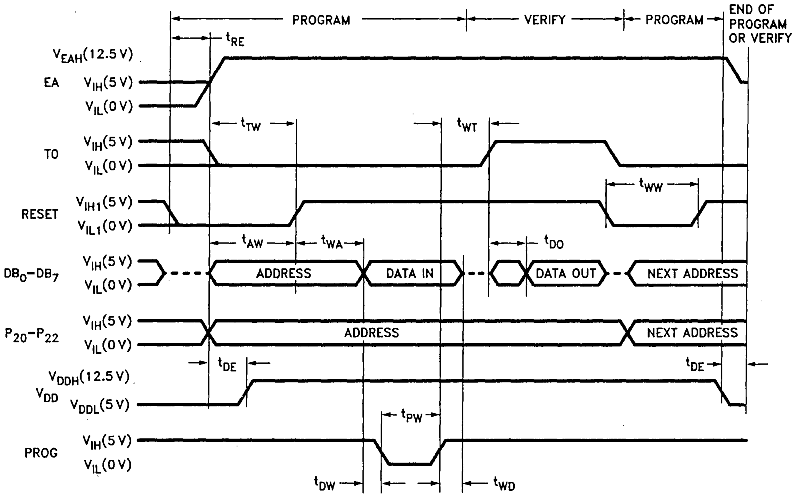
CLOCK TIMING



210393-22

WAVEFORMS (Continued)

COMBINATION PROGRAM/VERIFY MODE

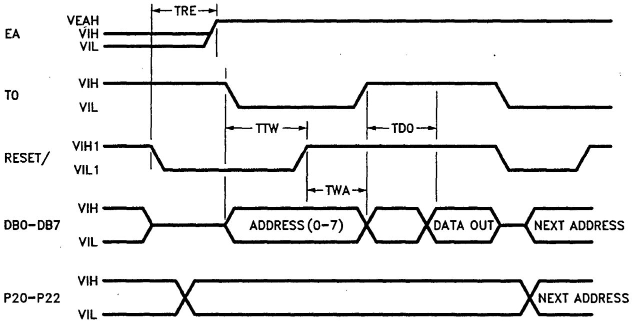


210393-23

NOTES:

1. A$_0$ must be held low (0V) during program/verify modes.
2. For V_{IH}, V_{IH1}, V_{IL}, V_{IL1}, V_{DDH}, and V_{DDL}, please consult the D.C. Characteristics Table.
3. When programming the 8741AH/8742AH, a 0.1 μF capacitor is required across V_{DD} and ground to suppress spurious voltage transients which can damage the device.

VERIFY MODE



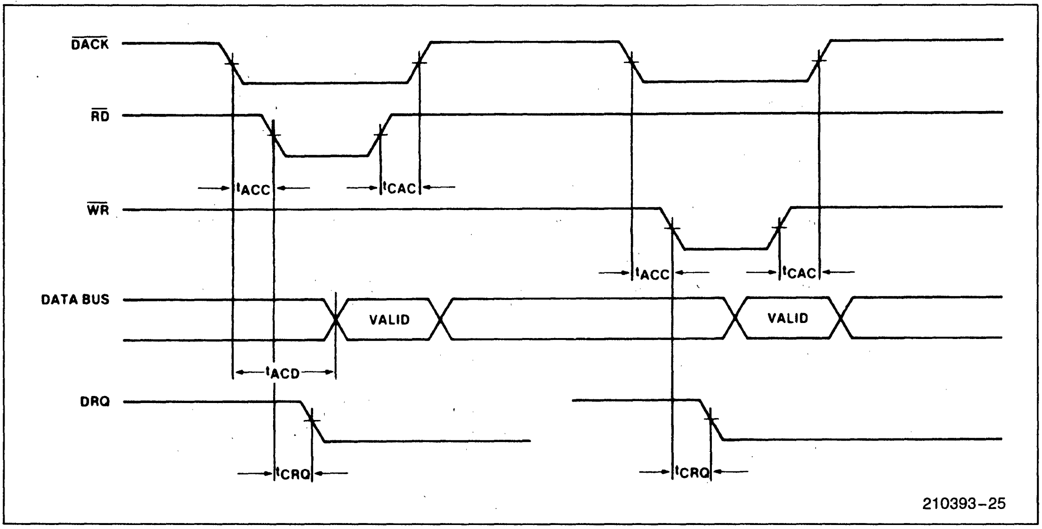
210393-29

NOTES:

1. PROG must float if EA is low.
2. PROG must float or = 5V when EA is high.
3. P_{10}–P_{17} = 5V or must float.
4. P_{24}–P_{27} = 5V or must float.
5. A$_0$ must be held low during programming/verify modes.

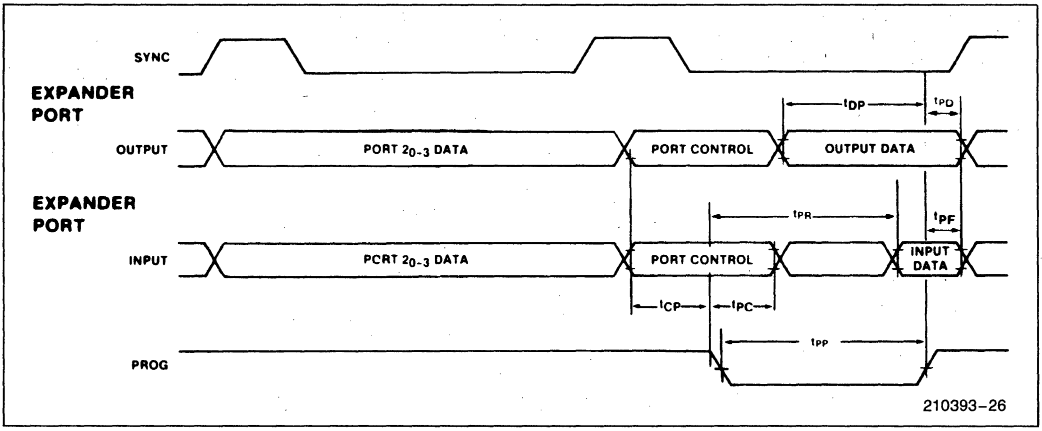
WAVEFORMS (Continued)

DMA



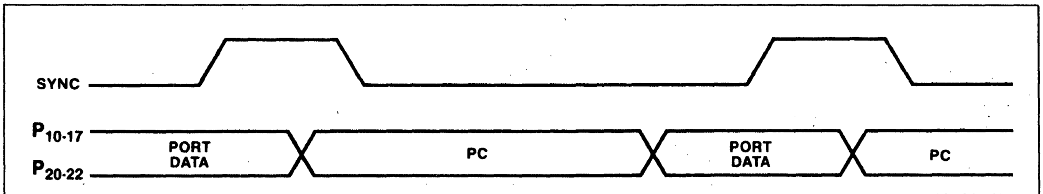
210393-25

PORT 2



210393-26

PORT TIMING DURING EXTERNAL ACCESS (EA)



210393-27

On the Rising Edge of SYNC and EA is Enabled, Port Data is Valid and can be Strobed on the Trailing Edge of Sync the Program Counter Contents are Available.

Table 2. UPI™ Instruction Set

Mnemonic	Description	Bytes	Cycles
ACCUMULATOR			
ADD A, Rr	Add register to A	1	1
ADD A, @Rr	Add data memory to A	1	1
ADD A, #data	Add immediate to A	2	2
ADDC A, Rr	Add register to A with carry	1	1
ADDC A, @Rr	Add data memory to A with carry	1	1
ADDC A, #data	Add immediate to A with carry	2	2
ANL A, Rr	AND register to A	1	1
ANL, A @Rr	AND data memory to A	1	1
ANL A, #data	AND immediate to A	2	2
ORL A, Rr	OR register to A	1	1
ORL, A, @Rr	OR data memory to A	1	1
ORL A, #data	OR immediate to A	2	2
XRL A, Rr	Exclusive OR register to A	1	1
XRL A, @Rr	Exclusive OR data memory to A	1	1
XRL A, #data	Exclusive OR immediate to A	2	2
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal Adjust A	1	1
SWAP A	Swap nibbles of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1
INPUT/OUTPUT			
IN A, Pp	Input port to A	1	2
OUTL Pp, A	Output A to port	1	2
ANL Pp, #data	AND immediate to port	2	2
ORL Pp, #data	OR immediate to port	2	2
IN A, DBB	Input DBB to A, clear IBF	1	1
OUT DBB, A	Output A to DBB, set OBF	1	1
MOV STS, A	A_4–A_7 to Bits 4–7 of Status	1	1
MOVD A, Pp	Input Expander port to A	1	2
MOVD Pp, A	Output A to Expander port	1	2
ANLD Pp, A	AND A to Expander port	1	2
ORLD Pp, A	OR A to Expander port	1	2
DATA MOVES			
MOV A, Rr	Move register to A	1	1
MOV A, @Rr	Move data memory to A	1	1
MOV A, #data	Move immediate to A	2	2
MOV Rr, A	Move A to register	1	1
MOV @Rr, A	Move A to data memory	1	1
MOV Rr, #data	Move immediate to register	2	2
MOV @Rr, #data	Move immediate to data memory	2	2
MOV A, PSW	Move PSW to A	1	1
MOV PSW, A	Move A to PSW	1	1
XCH A, Rr	Exchange A and register	1	1
XCH A, @Rr	Exchange A and data memory	1	1
XCHD A, @Rr	Exchange digit of A and register	1	1
MOVP A, @A	Move to A from current page	1	2
MOVP3, A, @A	Move to A from page 3	1	2
TIMER/COUNTER			
MOV A, T	Read Timer/Counter	1	1
MOV T, A	Load Timer/Counter	1	1
STRT T	Start Timer	1	1
STRT CNT	Start Counter	1	1
STOP TCNT	Stop Timer/Counter	1	1
EN TCNTI	Enable Timer/ Counter Interrupt	1	1
DIS TCNTI	Disable Timer/ Counter Interrupt	1	1
CONTROL			
EN DMA	Enable DMA Handshake Lines	1	1
EN I	Enable IBF Interrupt	1	1
DIS I	Diable IBF Interrupt	1	1
EN FLAGS	Enable Master Interrupts	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
NOP	No Operation	1	1
REGISTERS			
INC Rr	Increment register	1	1
INC @Rr	Increment data memory	1	1
DEC Rr	Decrement register	1	1

Table 2. UPI™ Instruction Set (Continued)

Mnemonic	Description	Bytes	Cycles
SUBROUTINE			
CALL addr	Jump to subroutine	2	2
RET	Return	1	2
RETR	Return and restore status	1	2
FLAGS			
CLR C	Clear Carry	1	1
CPL C	Complement Carry	1	1
CLR F0	Clear Flag 0	1	1
CPL F0	Complement Flag 0	1	1
CLR F1	Clear F1 Flag	1	1
CPL F1	Complement F1 Flag	1	1
BRANCH			
JMP addr	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ Rr, addr	Decrement register and jump	2	2
JC addr	Jump on Carry = 1	2	2
JNC addr	Jump on Carry = 0	2	2
JZ addr	Jump on A Zero	2	2
JNZ addr	Jump on A not Zero	2	2
JT0 addr	Jump on T0 = 1	2	2
JNT0 addr	Jump on T0 = 0	2	2
JT1 addr	Jump on T1 = 1	2	2
JNT1 addr	Jump on T1 = 0	2	2
JF0 addr	Jump on F0 Flag = 1	2	2
JF1 addr	Jump on F1 Flag = 1	2	2
JTF addr	Jump on Timer Flag = 1, Clear Flag	2	2
JNIBF addr	Jump on IBF Flag = 0	2	2
JOBF addr	Jump on OBF Flag = 1	2	2
JBb addr	Jump on Accumula-for Bit	2	2

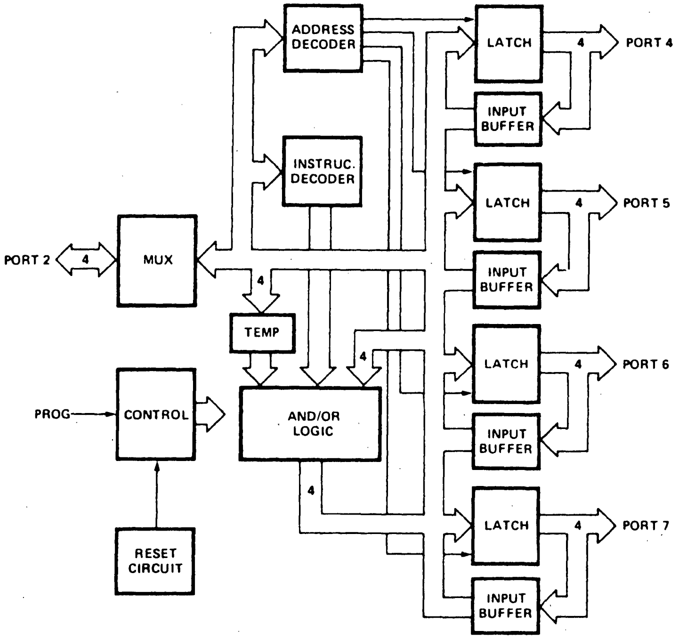
8243
MCS-48® INPUT/OUTPUT EXPANDER

- Low Cost
- Simple Interface to MCS-48® Microcomputers
- Four 4-Bit I/O Ports
- AND and OR Directly to Ports
- 24-Pin DIP
- Single 5V Supply
- High Output Drive
- Direct Extension of Resident 8048 I/O Ports

The Intel® 8243 is an input/output expander designed specifically to provide a low cost means of I/O expansion for the MCS-48 family of single chip microcomputers. Fabricated in 5 volts NMOS, the 8243 combines low cost, single supply voltage and high drive current capability.

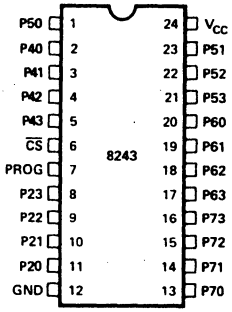
The 8243 consists of four 4-bit bidirectional static I/O ports and one 4-bit port which serves as an interface to the MCS-48 microcomputers. The 4-bit interface requires that only 4 I/O lines of the 8048 be used for I/O expansion, and also allows multiple 8243's to be added to the same bus.

The I/O ports of the 8243 serve as a direct extension of the resident I/O facilities of the MCS-48 microcomputers and are accessed by their own MOV, ANL, and ORL instructions.



231317-1

Figure 1. 8243 Block Diagram



231317-2

Figure 2. 8243 Pin Configuration

Table 1. Pin Description

Symbol	Pin No.	Function
PROG	7	**CLOCK INPUT.** A high to low transition on PROG signifies that address and control are available on P20–P23, and a low to high transition signifies that data is available on P20–P23.
$\overline{CS}$	6	**CHIP SELECT INPUT.** A high on CS inhibits any change of output or internal status.
P20–P23	11–8	Four (4) bit bi-directional port contains the address and control bits on a high to low transition of PROG. During a low to high transition, contains the data for a selected output port if a write operation, or the data from a selected port before the low to high transition if a read operation.
GND	12	0 volt supply.
P40–P43 P50–P53 P60–P63 P70–P73	2–5 1, 23–21 20–17 13–16	Four (4) bit bi-directional I/O ports. May be programmed to be input (during read), low impedance latched output (after write), or a tri-state (after read). Data on pins P20–P23 may be directly written, ANDed or ORed with previous data.
V_{CC}	24	+5 volt supply.

FUNCTIONAL DESCRIPTION

General Operation

The 8243 contains four 4-bit I/O ports which serve as an extension of the on-chip I/O and are addressed as ports 4–7. The following operations may be performed on these ports:

- Transfer Accumulator to Port
- Transfer Port to Accumulator
- AND Accumulator to Port
- OR Accumulator to Port

All communication between the 8048 and the 8243 occurs over Port 2 (P20–P23) with timing provided by an output pulse on the PROG pin of the processor. Each transfer consists of two 4-bit nibbles.

The first containing the "op code" and port address and the second containing the actual 4-bits of data. A high to low transition of the PROG line indicates that address is present while a low to high transition indicates the presence of data. Additional 8243's may be added to the 4-bit bus and chip selected using additional output lines from the 8048/8748/8035.

Power On Initialization

Initial application of power to the device forces input/output ports 4, 5, 6, and 7 to the tri-state and port 2 to the input mode. The PROG pin may be either high or low when power is applied. The first high to low transition of PROG causes device to exit power on mode. The power on sequence is initiated if V_{CC} drops below 1V.

P21	P20	Address Code	P23	P22	Instruction Code
0	0	Port 4	0	0	Read
0	1	Port 5	0	1	Write
1	0	Port 6	1	0	ORLD
1	1	Port 7	1	1	ANLD

Write Modes

The device has three write modes. MOVD Pi, A directly writes new data into the selected port and old data is lost. ORLD Pi, A takes new data, OR's it with the old data and then writes it to the port. ANLD Pi, A takes new data, AND's it with the old data and then writes it to the port. Operation code and port address are latched from the input port 2 on the high to low transition of the PROG pin. On the low to high transition of PROG, data on port 2 is transferred to the logic block of the specified output port.

After the logic manipulation is performed, the data is latched and output. The old data remains latched until new valid outputs are entered.

Read Mode

The device has one read mode. The operation code and port address are latched from the input port 2 on the high to low transition of the PROG pin. As soon as the read operation and port address are decoded, the appropriate outputs are tri-stated, and the input buffers switched on. The read operation is terminated by a low to high transition of the PROG pin. The port (4, 5, 6 or 7) that was selected is switched to the tri-stated mode while port 2 is returned to the input mode.

Normally, a port will be in an output (write mode) or input (read mode). If modes are changed during operation, the first read following a write should be ignored; all following reads are valid. This is to allow the external driver on the port to settle after the first read instruction removes the low impedance drive from the 8243 output. A read of any port will leave that port in a high impedance state.

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.*

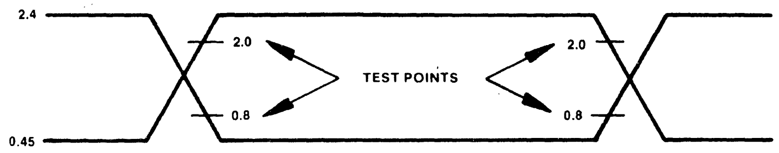
D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Min	Typ	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_{OL1}	Output Low Voltage Ports 4–7			0.45	V	I_{OL} = 4.5 mA*
V_{OL2}	Output Low Voltage Port 7			1	V	I_{OL} = 20 mA
V_{OH1}	Output High Voltage Ports 4–7	2.4			V	I_{OH} = 240 μA
I_{IL1}	Input Leakage Ports 4–7	−10		20	μA	V_{IN} = V_{CC} to 0V
I_{IL2}	Input Leakage Port 2, CS, PROG	−10		10	μA	V_{IN} = V_{CC} to 0V
V_{OL3}	Output Low Voltage Port 2			0.45	V	I_{OL} = 0.6 mA
I_{CC}	V_{CC} Supply Current		10	20	mA	
V_{OH2}	Output Voltage Port 2	2.4			V	I_{OH} = 100 μA
I_{OL}	Sum of all I_{OL} from 16 Outputs			72	mA	4.5 mA Each Pin

*See following graph for additional sink current capability.

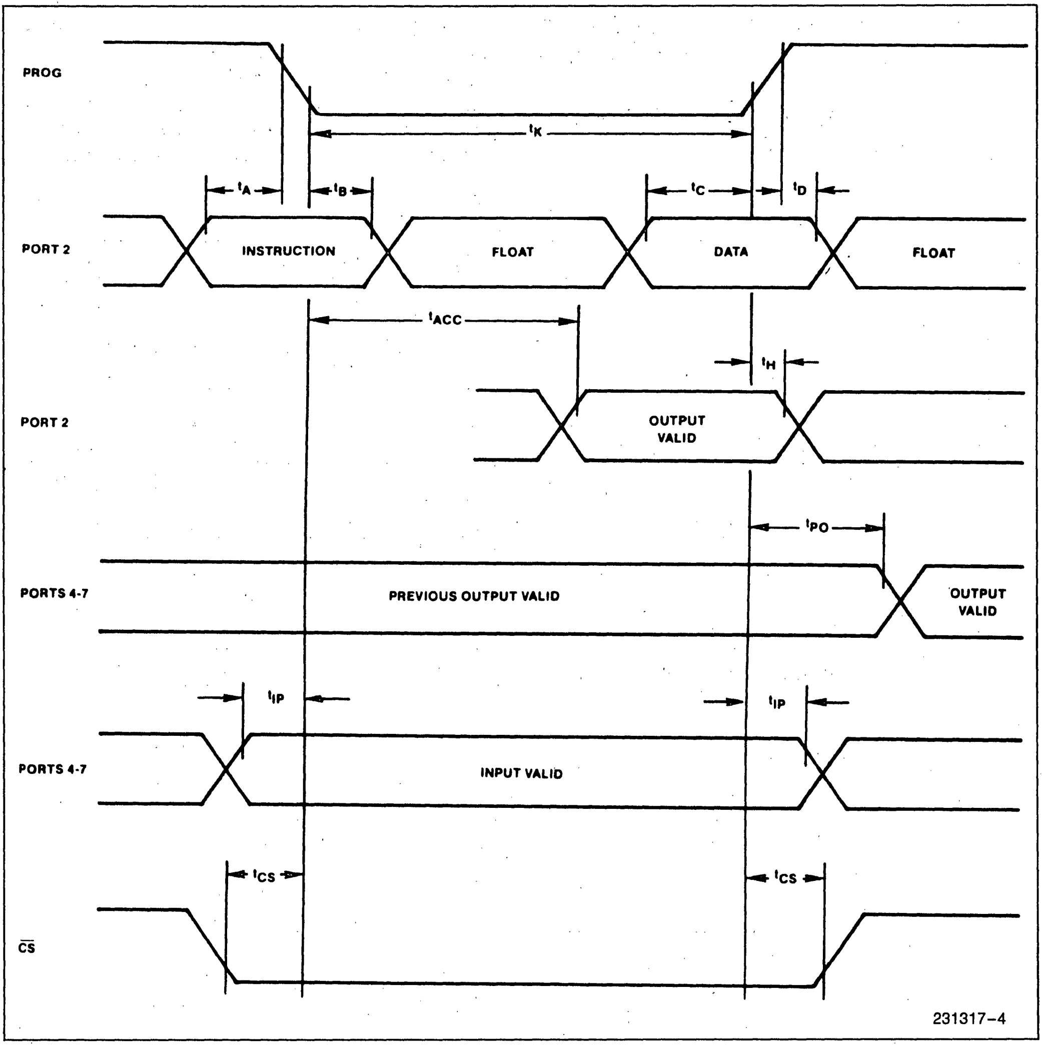
A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Min	Max	Units	Test Conditions
t_A	Code Valid Before PROG	100		ns	80 pF Load
t_B	Code Valid After PROG	60		ns	20 pF Load
t_C	Data Valid Before PROG	200		ns	80 pF Load
t_D	Data Valid After PROG	20		ns	20 pF Load
t_H	Floating After PROG	0	150	ns	20 pF Load
t_K	PROG Negative Pulse Width	700		ns	
t_{CS}	CS Valid Before/After PROG	50		ns	
t_{PO}	Ports 4–7 Valid After PROG		700	ns	100 pF Load
t_{LP1}	Ports 4–7 Valid Before/After PROG	100		ns	
t_{ACC}	Port 2 Valid After PROG		650	ns	80 pF Load



231317–3

WAVEFORMS


PROG
t_K
t_A
t_B
t_C
t_D
PORT 2
INSTRUCTION
FLOAT
DATA
FLOAT
t_{ACC}
t_H
PORT 2
OUTPUT VALID
t_{PO}
PORTS 4-7
PREVIOUS OUTPUT VALID
OUTPUT VALID
t_{IP}
t_{IP}
PORTS 4-7
INPUT VALID
t_{CS}
t_{CS}
$\overline{CS}$
231317-4

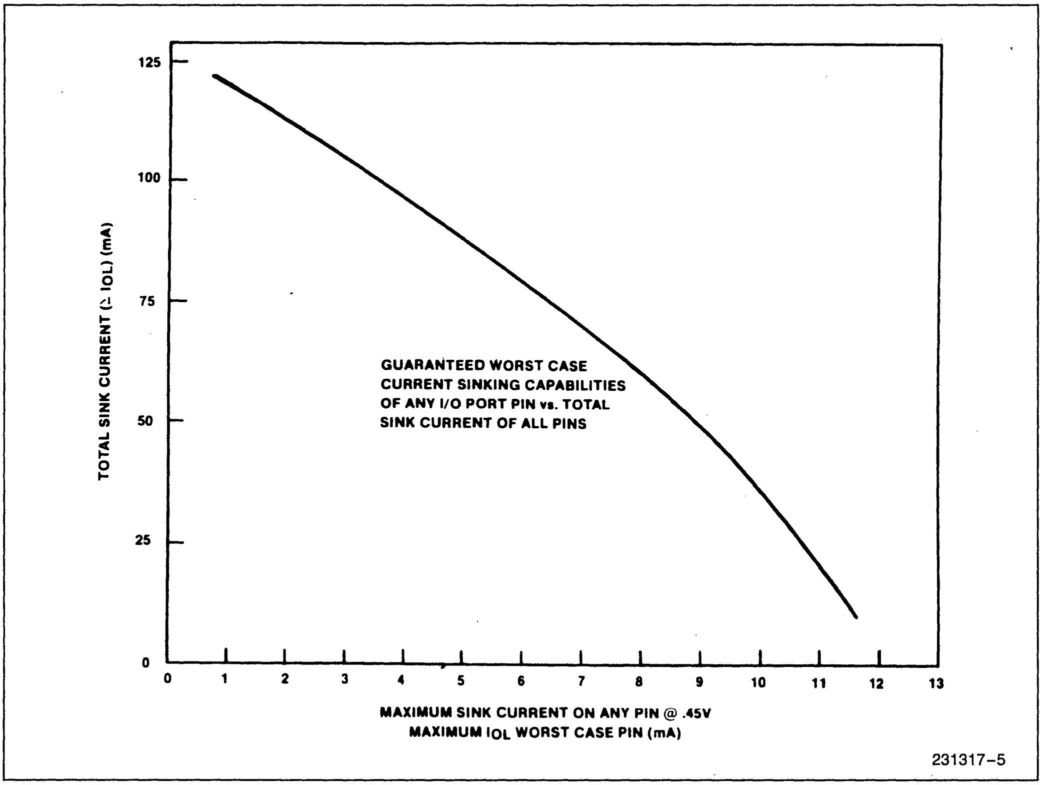


Figure 3

Sink Capability

The 8243 can sink 5 mA @ 0.45V on each of its 16 I/O lines simultaneously. If, however, all lines are not sinking simultaneously or all lines are not fully loaded, the drive capability of any individual line increases as is shown by the accompanying curve.

For example, if only 5 of the 16 lines are to sink current at one time, the curve shows that each of those 5 lines is capable of sinking 9 mA @ 0.45V (if any lines are to sink 9 mA the total I_{OL} must not exceed 45 mA or five 9 mA loads).

Example: How may pins can drive 5 TTL loads (1.6 mA) assuming remaining pins are unloaded?

$I_{OL} = 5 \times 1.6$ mA $= 8$ mA

$\epsilon I_{OL} = 60$ mA from curve

# pins $= 60$ mA $\div$ 8 mA/pin $= 7.5 = 7$

In this case, 7 lines can sink 8 mA for a total of 56 mA. This leaves 4 mA sink current capability which can be divided in any way among the remaining 8 I/O lines of the 8243.

Example: This example shows how the use of the 20 mA sink capability of Port 7 affects the sinking capability of the other I/O lines.

An 8243 will drive the following loads simultaneously.

2 loads—20 mA @ 1V (port 7 only)

8 loads—4 mA @ 0.45V

6 loads—3.2 mA @ 0.45V

Is this within the specified limits?

$\epsilon I_{OL} = (2 \times 20) + (8 \times 4) + (6 \times 3.2) = 91.2$ mA. From the curve: for $I_{OL} = 4$ mA, $\epsilon I_{OL} \approx 93$ mA. Since 91.2 mA < 93 mA the loads are within specified limits.

Although the 20 mA @ 1V loads are used in calculating ϵI_{OL}, it is the largest current required @ 0.45V which determines the maximum allowable ϵI_{OL}.

NOTE:

A 10 KΩ to 50 KΩ pullup resistor to +5V should be added to 8243 outputs when driving to 5V CMOS directly.

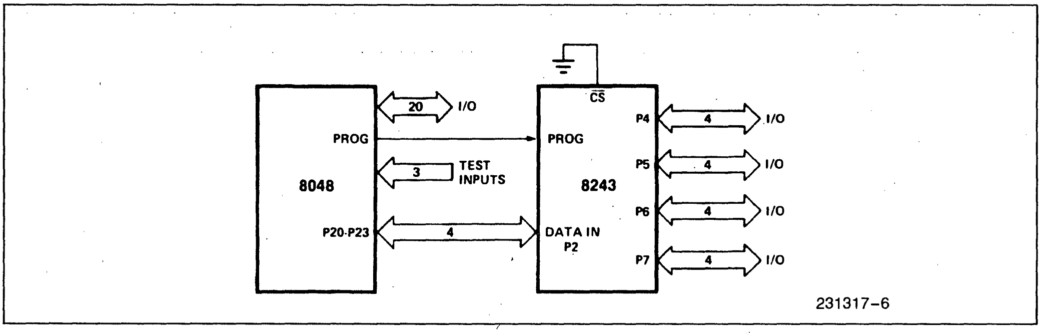


Figure 4. Expander Interface

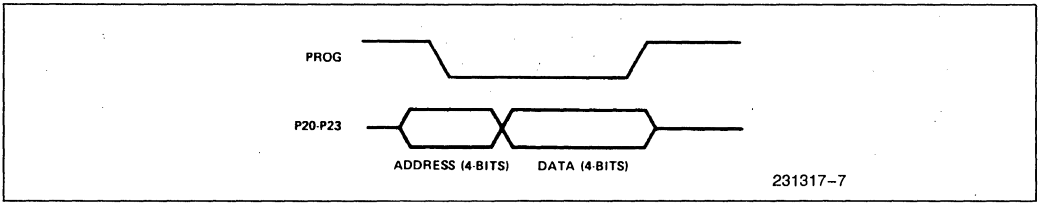


Figure 5. Output Expander Timing

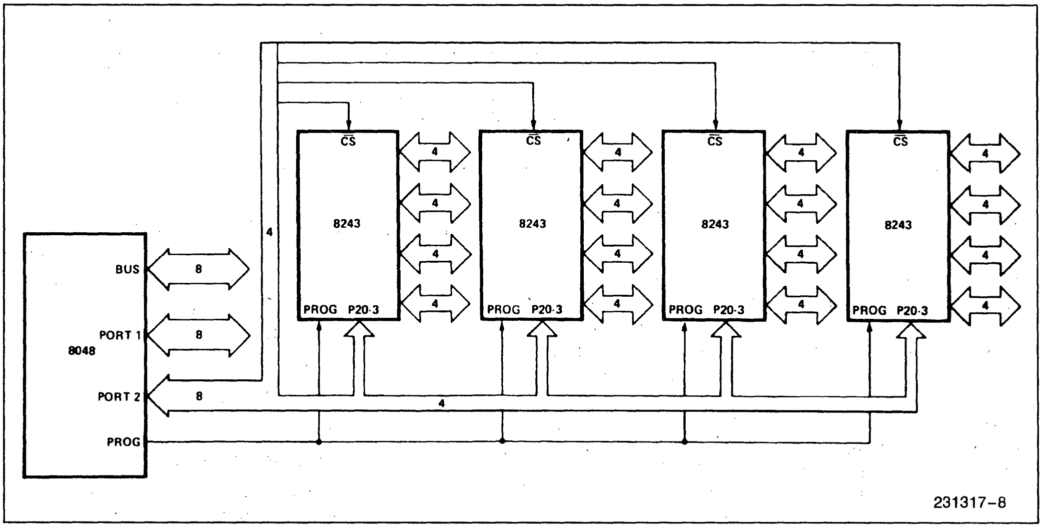


Figure 6. Using Multiple 8243's

June 1985

Applications Using the 8042 UPI™ Microcontroller

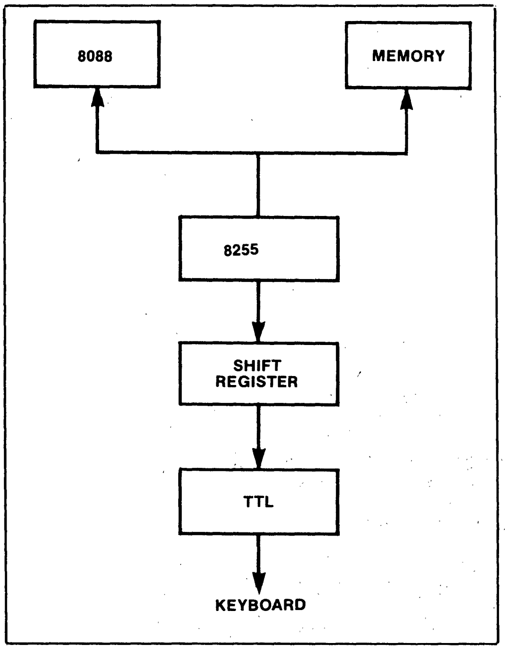
1. The 8042 in the IBM PC/AT
2. Using the 8042 vs. using microcontrollers
3. Custom serial protocol with the 8042

Order Number: 231600-001

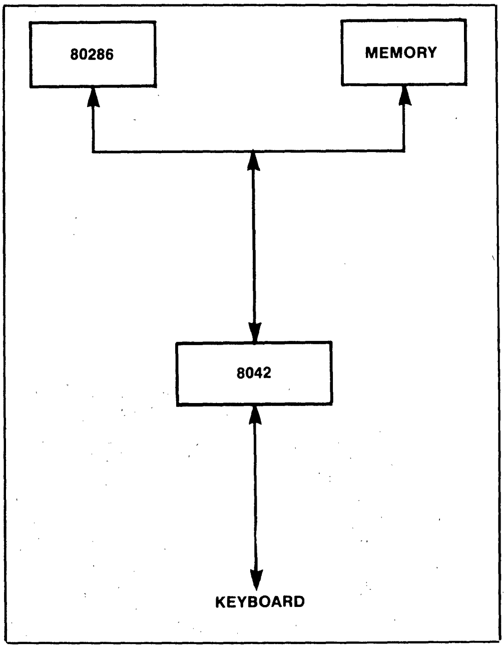
APPLICATION #1
THE 8042 UPI™ MICROCONTROLLER IN THE IBM PC/AT

The following example is an important application of UPIs but there are many more. It is truly a universal device that can be *customized* to all those "non-standard" peripheral control problems. Applications are limited only by imagination. Think UPIs for non-standard peripherals!!

IBM PC/AT (BEFORE) . . .



. . . IBM PC/AT (AFTER)



NEW FUNCTIONS

The 8042 also brings new functions to the PC/AT:

- Keyboard lockup security (front panel key)
- CRT type input to the system
- Diagnostics/self testing of keyboard interface
- Parity check and retry
- PC and PC/AT keyboard interchangeability
- Reset CPU to compatible mode
- Address wrap-around protect in compatible mode

THE FUTURE IS THE KEY

Modifications and upgrades are easy because of the 8042's programmable flexibility and power:

- Change keyboard scan codes (in 8042 ROM)
- Increase functionality of keyboard interface through software and/or unused I/O lines on 8042
- Control other PC/AT functions with these I/O lines

Summary

In short, IBM used the 8042 since it:

- Offloads housekeeping details from the CPU
- Facilitates modular system design
- Offers a customized peripheral
- Provides a clean, efficient upgrade path

These benefits can apply to many of your applications also.

APPLICATION #2
USING THE 8042 VS. USING MICROCONTROLLERS

PROBLEM

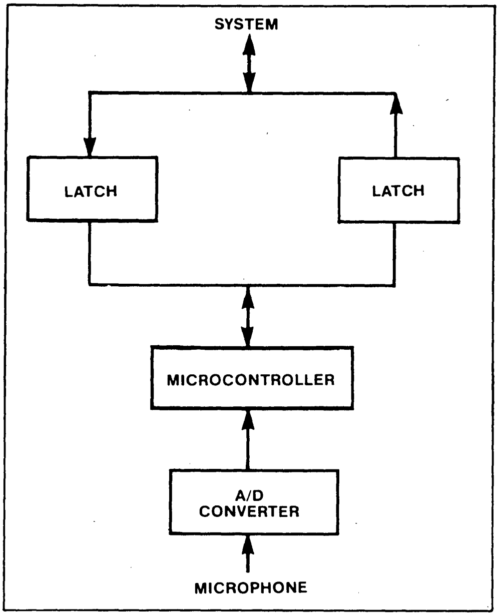
What do you do when you're making SBX and VME modules for a voice digitizing board and you need:

1) an interface to an A/D Converter with
2) 12 MHz operation,
3) an absolute minimum chip count, and
4) very low cost (for the PC market).

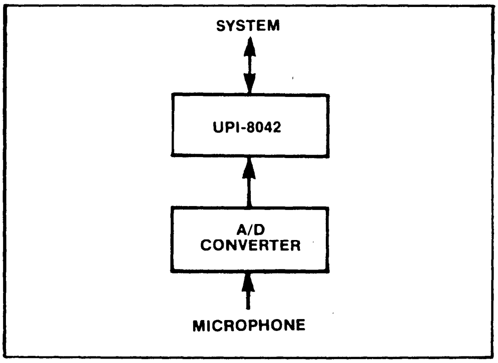
A leading vendor was faced with exactly this problem. Here is what they started with, and the bottom figure shows how they improved things with the 8042 UPI™ microcontroller.

SOLUTION

BEFORE . . .



AFTER . . .



The 8042 integrates two latches and the microcontroller into a single-chip solution.

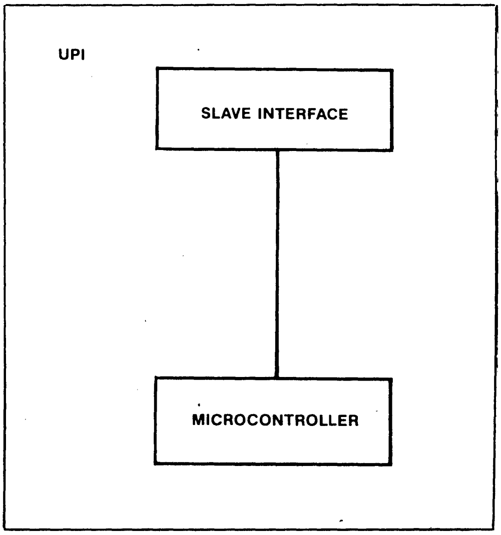
WHY THE SWITCH

After studying the four requirements for this module, it is easy to see why they switched. The first two (A/D interface and 12 MHz) were met by both solutions. However, it is clear the second alternative is much better on board space and on overall cost. There are fewer chips, so they could avoid a multi-layer board and thus save a lot in total cost. Actual chip costs are within 10% of each other (a typical microcontroller like a Z8 or 6801 plus 2 latches compared to an 8042), and they do the same thing.

WHAT'S THE DIFFERENCE

People tend to think of microcontrollers whenever there is a "non-standard" device to control. CRTs, disk drives and DRAMs all have dedicated controllers, but printers, front panels, displays and keyboards don't, because they are all "non-standard" devices. Microcontrollers can be customized to these applications.

The problem is when the device is a "slave" or a peripheral, regular microcontrollers need the extra circuitry shown previously. That's why we invented UPIs. They are simply microcontrollers with the slave interface built in. They are, therefore, more efficient to use in peripheral-type configurations.



UPIs may be misunderstood because of the funny name. They shouldn't be. It's really simple. When faced with non-standard device control, think microcontrollers.

If it's a master-only configuration, think regular microcontrollers. If it's a slave/peripheral configuration, think UPIs.

APPLICATION #3
CUSTOM SERIAL PROTOCOL WITH THE 8042 UPI™ MICROCONTROLLER

BACKGROUND

The 8042 UPI Microcontroller, because of its programmability is being used everywhere, and here is another example. A leading board vendor was designing a communications concentrator board. They wanted a way to:

1) interface 8 serial channels to a minicomputer bus
2) operate these at 4800 baud
3) use one board
4) provide a custom serial protocol that
   - — communicated commands and data packets
   - — assembled the data packets
   - — provided handshaking signals
   - — checked for framing, timing, parity, noise, modem and synchronization errors
   - — provided self-test diagnostics

THE 8042 SOLUTION

There certainly wasn't an "off-the-shelf" chip that would satisfy the above requirements, and using the main CPU would have caused tremendous system performance degradation. They needed all of these features to offer a competitive product, so they looked to the 8042 UPI Microcontroller. Since the speed requirements were not too great (4800 baud), they could implement the protocol in software. The 8042's programmability gave them all the flexibility needed to incorporate the formatting, handshaking and error checking desired. Moreover, the on-chip slave interface made communication with the minicomputer's bus a snap.

SUMMARY

In short, the 8042 allowed this company to implement a custom serial communication protocol that in turn allowed them to offer a customized, competitive interface board to their customers.

➧**Don't some of your applications need customized interfaces?**

APPLICATION NOTE

AP-161

November 1986

Complex Peripheral Control with the UPI-42AH

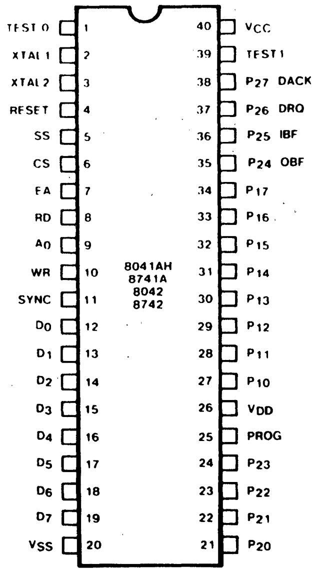
CHRISTOPHER SCOTT
TECHNICAL MARKETING ENGINEER

Order Number: 230795-002

INTRODUCTION

The UPI-42AH represents a significant growth in UPI capabilities resulting in a broader spectrum of applications. The UPI-42AH incorporates twice the EPROM/ROM of the UPI-41AH, 2048 vs 1024 bytes, 256 bytes of RAM, and operates at a maximum speed twice that of the UPI-41AH, i.e., 12MHz vs 6MHz. The ROM based 8042 and the EPROM based 8742AH provide more highly integrated solutions for complex stepping motor and dot matrix printer applications. Those applications previously requiring a microprocessor plus a UPI chip can now be implemented entirely with the UPI-42AH.

The software features of the UPI-42AH, such as indirect Data and Program Memory addressing, two independent and selectable 8 byte register banks, and directly software testable I/O pins, greatly simplify the external interface and software flow. The software and hardware design of the UPI-42AH allows a complex peripheral to be controlled with a minimum of external hardware.



230795-1

Figure 1. UPI-42AH Pin Configuration

Many microcomputer systems need real time control of peripheral devices such as a printer, keyboard, complex motor control or process control. These medium speed but still time consuming tasks require a fair amount of system software overhead. This processing burden can be reduced by using a dedicated peripheral control processor.

Until recently, the dedicated control processor approach was usually not cost effective due to the large number of components needed; CPU, RAM, ROM, I/O, and Timer/Counters. To help make the approach more cost effective, in 1977 Intel introduced the UPI-41 family of Universal Peripheral Interface controllers consisting of an 8041AH (ROM) device and an 8741AH (EPROM) device. These devices integrated the common microprocessor system functions into one 40 pin package. The UPI-42AH family, consisting of the 8042 and 8742AH, further extends the UPI's cost effectiveness through more memory and higher speed.

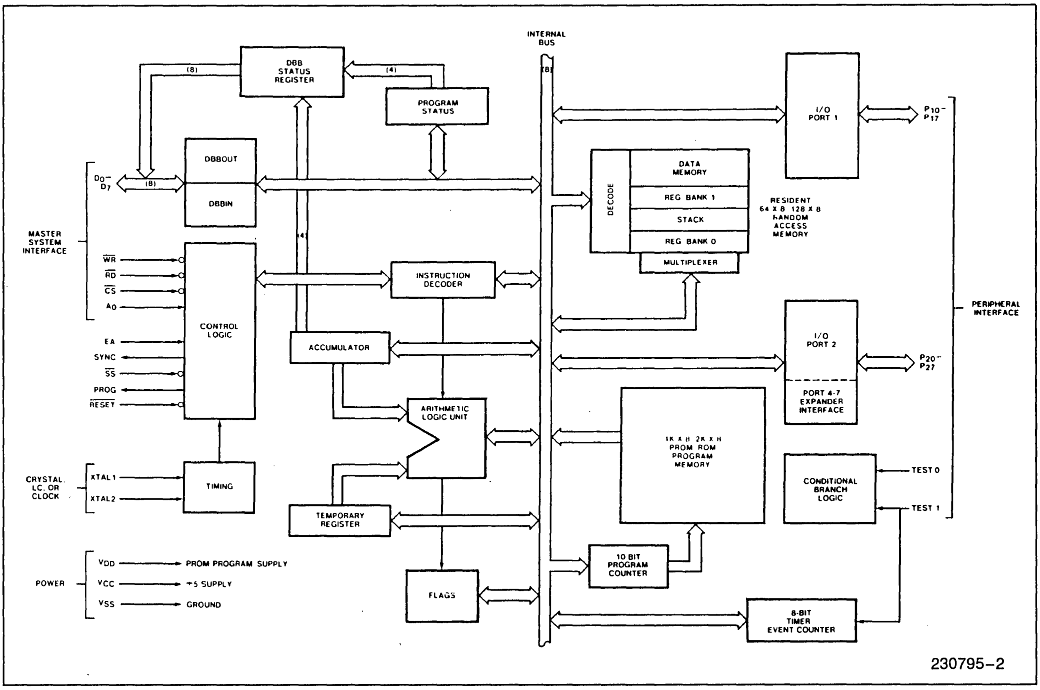
Another member of the UPI family is the Intel 8243 Input/Output Expander chip. This chip provides the UPI-41AH and UPI-42AH with up to 16 additional independently programmable I/O lines, and interfaces directly to the UPI-41AH/42AH. Up to seven 8243s can be cascaded to provide over 100 I/O lines.

The UPI is a single chip microcomputer with a standard microprocessor interface. The UPI's architecture, illustrated in Figure 3, features on-chip program memory, ROM (8041A/8042) or EPROM (8741A/8742AH), data memory (RAM), CPU, timer/counter, and I/O. Special interface registers are provided which enable the UPI to function as a peripheral to an 8-bit central processor.

Using one of the UPI devices, the designer simply codes his proprietary peripheral control algorithm into the UPI device itself, rather than into the main system software. The UPI device then performs the peripheral control task while the host processor simply issues commands and transfers data. With the proliferation of microcomputer systems, the use of UPIs or slave microprocessors to off load the main system microprocessor has become quite common.

This Application Note describes how the UPI-42AH can be used to control dot matrix printing and the printer mechanism, using stepper motors for carriage/print head assembly and paper feed motion. Previous Intel Application Notes AP-27, AP-54, and AP-91 describe using intelligent processors and peripherals to control single solenoid driven printer mechanisms with 80 character line buffering and bidirectional printing. This Application Note expands on these previous themes and extends the concept of complex device control by incorporating full 80 character line buffering, bidirectional printing, as well as drive and feedback control of two four phase stepper motors.

The Application Note assumes that the reader is familiar with the 8042/8742AH and 8243 Data Sheets, and UPI-41AH/42AH Assembly Language. Although some background information is included, it also assumes a basic understanding of stepper motors and dot matrix printer mechanisms. A complete software listing is included in Appendix A.



230795-2

Figure 2. UPI-42AH Block Diagram

DOT MATRIX PRINTING

A dot matrix printer print head typically consists of seven to nine solenoids, each of which drives a stiff wire, or hammer, to impact the paper through an inked ribbon. Characters are formed by firing the solenoids to form a matrix of "dots" (impacts of the wires). Figure 4 shows how the character "E" is formed using a 5 × 7 matrix. The columns are labeled C1 through C5, and the rows R1 through R7. The print head moves left-to-right across the paper, so that at time T1 the head is over column C1. The character is formed by activating the proper solenoid as the print head sweeps across the character position.

Dot matrix printers are a cost effective way of providing good quality hard copy output for microcomputer systems. There is an ever increasing demand for the moderately priced printer to provide more functionality with improved cost and performance. Using stepper motors to control the paper feed and carriage/print head assembly motion is one way of enabling the dot matrix printer to provide more capabilities, such as expanded or contracted characters, dot or line graphics, variable line and character spacing, and subscript or superscript printing.

However, stepper motors require fairly complex control algorithms. Previous solutions involved the use of a main CPU, UPI, RAM, ROM, and I/O onboard the peripheral. The CPU acted as supervisor and used parallel processing to achieve accurate stepper motor control via a UPI, character buffering via the I/O device, RAM, and ROM. The CPU performed real-time decoding of each character into a dot matrix pattern. This Application Note demonstrates that the increased memory and performance of the UPI-42AH facilitates integrating these control functions to reduce the cost and component count.

THE PRINTER MECHANISM

The printer mechanism used in this application is the Epson Model 3210. It consists of four basic sub-assemblies; the chassis or frame, the paper feed mechanism and stepper motor, the carriage motion mechanism and stepper motor, and the print head assembly.

The paper feed mechanism is a tractor feed type. It accommodates up to 8.5 inch wide paper (not including tractor feed portion). There is no platen as such; the paper is moved through the paper guide by two sprocketed wheels mounted on a center sprocket shaft. The sprocket shaft is driven by a four phase stepper motor. The rotation of the stepper motor is transmitted to the sprocket shaft through a series of four reduction gears.

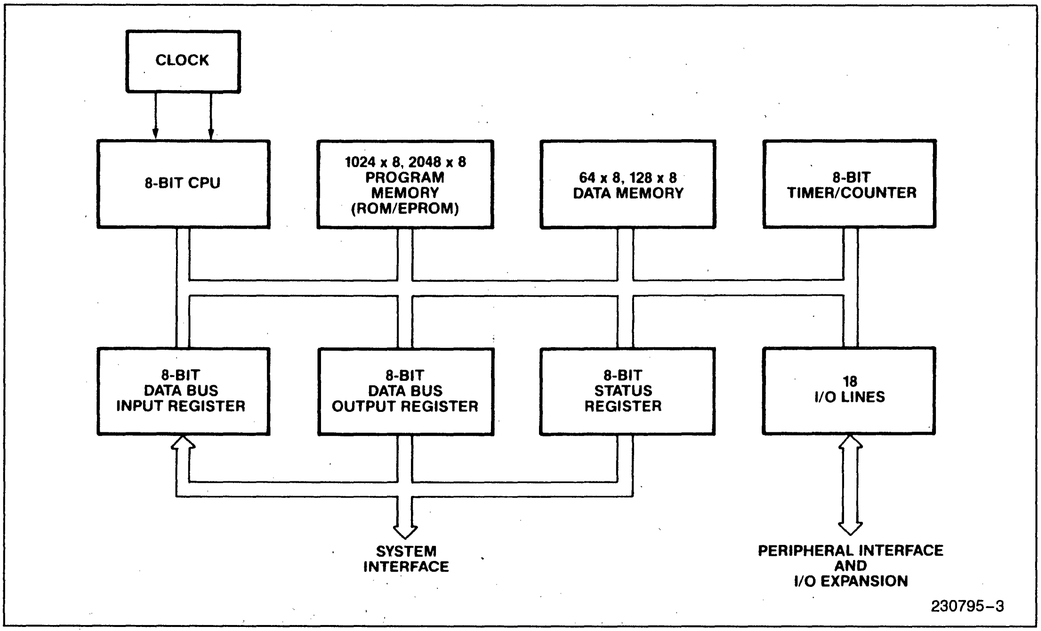

CLOCK
8-BIT CPU
1024 x 8, 2048 x 8 PROGRAM MEMORY (ROM/EPROM)
64 x 8, 128 x 8 DATA MEMORY
8-BIT TIMER/COUNTER
8-BIT DATA BUS INPUT REGISTER
8-BIT DATA BUS OUTPUT REGISTER
8-BIT STATUS REGISTER
18 I/O LINES
SYSTEM INTERFACE
PERIPHERAL INTERFACE AND I/O EXPANSION
230795-3

Figure 3. UPI-41AH, 42AH Functional Block Diagram

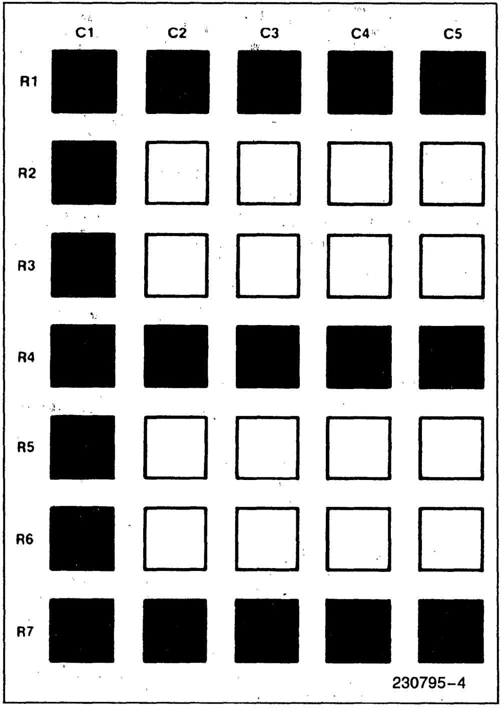


Figure 4. Character E in 5 x 7 Dot Matrix Format

The carriage motion mechanism consists of another four phase stepper motor which controls the left-to-right or right-to-left print head assembly motion. The print speed is 80 CPS maximum. Both the speed of the stepper motor and the movement of the print head assembly are independently controllable in either direction. The rotation of the stepper motor is converted to the linear motion of the print head assembly via a series of reduction gears and a toothed drive belt. The drive belt also controls a second set of reduction gears which advances the print ribbon as the print head assembly moves.

Two optical sensors provide feedback information on the carriage assembly position and speed. The first of these optical sensors, called the 'HOME RESET' or HR, is mounted near the left-most physical position to which the print head assembly can move. As the print head assembly approaches the left-most position, a flange on the print head assembly interferes with the light source and sensor, causing the output of the sensor to shift from a logic level one to zero. The right-most printer position is monitored in software rather than by another optical sensor. The right-most print position is a function of the number of characters printed and the distance required to print them.

The second optical sensor, called the 'PRINT TIMING SIGNAL' or PTS, provides feedback on carriage stepper motor velocity and relative position within a

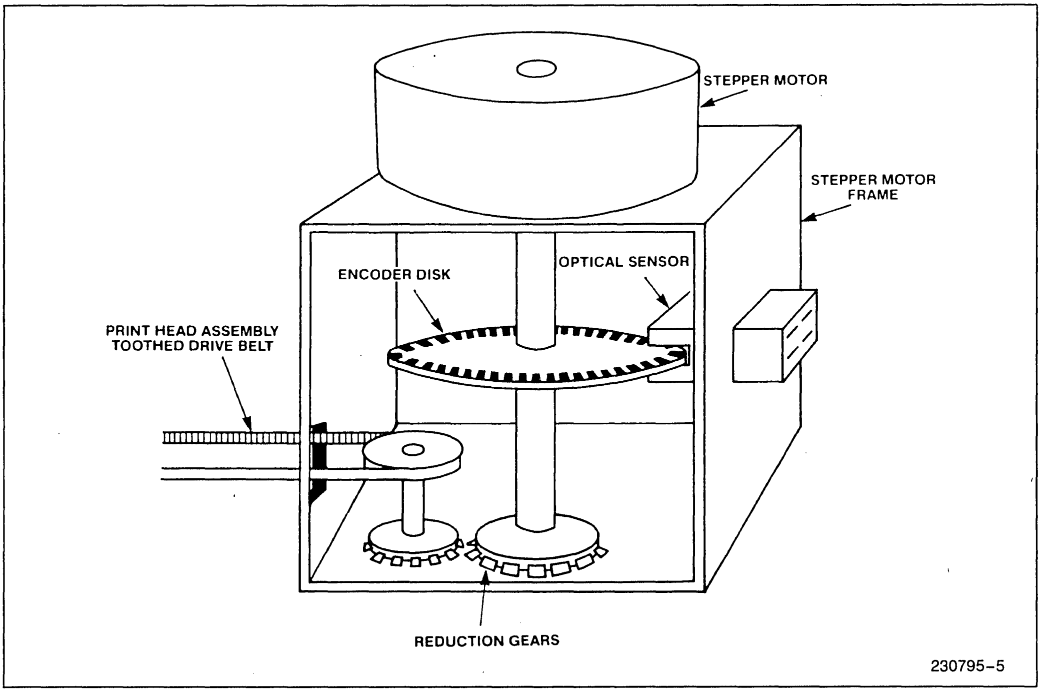


Figure 5. Carriage Stepper Motor Assembly

given step of the motor. The feedback is generated by the optical sensor as an "encoder disk" moves across it. Figure 5 illustrates the carriage stepper motor, optical sensor, encoder disk and reduction gears, and drive belt assembly. The optical sensor outputs a pulse train with the same period as the phase shift signal used to drive the stepper, but slightly out of phase with it when the motor is at a constant speed (see Software Functional Block: Phase Shift Data for additional details). The disk acts as a timing wheel, providing feedback to the UPI software of the carriage speed, position, and optimum position for energizing the print head solenoids. The two optical sensors are monitored under software and provide the critical feedback needed to control the print head assembly and paper feed motion accurately. The process of stepper motor drive and control via feedback signals is called "closed loop" stepper motor control, and is covered in more detail in the software discussion.

The print head assembly consists of nine solenoids and nine wires or hammers. Figure 6 illustrates a print head assembly. The available dot matrix measures 9 x 9. This large matrix enables the Epson 3210 print mechanism to print a variety of character fonts, such as expanded or contracted characters, as well as line or block graphics (see Appendix B, Printer Enhancements). It also facilitates printing lower case ASCII characters with "lower case descenders." That is to say, certain lower case letters (e.g., y, p, etc.) will print below the bottom part of all upper case letters.

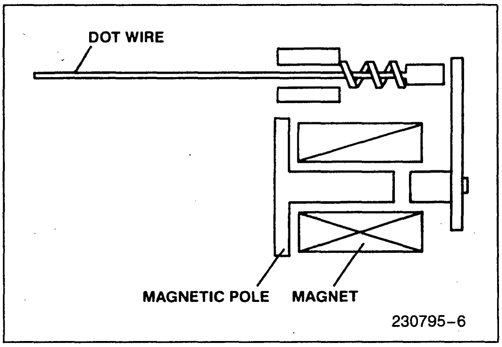


Figure 6. Print Head Solenoid Assembly

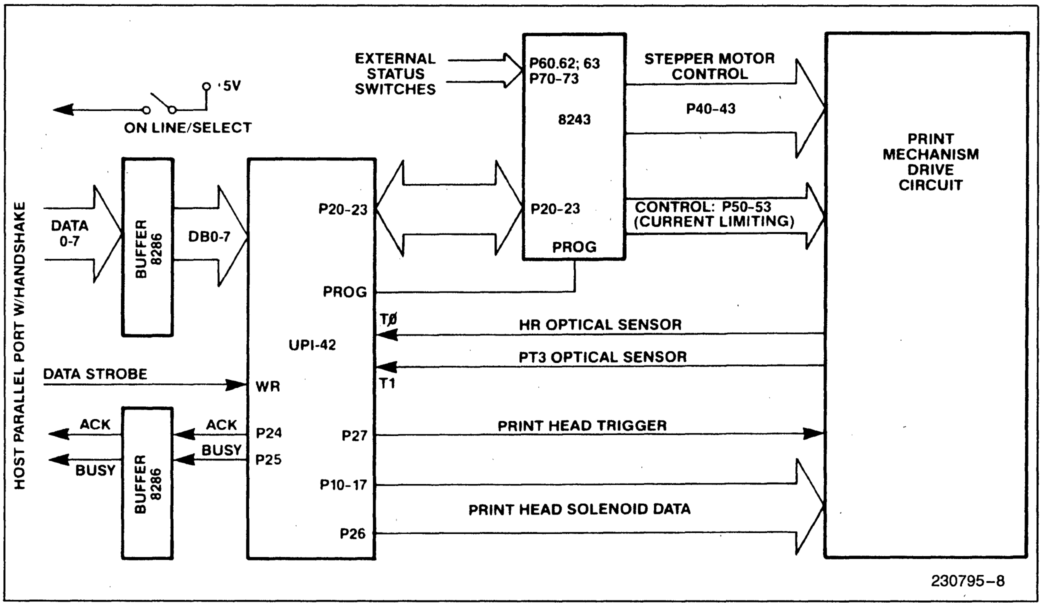


Figure 7. Hardware Interface Block Diagram

HARDWARE DESCRIPTION

Figure 7 shows a block diagram of the UPI-42AH and 8243 interface to the printer mechanism drive circuit. A complete schematic is shown in Figure 8. The UPI-42AH provides all signals necessary to control character buffering and handshaking, paperfeed and carriage motion stepper motor timing, print head solenoid activation, and monitoring of external status switches.

The Epson 3210 printer mechanism manual recommends a specific interface circuit to provide proper drive levels to the stepper motors windings and print head solenoids. The hardware interface used for this Application Note followed those recommendations exactly (see Appendix C, Printer Mechanism Drive Circuit Schematics).

I/O Ports

The lower half of the UPI-42AH Port 2, pins 0–3, provides an interface to the 82431 I/O expander. The PROG pin of the UPI-42AH is used as a strobe to clock address and data information via the Port 2 interface. The extra 16 I/O lines of the 8243 become PORTS 4, 5, 6, and 7 to the UPI software. Combined, the UPI-42AH and 8243 provide a total of 28 independently programmable I/O line. These lines are used as shown in Figure 9.

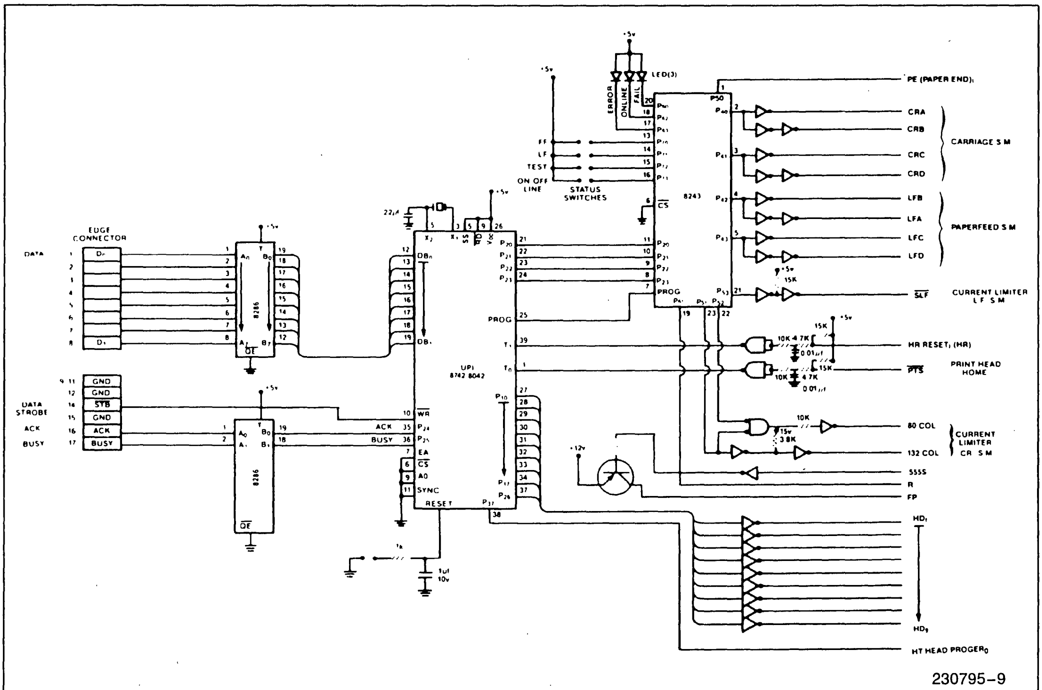


Figure 8. Hardware Interface Schematic

Port	No. of lines	Bits	I/O	Description
1	8	0–7	O	Character dot column data to print head solenoids
2	1	6	O	(same)
2	1	7	O	Print head solenoid trigger
2	2	4, 5	O	Host system data transfer handshaking (ACK/BUSY)
4	4	0–3	O	Carriage & paper feed stepper motors
5	3	1–3	O	Stepper motor select and current limiting
5	1	0	I	Paper End sense
6	1	1	O	Print head trigger reset
6	3	0, 2, 3	-	(unused)
7	5	0–3	I	External status switches; (LF, FF, TEST, ON/OFF Line)

Figure 9. UPI-42AH and 8243 I/O Port Map

NOTE:
The notation used in the balance of this Application Note, when referring to a port number and a particular pin or bit, is Port 23 rather than Port 2 bit 3.

The two printer mechanism optical sensors, discussed in the Printer Mechanism description are tied to the two "Test Input" pins, T0 and T1, of the UPI-42AH through a buffer circuit for noise suppression. These inputs are directly testable in software.

Host System Interface

The host system interfaces to the printer through a parallel port to the UPI-42AH Data Bus. Four handshaking signals are used to control data transfer; Data Strobe (STB/), Acknowledge (ACK), Busy (BUSY), and Online or Select. The Data Strobe line of the host parallel port is tied directly to the UPI-42AH WR/ pin. This provides a low going pulse on the UPI-42AH WR/ pin whenever a data byte is written to the UPI-42AH. The ACK and BUSY handshake signals are tied to two UPI-42AH I/O port lines for software control of data transfer. The "On Line" handshake signal is tied to a single-pole single-throw fixed position switch, which externally enables or disables character transfer from the host system. Characters transmitted to the UPI-42AH by the host are loaded into the UPI-42AH Data Bus Buffer In (DBBIN) register, and the Input Buffer Full (IBF) interrupt and UPI-42AH status flag are set (see Figure 9. UPI-42AH and 8243 I/O Ports).

Stepper Motor Interface

Port 4 (41-43) of the 8243, provides both carriage and paper feed stepper motor phase shift signals to the printer mechanism drive circuit. Each of the two stepper motors is driven by 2 two phase excitation signals (4 phases). Figure 10 shows the wave form for each stepper motor. Each signal consists of two components (Sig. 1 A/B & Sig. 2 C/D) 180 degrees out of phase with the other. Each of these signal pairs (A/B & C/D) is 90 degrees out of phase with the other pair. For each signal pair, one port line supplies both halves by using an inverter.

Each of the resulting eight stepper motor drive signals is interfaced to a discrete drive transistor through an inverter. The emitter of the drive transistor is tied to the open collector of the inverter to provide high current sinking capability for the drive transistor. Each half of the motor winding is tied to the collector of the drive transistor (see Appendix C, Printer Mechanism Drive Circuit Schematic).

Each stepper motor requires two current levels for operation. These levels are called "Rush" current and "Hold" current. Rush current refers to the high current required to cause the rotor to rotate within its windings as the polarity of the power applied to the windings is changing. Each change in the polarity of the power applied to the motor windings is called a step of phase shift. Hold current refers to the low level of current required to stabilize and maintain the rotor in a fixed position when the polarity is applied to the windings is not changing. Hold current is simply Rush current with a current limiting transistor switched in. Switching from Hold to Rush current "selects" or enables that stepper motor to move with the next step signal output. In the balance of this Application Note, the term "select" will be used to refer to turning on Rush current, and "deselect" will refer to switching to Hold current.

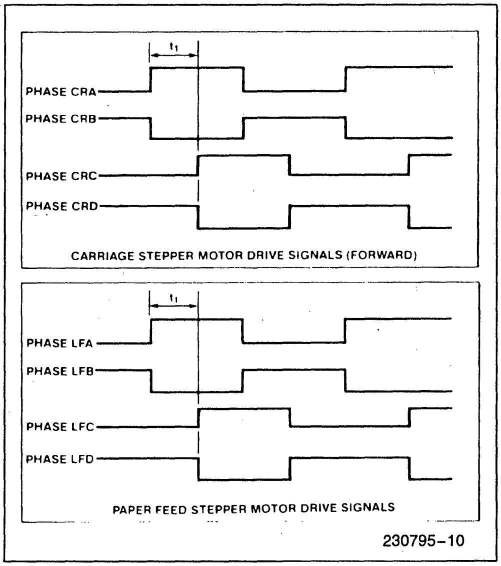


Figure 10. Stepper Motor Step Sequence Waveforms

Three 8243 port lines are dedicated to the select/deselect control of the two stepper motors. One line is for the paper feed stepper motor, and two lines are for the carriage motion stepper motor (80 and 132 column). These lines are labeled SLF, 80Col, and 132Col, and are 8243 PORT 53, 52, and 51, respectively.

By varying the voltage applied to the stepper motor biasing circuit and the current, it is possible to vary the distance the motor moves the print head assembly with each step. Enabling one of two different voltage biasing levels, and changing the timing rate at which the motor is stepped, facilitates either 80 or 132 character column printing. Only 80 character column printing is implemented in the software design. Appendix B, Printer Enhancements, details the software algorithm for handling 132 character printing.

Print Head Interface

A total of eleven I/O lines are used to control the print head solenoids and solenoid firing (see Figure 9). Nine are used for character dot data, one for the Print Head Trigger, and one for Reset of the Print Head Trigger circuit. Each of the nine character dot data lines is buffered by an open collector hex inverter.

The Print Head Trigger output is tied to the Trigger input of a 555 Monostable Multivibrator. The output pulse generated by the 555 triggers the print head solenoids to fire. The 555 Output pulse width is independent of the input trigger waveform. The pulse width is determined by an RC network across the 555 inputs and the voltage level applied to the Control Voltage 555 input. The 555 Output is tied to the base of a PNP transistor through an inverter, biased in a normally off configuration. The PNP transistor supplies enough drive to pull up the open collector inverter on each print head solenoid line, Port 10-17 and 26. The 555 output pulse momentarily enables the print head solenoid line open collector inverter output, turning on the solenoid drive transistor, and firing the print head hammer. The 555 Output pulse width is approximately 400 us. Further details of the print head firing operation can be found in the software description below.

Miscellaneous Interface Signals

The 8243 Port 5 pin 0 is tied to the Paper End Detector, a reed switch located on the printer paper guide. This sensor detects when the paper is nearly exhausted.

Three LED status lights complete the hardware interface design. One status light is used for each of the following: Power ON/OFF, On/Off Line, and Out of Paper.

BACKGROUND

Before a detailed discussion of the software begins, a few terms and software functions reference throughout the software need introduction.

A. What is a Stepper Motor?

A stepper motor has the ability to rotate in either direction as well as start and stop at predetermined angular positions. The stepper motor's shaft (rotor) moves in precise angular increments for each input step. The displacement is repeated for each input step command, accurately positioning the rotor for a given number and sequence of steps.

The stepper motor controls position, velocity, and direction. The accuracy of stepper motors is generally 5 percent of one step. The number of steps in each revolution of the shaft varies, depending on the intended application.

B. Open/Closed Loop Stepper Motor Drive and Control

The carriage stepper motor is closed loop driven. The paper feed stepper motor is open loop driven.

There are two major types of stepper motor control known by the broad headings of open and closed loop. Open loop is simply continuous pulses to drive the motor at a predetermined rate based on the voltage, current, and the timing of the step pulses applied. Closed loop control is characterized by continuous monitoring of the stepper motor, through feedback signals, and adjusting the motor's operation based upon the feedback received.

C. Stepper Motor Drive Phase Shift of Step Sequence

Each change in the polarity of the power applied to the motor windings is called a step or phase shift. The sequence of the steps or phase shifts, and the pattern of polarity changes output to the stepper motor, determines the direction of rotation.

Figure 10 shows the waveforms for each of the two stepper motors. Figure 11 lists the step sequence for carriage motor clockwise rotation, whch moves the print head assembly Left-to-Right. Figure 11 also lists the step sequence for counterclockwise rotations; the print head assembly moves Right-to-Left. Figure 12 lists the step sequence for the paper feed stepper motor clockwise drive. The phase sequence, for either stepper motor, may begin at any point within the sequence list, but must then continue in order.

Step No.	A-Step	B-Step	C-Step	D-Step
1	On	Off	Off	On
2	On	Off	On	Off
3	Off	On	On	Off
4	Off	On	Off	On

Carriage stepper motor rotates clockwise Print head assembly moves from left to right

Step No.	A-Step	B-Step	C-Step	D-Step
1	On	Off	On	Off
2	On	Off	Off	On
3	Off	On	Off	On
4	Off	On	On	Off

Carriage stepper motor rotates counter clockwise Print head assembly moves from right to left

Figure 11. Carriage Stepper Motor Step Sequence

Step No.	A-Step	B-Step	C-Step	D-Step
1	On	Off	On	Off
2	On	Off	Off	On
3	Off	On	Off	On
4	Off	On	On	Off

Figure 12. Paper Feed Stepper Motor Step Sequence

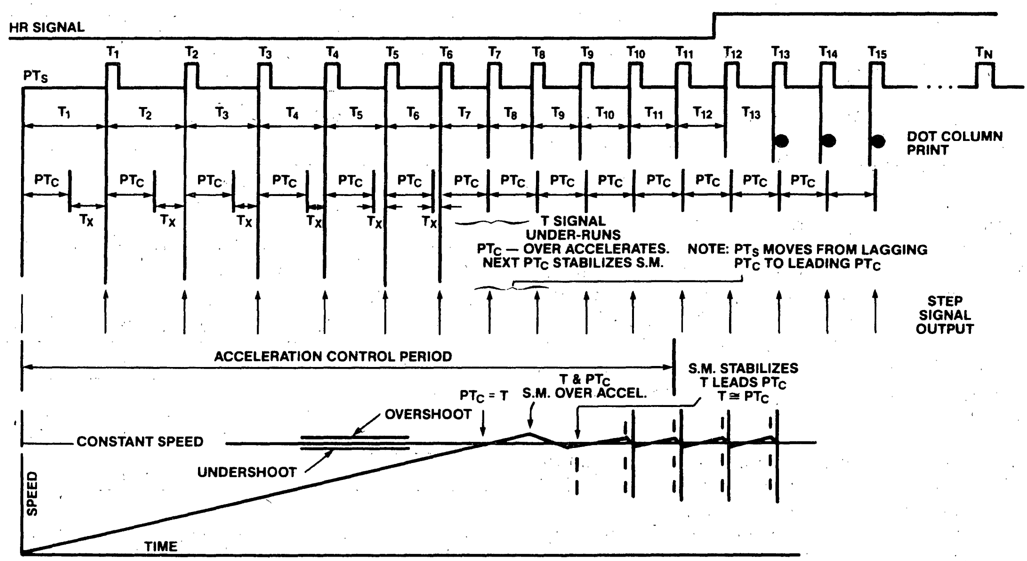
D. Acceleration and Deceleration of Stepper Motors

The carriage stepper motor starts from a fixed position, accelerates to a constant speed, maintains constant speed, and then decelerates to a fixed position. Printing may occur from the time and position the print head assembly reaches constant speed, until the time and position the print head assembly begins to decelerate from constant speed. Whether printing occurs during any carriage stepper motor drive sequence is controlled by software. Figure 18, below, illustrates these components of print head assembly line motion.

Due to inertia, a finite time interval and angular displacement is required to accelerate a stepper motor to its full speed. Conversely, deceleration must begin some time before the final angular position. The time interval and angular displacement of the carriage stepper motor translates into the distance the print head assembly travels before it reaches a constant speed. The distance traveled during acceleration is constant. The distance the print head assembly travels during deceleration must be the same as the distance traveled during acceleration in order to accurately align the character dot columns from one line to the next.

E. Stepper Motor Predetermined Time Constant

Whenever the stepper motor is stepped, or energized, the angular velocity of the rotor is greater than the constant speed which is ultimately required. This is called "overshoot." The frictional load of the carriage assembly (motor rotor, reduction gears, drive belt and print head assembly, or paper feed sprocket shaft and wheels) provides damping or frictional load. Damping slows the motor to less than the required constant speed and is called "undershoot" (see Figure 13, Carriage Stepper Motor Drive Timing). A constant rate of speed is achieved through the averaging of the overshoot and undershoot within each step.


HR SIGNAL
T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8 T_9 T_{10} T_{11} T_{12} T_{13} T_{14} T_{15} T_N
PT_S
T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8 T_9 T_{10} T_{11} T_{12} T_{13}
DOT COLUMN PRINT
PT_C
T_X
T SIGNAL UNDER-RUNS
PT_C — OVER ACCELERATES. NEXT PT_C STABILIZES S.M.
NOTE: PT_S MOVES FROM LAGGING PT_C TO LEADING PT_C
STEP SIGNAL OUTPUT
ACCELERATION CONTROL PERIOD
S.M. STABILIZES T LEADS PT_C $T \cong PT_C$
T & PT_C
PT_C = T
S.M. OVER ACCEL.
OVERSHOOT
CONSTANT SPEED
UNDERSHOOT
SPEED
TIME

EQUATIONS:

PT_C = PREDETERMINED TIME CONSTANT
$PT_S = T_1 - T_N$

$T_1 \ldots T_6$ TIME = $PT_C + T_X$
T_7 TIME = PT_C
$T_8 \ldots T_{11}$ Time = PT_C

230795-11

Figure 13. Carriage Stepper Motor Drive Timing

The Predetermined Time (PT) Constant is the time required to average the overshoot and undershoot of the particular stepper motor for a desired constant rate of speed. The PT also is the time required to move the print head assembly a specific distance, accounting for both overshoot and undershoot of the stepper motor.

Changing the Predetermined Time Constant changes the angular displacement of the stepper motor rotor, this in turn changes the output. Figure 14 lists the Time Constants for both standard and condensed character printing. Figure 15 lists the paper feed stepper motor Time Constants used for various line spacing formats. This Application Note implements standard character print and paper feed (6 lines per inch) Time Constants. See Appendix B, Printer Enhancement, for details on implementing non-standard Time Constants.

Character mode	Predetermined time	
Standard or Enlarged Character	2.08 ms	+10% −4%
Condensed Character	4.16 ms	+10% −4%

Figure 14. Carriage Stepper Motor Predetermined Time Constants

Paper feed pitch	0.12 mm (1/216″)/1 pulse
	4.23 mm (1/16″)/36 pulses
	3.18 mm (1/8″)/27 pulses
	2.82 mm (1/9″)/24 pulses
Paper feed time	
150 ms/4.23 mm	Approx. 6.6 lines/s (continuous feed)
113 ms/3.18 mm	Approx. 8.8 lines/s (continuous feed)
100 ms/2.82 mm	Approx. 10 lines/s (continuous feed)

Figure 15. Paper Feed Stepper Motor Predetermined Time Constants

F. Relationship Between PTS and PT

Figure 13 illustrates how PTS lags PT at the start of acceleration, and moves to lead PT as the motor achieves constant speed. Figure 13 also illustrates the relationship between HR, PTS, PT, acceleration, constant speed, and printing. Figures 16 and 17 illustrate the relationship between PTS and PT during acceleration and at constant speed.

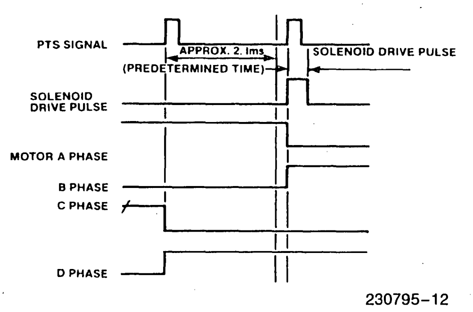


Figure 16. PTS Lags PT Timing

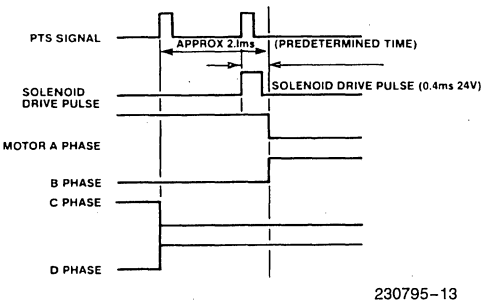


Figure 17. PTS Leads PT Timing

PTS is the point of peek angular velocity within a step of the motor. PTS is a function of the slot spacing on the encoder disk, shown in Figure 5. The spacing is determined by the mechanics of the printer mechanism.

When the carriage stepper motor is accelerated from a fixed position, the effects of damping slows the angular velocity of energizing the stepper motor. This causes PTS to occur after the PT, or PTS lags PT. When PTS lags PT, the next step signal is output at PTS rather than at PT. If the step signal is outputted at PTS, the rotor could be midway through a rotation. Energizing the motor at PT could cause it to bind or shift in the wrong direction. When the carriage stepper motor is at a constant rate of speed, PTS leads PT and the step signal is output at PT (see Figure 13).

G. Stored Time Constants

The time between each step, for a constant number of steps, required for the motor to reach a constant speed, is calculated and stored in Data Memory during acceleration. The values stored are used, in reverse order, during deceleration as the Predetermined Time (PT) Constants. This ensures that the acceleration and deceleration distance traveled by the print head assembly is the same, and that it accurately aligns character dot columns from one line to the next during printing. The time values stored are called "Stored Time Constants." Steps T1 through T11 in Figure 13, represent the Stored Time Constants.

The equations for the Stored Time Constants are given at the bottom of Figure 13, Carriage Stepper Motor Drive Timing.

H. Print Head Assembly "Home" Position

The "logical" Home position for the print head assembly is the left-most position at which printing begins (for L-to-R motion) or ends (for R-to-L motion). The "physical" Home position is the logical HOME position, plus the distance required by the carriage stepper motor to fully accelerate the print head assembly to a constant speed. Printing can only occur when the print head is moving at a constant speed. The printer mechanism manual stipulates eleven step time periods are required to ensure the print head assembly is at a constant speed. These eleven step time periods are the Stored Time Constants described above. Figure 18 illustrates the components of print head assembly line motion and character printing.

SOFTWARE

Introduction

The software description is presented in three sections. First, a brief overview of the software to familiarize the reader with the interdependencies and overall program flow. Second, data and program memory allocation and status register flag definitions. And third, each of the ten software blocks is presented with its own flowchart.

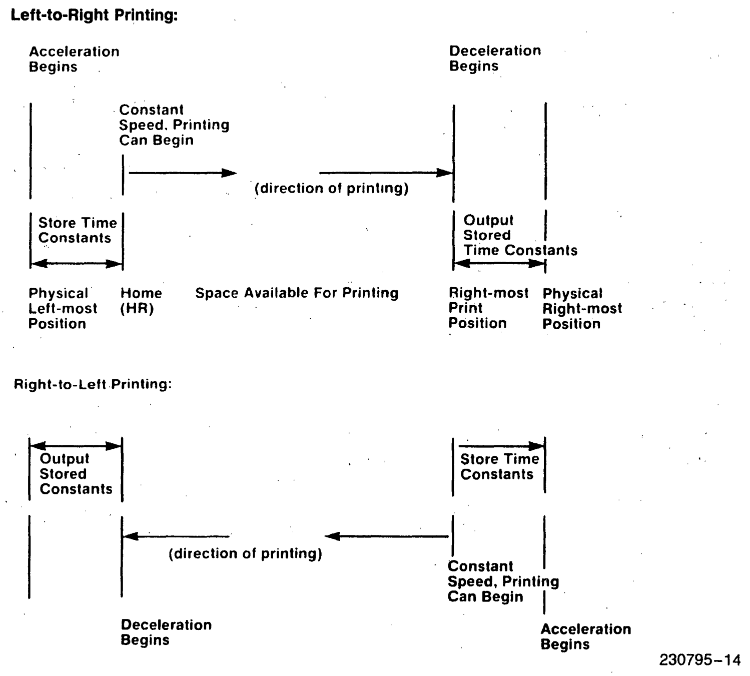


Figure 18. Components of Print Head Assembly Line Motion and Printing

Software Overview

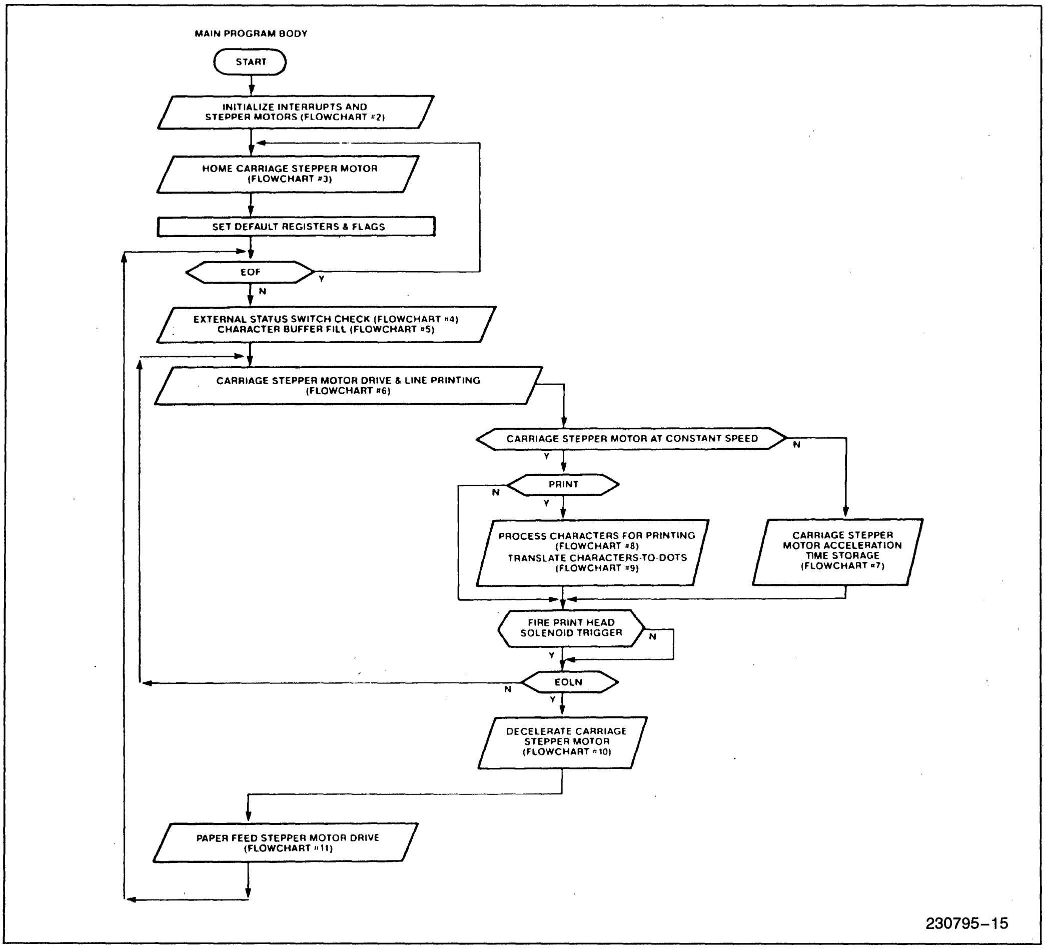
The software is written in Intel UPI-41AH/42AH Assembly Language. A block structure approach is used for ease of development, maintenance, and comprehension. The software is divided into five principal parts.

1. Initialization
2. Character Buffering or Input
3. Stepper Motor Drive and Control
4. Character Processing
5. Character Printing or Output

The five principal parts are incorporated into ten software blocks, listed below.

1. Power On/Reset Initialization
2. Home Print Head Assembly
3. External Status Switch Check
4. Character Buffer Fill
5. Carriage Stepper Motor Drive and Line Printing
6. Accelerate Stepper Motor Time Storage
7. Process Characters for Printing
8. Translate Character-to-Dots
9. Decelerate Carriage Stepper Motor
10. Paperfeed Stepper Motor Drive

Flow Chart No. 1 illustrates the overall software algorithm. Below, is a description of the algorithm.


MAIN PROGRAM BODY
START
INITIALIZE INTERRUPTS AND STEPPER MOTORS (FLOWCHART #2)
HOME CARRIAGE STEPPER MOTOR (FLOWCHART #3)
SET DEFAULT REGISTERS & FLAGS
EOF
Y
N
EXTERNAL STATUS SWITCH CHECK (FLOWCHART #4) CHARACTER BUFFER FILL (FLOWCHART #5)
CARRIAGE STEPPER MOTOR DRIVE & LINE PRINTING (FLOWCHART #6)
CARRIAGE STEPPER MOTOR AT CONSTANT SPEED
N
Y
PRINT
N
Y
PROCESS CHARACTERS FOR PRINTING (FLOWCHART #8) TRANSLATE CHARACTERS-TO-DOTS (FLOWCHART #9)
CARRIAGE STEPPER MOTOR ACCELERATION TIME STORAGE (FLOWCHART #7)
FIRE PRINT HEAD SOLENOID TRIGGER
N
Y
EOLN
N
Y
DECELERATE CARRIAGE STEPPER MOTOR (FLOWCHART #10)
PAPER FEED STEPPER MOTOR DRIVE (FLOWCHART #11)
230795-15

Flow Chart No. 1. Main Program Body

Upon power-on reset, a software and hardware initialization is performed. This stabilizes and sets inactive the printer hardware and electronics. The print head assembly is then moved to establish its HOME position. The default status registers are set for character buffering, carriage, and paper feed stepper motor drive. The External Status switches are checked; FORMFEED, LINEFEED, ON/OFF LINE, and Character Print TEST. If the printer is ON LINE, the software will loop on filling the Data Memory Character Buffer.

Character or data input to the UPI-42AH is interrupt driven. Characters sent by the host system set the Input Buffer Full (IBF) interrupt and the IBF Program Status flag. Character input servicing (completed during the paper feed and carriage stepper motor drive end Delay subroutine) tests for various ASCII character codes, loads characters into the Character Buffer (CB), and repeats until one of several conditions sets the CB Full status flag. Once the CB Full flag is set, further character transmission by the host system is inhibited and printing can begin.

The carriage stepper motor is initialized, and drive begins for the direction indicated. The motor is accelerated to constant speed, printable character codes are translated to dot patterns and printed (if printing is enabled), and the motor is decelerated to a stop. Two timing loops guarantee both constant speed and protection (Failsafe Time) against stepper motor burn out due to high current overload. The two optical sensors, described in the Printer Mechanism section above, are constantly monitored to maintain constant speed, and trigger print head solenoid firing.

Once the line is printed and the carriage stepper motor drive routine has been completed, a Linefeed is forced. The paper feed stepper motor drive subroutine tests the number of lines to move, and energizes the paper feed stepper motor for the required distance. The lines per page default is 66; if 66 lines have been received, a Formfeed to Top-of-Next-Page is performed. The Top-Of-Page is set at Power On/Reset.

When the EOF code is received, the EOF status flag is set. When the last line has been printed, the EOF check will force the print head assembly to the HOME position. The EOF flag is tested following each paperfeed stepper motor drive. The next entry to the External Status Check subroutine begins a loop which waits for input from either the external status switches or the host system.

The software character dot matrix used in this application is 5 x 7 of the available 9 x 9 print head solenoid matrix. Although lower case descenders and block/line graphics characters are not implemented, Appendix B, Printer Enhancements, discusses how and where these enhancements could be added. The software implements the full 95 ASCII printable characters set.

Memory and Register Allocation

DATA MEMORY ALLOCATION (RAM)

The UPI-42AH has 256 bytes of Data Memory. Sixteen bytes are used by the two 8 byte register banks (RB0 and RB1). Sixteen additional bytes are used for the Program Stack. The Stored Time Constants utilize 11 bytes, while the stepper motor phase storage requires 4 bytes. Below is a detailed description of Data and Program Memory.

Hex Address	Description
2F-7FH	80 Character Line Buffer (80 Bytes)
25-2EH	Stored Time Constants Buffer (11 Bytes)
24H	Unused
23H	Character Print Test ASCII Code Start Temporary Storage
22H	Pseudo Register: Paperfeed Stepper Motor Last Phase Indirect Address
21H	Pseudo Register: Carriage Stepper Motor Forward/Reverse Last Phase
20H	Pseudo Register: Last Phase of Stepper Motor Not Being Driven
18-1FH	Register Bank 1: Character Processing
8-17H	8 Level Stack
0-07H	Register Bank 0: Stepper Motor Forward/Reverse Acceleration/Drive

Figure 19. Data Memory Allocation Map

Register Bank 0 is used for stepper motor drive functions. Register Bank 1 is used for character processing. Each register bank's register assignments is listed in Figures 20 and 22, respectively. Each register bank has one register allocated as a Status Register. Figures 21 and 23 detail the Status Register flag assignments. Note that bit 7 of each Status Byte is used as a print head assembly motion direction flag. This saves coding of the Select Register Bank (SEL RBn) instruction at each point the flag is checked.

Register Bank 0

Register	Program Label	Description
R0	TmpR00	RB0 Temporary Register
R1	TStrR0	Store Time Register
R2	GStR20	General Status Register
R3	PhzR30	Stepper Motor Step Register
R4	CntR40	Count Register
R5	TConR0	Time Constant Register
R6	LnCtR0	Line Count Register
R7	OpnR70	Available, Scratch

Figure 20. Register Bank 0 Register Assignment

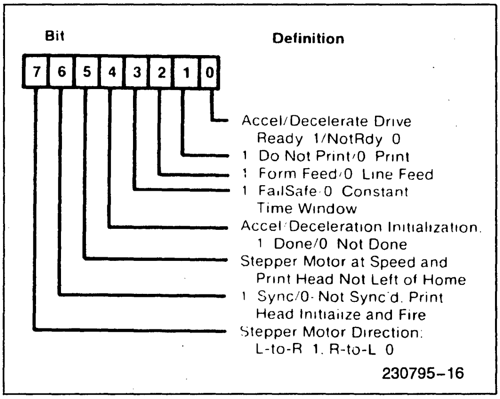

Bit
Definition
7 6 5 4 3 2 1 0
Accel/Decelerate Drive Ready 1/NotRdy 0
1 Do Not Print/0 Print
1 Form Feed/0 Line Feed
1 FailSafe/0 Constant Time Window
Accel/Deceleration Initialization. 1 Done/0 Not Done
Stepper Motor at Speed and Print Head Not Left of Home
1 Sync/0 Not Sync'd. Print Head Initialize and Fire
Stepper Motor Direction: L-to-R 1, R-to-L 0
230795-16

Figure 21. Register Bank 0 Status Byte Flag Assignments

Register	Program Label	Description
R0	TmpR10	RB0 Temporary Register
R1	CAdrR1	Character Data Memory Address Register
R2	ChStR1	Character Processing Status Byte Register
R3	CDtCR1	Character Dot Count Register
R4	CDotR1	Character Dot Temporary Storage Register
R5	CCntR1	Character Count Temporary Register
R6	StrCR1	Store Character Register
R7	OpnR71	Available/Scratch

Figure 22. Register Bank 1 Register Assignment

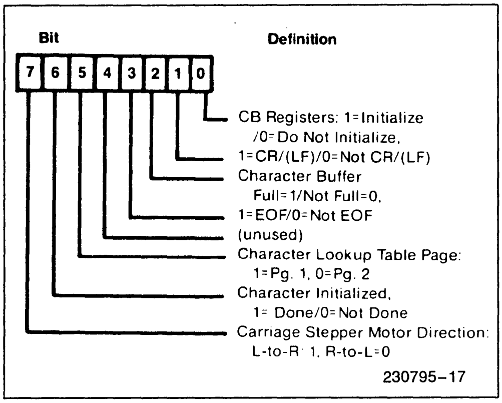

Bit
Definition
7 6 5 4 3 2 1 0
CB Registers: 1=Initialize /0=Do Not Initialize.
1=CR/(LF)/0=Not CR/(LF)
Character Buffer Full=1/Not Full=0.
1=EOF/0=Not EOF
(unused)
Character Lookup Table Page: 1=Pg. 1, 0=Pg. 2
Character Initialized, 1= Done/0=Not Done
Carriage Stepper Motor Direction: L-to-R 1, R-to-L=0
230795-17

Figure 23. Register Bank 1 Status Byte Flag Assignments

PROGRAM MEMORY ALLOCATION (EPROM/ROM)

The UPI-42AH has 2048 bytes of Program Memory divided into eight pages, each 256 bytes. Figure 24 illustrates the Program Memory allocation map by page.

Page	Hex Address	Description
Page 7	1792–2047	Character to Dot Pattern Lookup Table; Page 2: ASCII 50H-7EH
Page 6	1536–1791	Character to Dot Pattern Lookup Table; Page 1: ASCII 20H-4FH (sp-M)
Page 5	1280–1535	Miscellaneous Subroutines: InitAl/AllOff Clear Data Memory Home Print Head Assembly Character Print Test Initialize Carriage Stepper Motor Delay Stepper Motor Deselect
Page 4	1024–1279	Paper Feed Stepper Motor Drive
Page 3	768–1023	Stepper Motor Step Look Up Table (Indexed) Character Processing and Translation Print Head Firing
Page 2	51–767	Carriage Stepper Motor Acceleration Time Calculation and Storage Stepper Motor Deceleration
Page 1	256–511	Carriage Stepper Motor Drive
Page 0	0–255	Initialization–Jump-on-Reset Main Program Body External Status Switch Check Character Buffer Fill

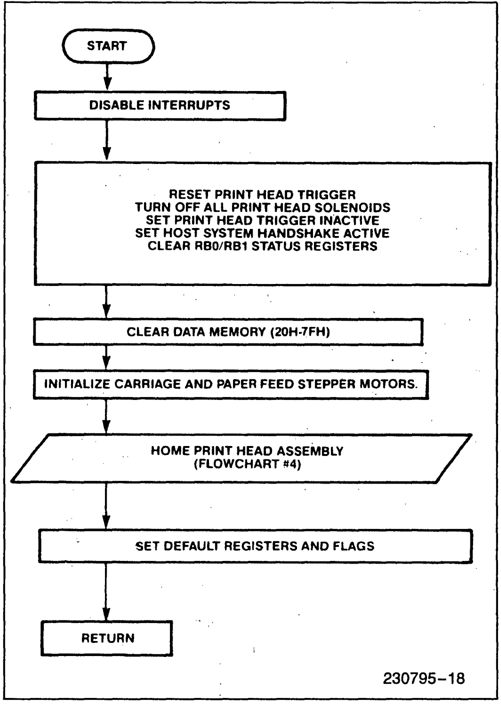
Figure 24. Program Memory Allocation Map

Software Functional Blocks

Below is a description and flow chart for each of the ten software blocks listed.

1. Power-On/Reset Initialization

The first operational part in Flow Chart No. 1 is the Power-On or Reset Initialization. Flowchart No. 2 illustrates the Initialization sequence in detail.


START
DISABLE INTERRUPTS
RESET PRINT HEAD TRIGGER
TURN OFF ALL PRINT HEAD SOLENOIDS
SET PRINT HEAD TRIGGER INACTIVE
SET HOST SYSTEM HANDSHAKE ACTIVE
CLEAR RB0/RB1 STATUS REGISTERS
CLEAR DATA MEMORY (20H-7FH)
INITIALIZE CARRIAGE AND PAPER FEED STEPPER MOTORS.
HOME PRINT HEAD ASSEMBLY (FLOWCHART #4)
SET DEFAULT REGISTERS AND FLAGS
RETURN
230795-18

Flow Chart No. 2. Power-On/Reset Initialization

Initialization first disables both interrupts. This is done as a precaution to prevent the system software from hanging-up should an interrupt occur before the proper registers and Data Memory values are initialized.

Initialization then deactivates the system electronics. This is also a precaution to protect the printer mechanism and includes the print head solenoid (trigger and data) lines and the stepper motor select lines. The host system handshake signals are activated to inhibit data transfer from the host until the printer is ready to accept data.

Next, Data Memory is cleared from 20H to 7FH. This includes the 80 byte Character Buffer, the 11 byte Stored Time Constants buffer, and the 4 bytes used as pseudo registers. The pseudo registers are Data Memory locations used as if they were registers. They serve as storage locations for step data used in accurately reversing the direction of the carriage stepper motor, and stabilizing either of the stepper motors not being driven.

The Data Memory locations 00H through 1FH are not cleared. These locations are Register Bank 0 (00H–07H), Program Stack (08H–17H), and Register Bank 1 (18H–1FH) (see Figure 19). Clearing the Program Registers or Stack would cause the initialization subroutine to become lost. The registers are used from the beginning of the program. Care is taken to initialize the registers and stack accurately prior to each program subroutine as required.

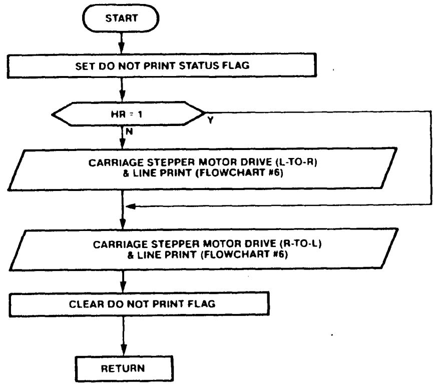
Upon power-on, it is necessary to initialize the two stepper motors, verify their operation, and locate the print head assembly in the left-most 'HOME' position. This sequence serves as a system checkout. If a failure occurs, the motors are deselected and the external status light is turned on. Each stepper motor is selected and energized for a sequence of four steps. This serves to align and stabilize each stepper motor's rotor position, preventing the rotor from skipping or binding when the first drive sequence begins.

At the end of each stepper motor's initialization, the last step data address is stored in one of the Data Memory pseudo registers. The last step data address is recalled at the beginning of the next corresponding stepper motor drive sequence, and used as the basis of the next step sequence. This ensures that the stepper motor always receives the exact next step data, in sequence, to guarantee smooth stepper motor motion. This also guarantees the motor never skips or jerks, which would misalign the start, stop, and character dot column positions. A stepper motor not being driven has its last phase data output held constant to stabilize it.

Following any stepper motor drive sequence of either motor, a delay of 30–60 ms occurs by switching the current to Hold Current, stabilizing the motor before it is deselected.

2. Home Print Head Assembly

At the end of the carriage stepper motor four step initialization, the output of the HR optical sensor is tested. The level of the HR signal indicates which drive sequence will be required to 'HOME' the print head assembly. If the print head assembly is to the right of HR, HR is high, the print head assembly need only be moved to from Right-to-Left until HR is low, then decelerated to locate the physical home position. If HR is low, the print head assembly must be moved first Left-to-Right until HR is high, then Right-to-Left to locate both the logical and physical 'HOME' positions. In each case, the software accelerates the carriage stepper motor, generating the Stored Time Constants then decelerates the stepper motor using the Stored Time Con-


START
SET DO NOT PRINT STATUS FLAG
HR = 1
Y
N
CARRIAGE STEPPER MOTOR DRIVE (L-TO-R) & LINE PRINT (FLOWCHART #6)
CARRIAGE STEPPER MOTOR DRIVE (R-TO-L) & LINE PRINT (FLOWCHART #6)
CLEAR DO NOT PRINT FLAG
RETURN

230795–19

Flow Chart No. 3. HOME Print Head Assembly

stants (see Background section above). Flow Chart No. 3 details the HOME print head assembly subroutine. Figures 13 and 18 illustrate the components of acceleration and print head assembly line motion.

The carriage stepper motor drive subroutines used to HOME the print head assembly and to print, are the same. A status flag, called Do-Not-Print, determines whether the Character Processing subroutine is called. The flag is set by the subroutine which calls the Carriage Stepper Motor Drive subroutine. Details of the carriage and paper feed stepper motor drive and character processing subroutines are covered separately below.

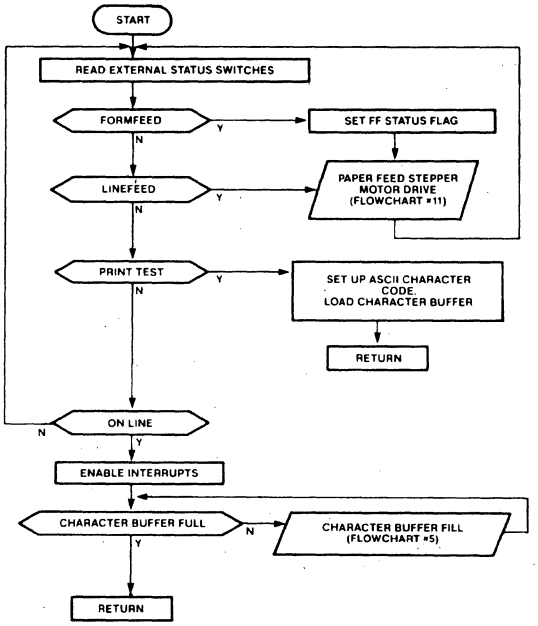
3. External Status Switch Check

Once the system is initialized and the print head is at the HOME position, the software enters a loop which continually monitors the four external status switches, and exits if any one is active. Flow Chart No. 4 details the External Status Switch Check subroutine.

Flow Chart No. 4. External Status Switch Check

If the LINEFEED or FORMFEED switch is set, the Paper Feed subroutine is called. The Paper Feed subroutine is discussed in detail below. If the ONLINE switch is set, the Character Buffer (CB) Fill subroutine is called.

If the Character Print TEST switch is set, the Data Memory Character Buffer (CB) is automatically loaded with the ASCII code sequence, beginning at 20H (a Space character), the first ASCII printable character code. The software then proceeds as if the CB had been filled by characters received from the host system. The


START
READ EXTERNAL STATUS SWITCHES
FORMFEED
Y
N
SET FF STATUS FLAG
LINEFEED
Y
N
PAPER FEED STEPPER MOTOR DRIVE (FLOWCHART #11)
PRINT TEST
Y
N
SET UP ASCII CHARACTER CODE. LOAD CHARACTER BUFFER
RETURN
ON LINE
N
Y
ENABLE INTERRUPTS
CHARACTER BUFFER FULL
N
Y
CHARACTER BUFFER FILL (FLOWCHART #5)
RETURN

230795–20

Flow Chart No. 4. External Status Switch Check

External Status Switch Check subroutine is exited and character printing begins. When the line has finished printing, a linefeed occurs (as shown in the main program Flow Chart No. 1) and the program returns to the External Status Switch Check subroutine. If the TEST switch remains active, the ASCII character code is incremented and program continues as before. This will eventually print all 95 ASCII printable characters. An example of the TEST printer output, the complete ASCII character code printed, is shown in Figure 25.

4. Character Buffer Fill

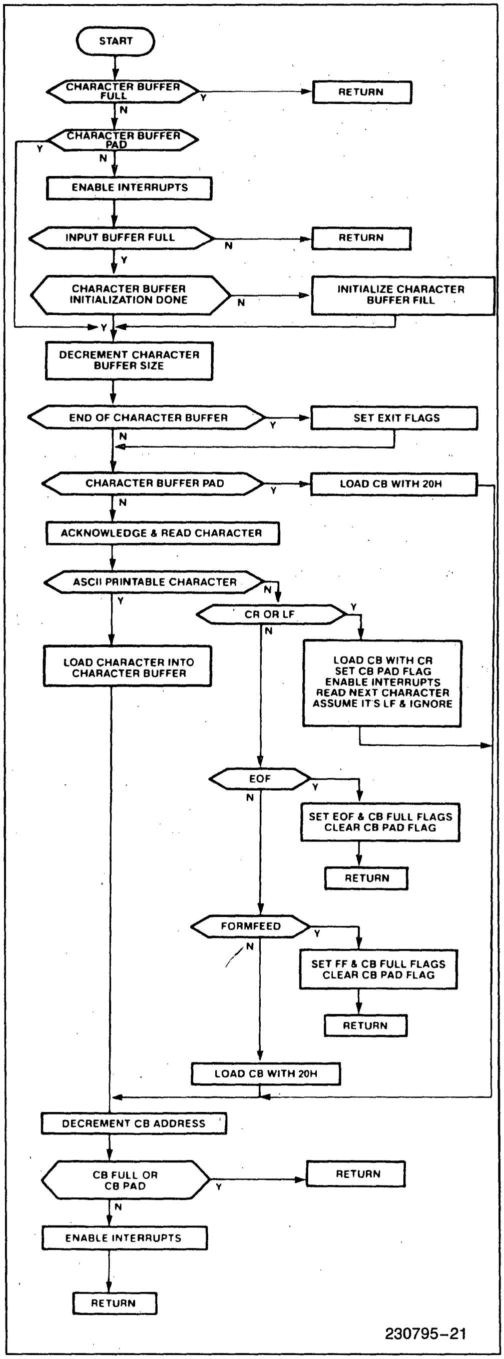
The Character Buffer (CB) Fill subroutine is called from three points within the main program; External Status Switch subroutine, and the Delay subroutine following the carriage and paper feed stepper motor drive subroutines. Flowchart No. 5 details the Character Buffer Fill subroutine operation.

The approximate 80 ms total pre-deselect delay at the end of each stepper motor drive sequence, 40 ms carriage and 40 ms paper feed stepper motor pre-deselect delay, is sufficient to load an entire 80 character line. Half the CB is filled at the end of printing the current line, and the second half is filled at the end of a paper feed. There is no time lost in printing throughput due to filling the character buffer.

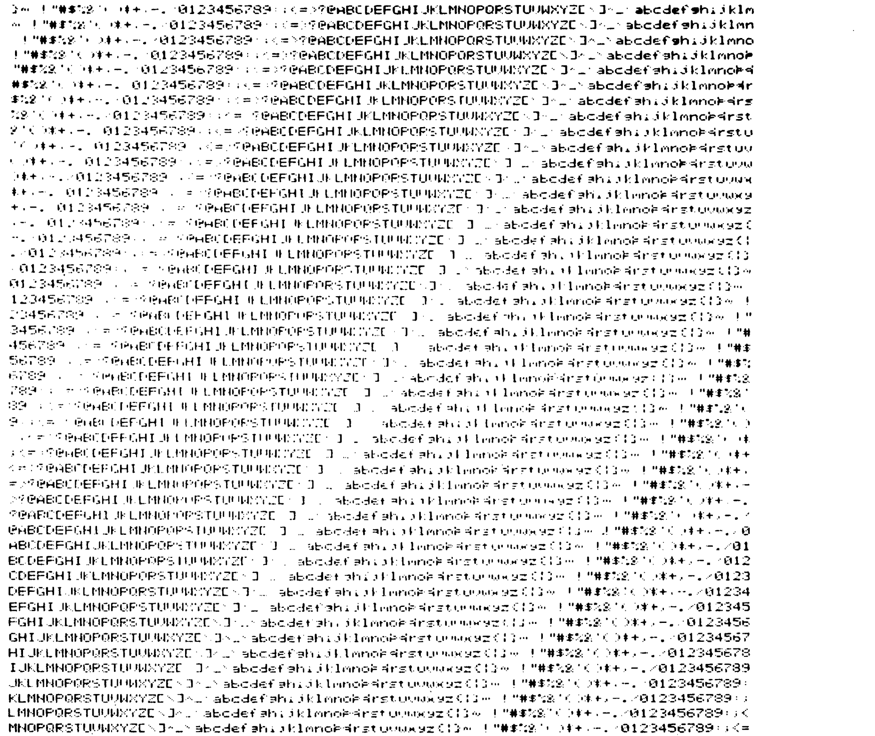
Character input is interrupt driven. When the IBF interrupt is enabled, a transmitted character sets the IBF interrupt and IBF Program Status flag. Three instructions make up the IBF interrupt service routine. This short routine disables further interrupts, sets the BUSY handshake line active, inhibiting further transmission by the host, and returns. The subroutine can be executed at virtually any point in the software flow without affecting the printer mechanism operation. Processing of the received character takes place during one of the three program segments mentioned above. The BUSY line remains active until the character is processed by the CB Fill subroutine.

The CB is 80 bytes from the top of Data Memory (30H–7FH). It is a FIFO for forward, left-to-right printing, and a LIFO for reverse, right-to-left, printing. Loading the CB always begins at the top, 7FH. One character may be loaded into the buffer each time the CB Fill subroutine is called.

The CB is always filled with 80 bytes of data prior to printing. If the total number of characters input up to a Carriage Return (CR)/Linefeed (LF), does not completely fill the CB, the CR code is loaded into the CB and the balance of the CB is padded with 20H (Space Character) until the CB is full. A Linefeed (LF) character following a Carriage Return is ignored. A LF is always forced at the end of a printed line. When the CB is full, the CB Full status byte flag is set and printing can begin.



Flow Chart No. 5. Character Buffer Fill



```
}~ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklm
~ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmn
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnop
"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopq
#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr
$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrs
%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrst
&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstu
'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuv
()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvw
)*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwx
*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxy
+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz
,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{
-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|
./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}
/0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ 
123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !
23456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"
3456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#
456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$
56789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%
6789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&
789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'
89:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'(
9:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()
:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*
;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+
<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,
=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,-
>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,-.
?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,-./
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,-./0
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,-./01
BCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,-./012
CDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,-./0123
DEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,-./01234
EFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,-./012345
FGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,-./0123456
GHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,-./01234567
HIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,-./012345678
IJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,-./0123456789
JKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,-./0123456789:
KLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,-./0123456789:;
LMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,-./0123456789:;<
MNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ !"#$%&'()*+,-./0123456789:;<=
```

230795-22

Figure 25. ASCII Character Code TEST Output and Print Example

A LF character alone is treated as a CR/LF at the end of a full 80 character line. This is a special case of a printed line and is handled during character processing for printing (see No. 7, Processing Characters for Printing, below). A Formfeed (FF) character sets the FF status byte flag. The flag is tested at each paper feed stepper motor drive subroutine entry.

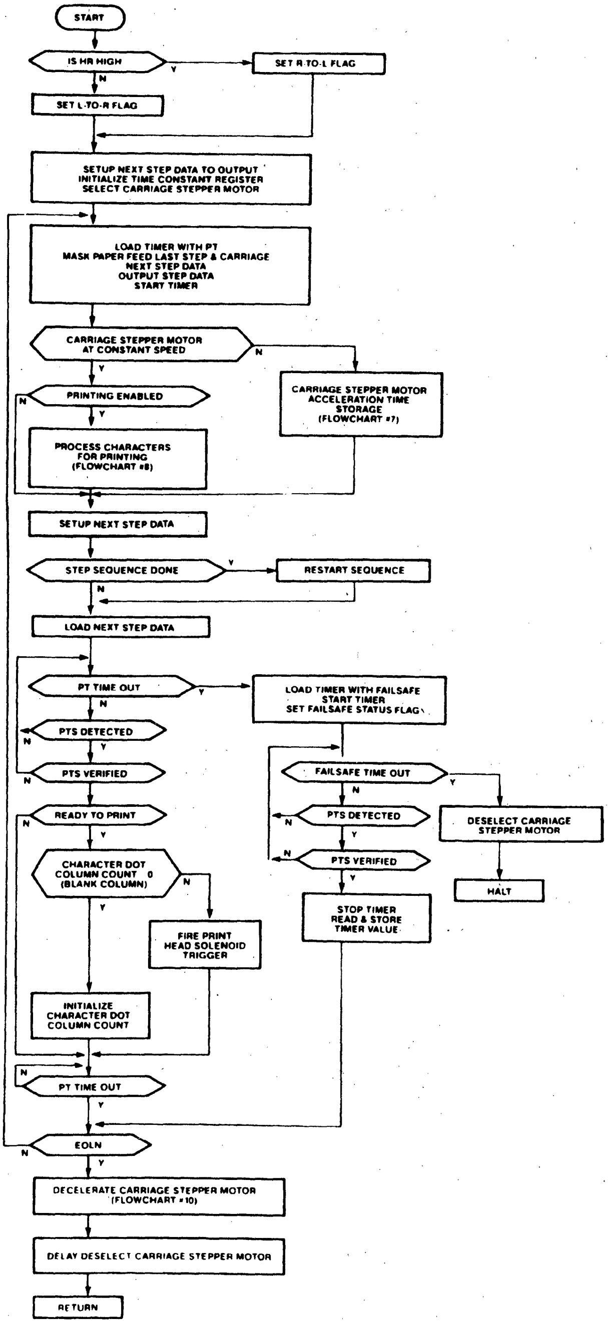
When the software is available to load the CB with a character, entry to the CB Fill subroutine checks three status flags; CB Full, CB Pad, and IBF flag. If the CB Full flag is set, the program returns without entering the body of the CB Fill subroutine. The CB Pad flag will cause another Space character to be loaded. If the IBF flag is not set, the program returns. If the IBF flag is set, the character is read from the Data Bus Buffer register, tested for printable or nonprintable ASCII code, and, if printable, loaded into the CB. If the character is a non-printable ASCII code and not an acceptable ASCII control code (CR, LF, FF, EOF), a 20H (Space Character) is loaded into the CB.

Exiting the CB Full subroutine with the CB Full or CB Pad flag set does not re-enable IBF interrupts or reset the BUSY line. If neither of these flags is set, exiting the CB Fill subroutine sets BUSY inactive and IBF interrupts are enabled. Once the CB Full status byte flag is set, IBF interrupts are disabled until the CB has been entirely emptied, the line printed, or the system Reset.

5. Carriage Stepper Motor Drive and Line Printing

The carriage stepper motor drive subroutine controls both L-to-R and R-to-L print head assembly motion. Upon entering the subroutine, the HR signal level is tested to determine the direction of print head assembly motion and the Direction status flag is set. The default control register values are loaded and balance of the default status flags are set for stepper motor control and character processing. The default control register values include PT and the step sequence look-up table start address for the direction indicated.

The direction flag is tested throughout the carriage stepper motor drive and character processing subroutines. This enables the same subroutines to control activities for either direction, simplifying and shorting the overall program. Flow Chart No. 6 illustrates the carriage stepper motor drive subroutine.



230795-23

Flow Chart No. 6. Carriage Stepper Motor Drive/Line Printing

Next, the carriage and paper feed stepper motor step data is initialized. The last step data output to the paper feed stepper motor is loaded into the Last Phase pseudo register. This data is masked with each step data output to the carriage stepper motor. Masking the step data in this manner guarantees the paper feed motor signals do not change as the carriage stepper motor is being driven.

Figure 26 illustrates the carriage stepper motor step sequence verses the actual step data output for clockwise rotation, Left-to-Right motion, and counterclockwise rotation, Right-to-Left print head assembly motion. An eight step sequence is depicted in the figure. Note that the sequence for Right-to-Left motion is the reverse of the sequence for Left-to-Right motion. Note also, that for the L-to-R sequence step 4 is the same as step 0, step 5 the same as step 1, etc., through step 7 matching step 3. The four step sequence simply repeats itself until the motor is stopped via the Deceleration subroutine.

L-to-R Motion Sequence	Phase/Step Data (3 2 1 0)	R-to-L Motion Sequence	BCD (3 2 1 0)
0	1 0 0 1	7	0 0 0 0
1	1 0 1 0	6	0 0 0 1
2	0 1 1 0	5	0 0 1 0
3	0 1 0 1	4	0 0 1 1
4	1 0 0 1	3	0 1 0 0
5	1 0 1 0	2	0 1 0 1
6	0 1 1 0	1	0 1 1 0
7	0 1 0 1	0	0 1 1 1

Figure 26. Carriage Stepper Motor Phase/Step Data

When the carriage stepper motor is driven for a specific direction of print head assembly motion, the step sequence must be consistent for the motion to be smooth and accurate. The same holds true for the transition from one direction of motion to the other. Since the sequence for one direction is the opposite for the other direction, incrementing the sequence for L-to-R and decrementing for R-to-L provides the needed step data flow. For example, referring to Figure 26, if the print head assembly moved L-to-R and the last step output was #1, the first step for R-to-L motion would be #7. Thus, when the carriage stepper motor is initialized for a clockwise (L-to-R) or counterclockwise (R-to-L) rotation, the last step sequence number is incremented or decremented to obtain the proper next step. In this way, the smooth motion of the stepper motors is assured.

The step data is referenced indirectly via the step sequence number. The step data is stored in a Program Memory look-up table whose addresses correspond to the step sequence numbers. For example, as shown in Figure 26, at location 0 the step data "1001" is stored. This method is particularly well suited to the UPI-42AH software. The UPI-42AH features a number of instructions which perform an indirect move or data handling operation. One of these instructions, MOVP3A,@A, unlike the others, allows data to be moved from Page 3 of Program Memory to any other page of Program Memory. This instruction allows the step data to be centrally located on Page 3 of Program Memory and accessed by various subroutines.

Each time the carriage stepper motor step data is output, the step data lookup table address is incremented or decremented, depending upon the direction of rotation, and tested for restart of the sequence. The address is tested because the actual step data, Figure 26, is not a linear sequence and thus is not an easily testable condition for restarting the sequence. The sequence number is tested for rollover of the sequence count from 03H to 04H and clockwise motor rotation via the Jump on Accumulator Bit instruction (JBn), with 00H loaded to restart the sequence. The same bit is tested when decrementing the sequence count for counterclockwise motor rotation, R-to-L motion, because the count rolls over from 00H to 0FFH, with 03H loaded to restart the sequence.

At this point the UPI-42AH Timer/Counter is loaded, the step signal is output, and the timer started. The next step data to be output has been determined and the At-Speed flag is tested for entry to one of two subroutines; Stepper Motor Acceleration Time Storage or Character Processing.

The first entry to the Acceleration Time Storage subroutine initializes the subroutine and returns. All other entries to one of the two subroutines perform the necessary operations, detailed below (Blocks 6 and 7), and returns. The program loops until the PT times out or the PTS level change is detected. PTS is tied to T0 of the UPI-42AH. The level present on T0 is directly tested via conditional jump instructions. The software loops on polling the timer Time Out Program Status flag and the T0 input level.

As described in the Background section above (shown in Figure 13), if PT times out before PTS is detected, the software waits for PTS before outputting the next step signal. If PT times out before PTS, a second timer count value is loaded into the UPI-42AH timer. The timer value is called "Failsafe." This is the maximum time the stepper motor can be selected, with no rotor motion, and not damage the motor. If PTS is not detected, either the carriage stepper motor is not rotating or the optical sensor is defective. In either case, program execution halts, the motor is deselected, and the external status light is turned on to indicate a malfunction. A system reset is required to recover from this condition. The Failsafe time is approximately 20 milliseconds, including PT.

The Failsafe time loop also serves as a means of tracking the elapsed time between PT time out and PTS. Entry to the Failsafe time loop sets the Failsafe/Constant Time Window status flag. This flag is tested by the Acceleration Time Storage subroutine for branching to the proper time storage calculation to be performed (see Figure 13 and Block 6 below for further description).

During the Failsafe timer loop, if PTS is detected and verified as true, the Failsafe timer value is read and stored in the Time Storage register. This value is used during the next Acceleration Time Storage subroutine call to calculate the Stored Time Constant (see Block 6 below). If PTS is invalid, the flow returns to the timer loop just exited, again waiting for PTS or Failsafe time out.

During the PT time loop, if PTS is detected and verified, the Sync flag is tested for entry to the print head solenoid firing subroutine. This flag is set by the first entry to the Character Processing subroutine. The flag synchronizes the solenoid firing with character processing. Only if characters are processed for printing will the solenoids be enabled, via the Sync flag, for firing. This prevents the solenoids from being fired without valid character dot data present.

As described in the Background section "Relationship Between PTS and PT," PTS is the point of peek angular velocity within a step of the motor. After PTS is detected the motor speed ramps down, compensating for the overshoot of the rotor motion. PTS is the optimum time for print head solenoid firing, as shown in Figure 13. This is the stable point of motor rotation and, thus, the print head assembly motion. If PTS is detected during PT, printing is enabled, the Sync flag is set, and the solenoid trigger is fired.

The firing of the solenoid trigger, following PTS, is very time critical. The time between PTS and solenoid firing must be consistent for accurate dot column alignment throughout the printed line. The software is designed to meet this requirement by placing all character processing and motor control overhead before the solenoid firing subroutine is called. The actual instruction sequence which fires the print head solenoid trigger is plus or minus one instruction for any call to the subroutine.

Once the timer loop is complete, the software tests for Exit conditions. If the Exit conditions fail, the software loops to output the next step signal, starts the PT timer, and continues to accelerate the carriage stepper motor, or process, and print characters. If the Exit test is true, the carriage stepper motor is decelerated to a fixed position, and the program returns to the main program flow (see Flowchart 1).

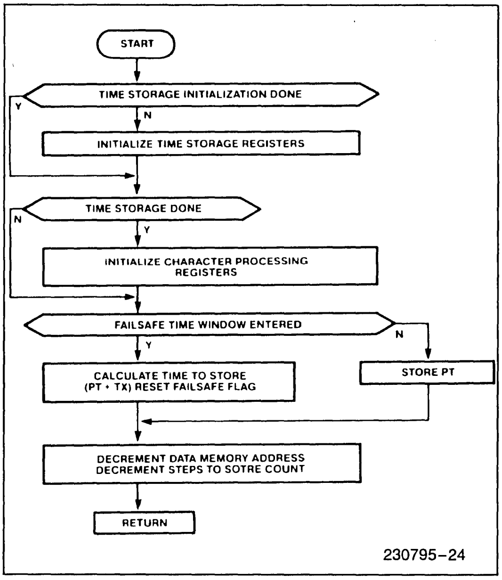
The exit conditions are different for the two directions of print head assembly motion. For L-to-R printing, if a Carriage Return (CR) character code is read from CB, the carriage stepper motor drive terminates and the motor is decelerated to a fixed position. There are two conditions for terminating carriage stepper motor drive upon detecting a CR during L-to-R motion. If less than half a character line (40 characters) has been printed, the print head assembly returns to the HOME position to start the next printed line. Otherwise, the print head assembly continues to the right-most position for a full 80 character line, and then begins printing the next line from R-to-L. R-to-L printing always returns the print head assembly to the HOME position before the next line is printed L-to-R. When HR is high, character printing always stops and the carriage stepper motor drive subroutine exits to the deceleration subroutine.

6. Accelerate Stepper Motor Time Storage

As described above, when the carriage stepper motor is accelerated the step time required to guarantee the motor is at a constant rate of speed translates to a specific distance traveled by the print head assembly (see Figure 18). In order to position the print head assembly accurately for bi-directional printing, the distance traveled during deceleration must be the same as during acceleration. The Carriage Motor Acceleration Time Storage subroutine calculates the step times needed to accelerate the carriage stepper motor, and stores them in Data Memory for use as PT during deceleration.

The first call of the Carriage Stepper Motor Acceleration Time Storage subroutine initializes the required registers and status flags. The time calculation begins with the second carriage stepper motor step signal output. The program returns to the carriage stepper motor drive subroutine and loops on PT. Each subsequent call of the Acceleration Time Storage subroutine tests the Failsafe/Constant flag and branches accordingly (see Flow Chart 7). The Acceleration Time Storage subroutine has two parts which correspond to PTS leading or PTS lagging PT.

If the Failsafe/Constant flag is set, PTS lagged PT. The time from PT time out to PTS, Tx (see Figure 13), must be added to the PT and stored in Data Memory. As described above, if PT lagged PT, the Failsafe time is loaded and PTS is again polled during the time loop. When PTS occurs within the Failsafe time, the timer is stopped and the timer value stored. The UPI-42AH timer is an up timer, which means that the value stored is the time remaining of the Failsafe time when PTS occurred. The elapsed time must be calculated by subtracting the time remaining (the value stored) from the Failsafe time constant. This is done in software by using two's complement arithmetic. If the Failsafe flag is not set PTS led PT, and PT is the Stored Time Constant stored.


START
TIME STORAGE INITIALIZATION DONE
Y
N
INITIALIZE TIME STORAGE REGISTERS
TIME STORAGE DONE
N
Y
INITIALIZE CHARACTER PROCESSING REGISTERS
FAILSAFE TIME WINDOW ENTERED
N
Y
CALCULATE TIME TO STORE (PT + TX) RESET FAILSAFE FLAG
STORE PT
DECREMENT DATA MEMORY ADDRESS
DECREMENT STEPS TO SOTRE COUNT
RETURN
230795-24

Flow Chart No. 7. Carriage Stepper Motor Acceleration Time Storage

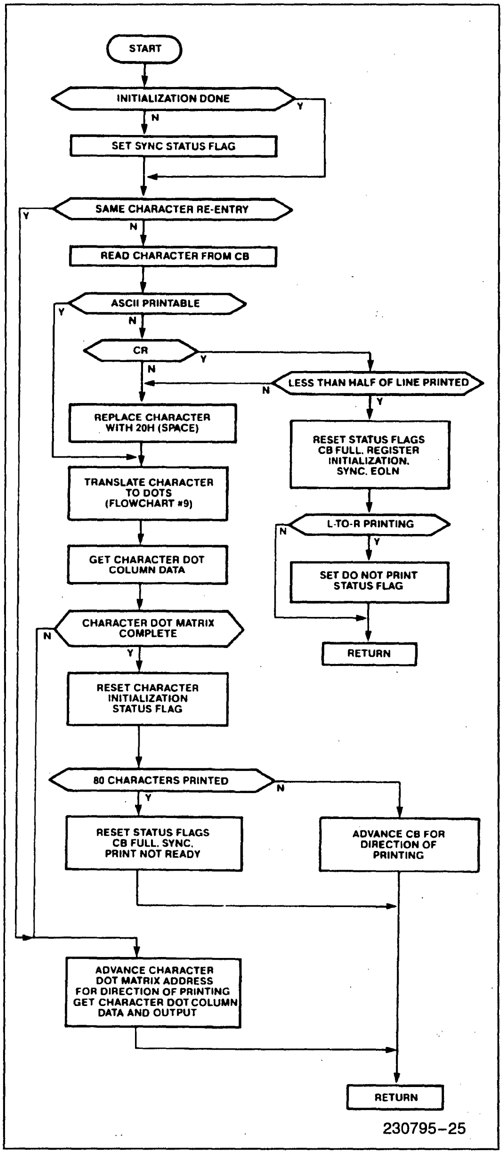
Indirect addressing of Data Memory is used to reference the Stored Time Constant Data Memory location. The Data Memory location address is decremented each time the Acceleration Time Storage subroutine is exited and a Stored Time Constant has been generated.

The last Acceleration Time Storage subroutine exit sets the At-Speed status flag and initializes the character processing registers and flags.

7. Process Characters for Printing

The Character Processing subroutine is entered only if the Home Reset (HR) optical sensor signal is high and printing is enabled. Otherwise, the software simply returns to the Carriage Stepper Motor Drive subroutine. There are two cases when printing is not enabled; during the HOME subroutine operation, and when the print head assembly returns to the HOME position after printing less than half an 80 character line. If printing is enabled, the Sync status flag is set.

All character processing operations use the second UPI-42AH Data Memory Register Bank, RB1. Register Bank 1 is independent of Data Memory Register Bank 0, used for stepper motor control. The use of two independent register banks greatly simplifies the software flow, and helps to ensure the accuracy of event sequences that must be handled in parallel. Each register bank must be initialized only once for any entry to either the Carriage Stepper Motor Drive or Character Processing subroutines. A single UPI-42AH Assembly Language instruction selects the appropriate register bank. Initializing the character processing registers includes loading the maximum character count (80), dot matrix size count (6), and CB start address. The CB start address is print direction dependent, as described in Block 4, above.


START
INITIALIZATION DONE
Y
N
SET SYNC STATUS FLAG
SAME CHARACTER RE-ENTRY
Y
N
READ CHARACTER FROM CB
ASCII PRINTABLE
Y
N
CR
Y
N
LESS THAN HALF OF LINE PRINTED
N
Y
REPLACE CHARACTER WITH 20H (SPACE)
RESET STATUS FLAGS CB FULL. REGISTER INITIALIZATION. SYNC. EOLN
TRANSLATE CHARACTER TO DOTS (FLOWCHART #9)
L-TO-R PRINTING
N
Y
SET DO NOT PRINT STATUS FLAG
GET CHARACTER DOT COLUMN DATA
CHARACTER DOT MATRIX COMPLETE
N
Y
RETURN
RESET CHARACTER INITIALIZATION STATUS FLAG
80 CHARACTERS PRINTED
N
Y
RESET STATUS FLAGS CB FULL. SYNC. PRINT NOT READY
ADVANCE CB FOR DIRECTION OF PRINTING
ADVANCE CHARACTER DOT MATRIX ADDRESS FOR DIRECTION OF PRINTING GET CHARACTER DOT COLUMN DATA AND OUTPUT
RETURN
230795-25

Flow Chart No. 8. Process Characters for Printing

Character processing reads a character from the CB, tests for control codes, translates the character to dots, and conditionally exits, returning to the Carriage Stepper Motor Drive subroutine. Flow Chart 8 details the character processing subroutine.

Each character requires six steps of the carriage stepper motor to print; five for the 5 character dot columns and 1 for the blank dot column between each character. Reading a character from the CB and character-to-dot pattern translation takes place during the last character dot column, or blank column, time.

The first character line entry to the Character Processing subroutine appears to the software as if a last character dot column (blank column) had been entered. The next character, in this case the first character in the line, is translated and printing can begin. This method of initializing the Character Processing subroutine utilizes the same software for both start-up and normal character flow. Once a character code has been translated to a dot matrix pattern starting address in the look-up table, all subsequent entries to the Character Processing subroutine simply advance the dot column data address and outputs the data.

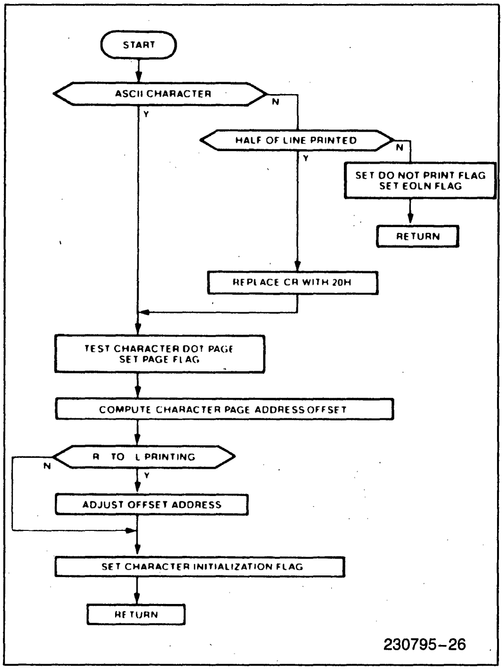
The decision to translate the character to dots during the blank column time was an arbitary one. As was the choice of the blank column following rather than preceding the actual character dot matrix printing.

8. Translate Character-to-Dots

Characters-to-dot pattern translation involves converting the ASCII code into a look-up table address, where the first of the five bytes of character dot column data is stored. The address is then incremented for the next column of dot pattern data until the full character has been printed.

The dot pattern look-up table occupies two pages, or approximately 512 bytes of Program Memory. A printable ASCII character is tested for its dot pattern location page and the offset address, from zero, on that page. Both the page test and page offset calculations use two's complement arithmetic, with a jump on carry or not carry causing the appropriate branching. Once the pattern page and address are determined the indirect addressing and data move instructions are used to read and output the data to the print head solenoids. Flowchart 9 details the Character-to-Dots Translation subroutine.

In the case of R-to-L printing, although the translation operation is the same, the character is printed in reverse. This requires that the character dot pattern address be incremented by five, before printing begins, so that the first dot column data output is the last dot column data of the character. The dot pattern look-up table address is then decremented rather than incremented, as in L-to-R printing, for the balance of the character. Translation still takes place during the last character dot column, the blank column, and the blank column follows the character matrix.

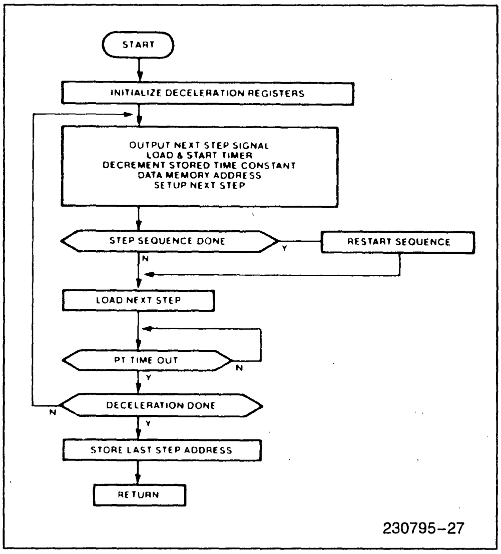


Flow Chart No. 9. Translate Character-to-Dots

Only one control code, a Carriage Return (CR), is encountered by the character translation subroutine. Linefeed (LF) characters are stripped off by the CB Fill subroutine. If a CR code is detected the software tests for a mid-line exit condition; less than half the line printed exits the stepper motor drive subroutine and HOMEs the print head assembly before printing the next line. If the test fails, more than half the line has been printed, the CR is replaced by a 20H (Space character) and printing continues for the balance of the line; the space characters padding the CB are printed.

As mentioned above, the character dots are printed and the print head trigger is fired when the PTS signal is detected and verified and the carriage stepper motor is At Speed.

When the character to print test fails the CB Buffer size count equals zero, the Carriage Stepper Motor Drive subroutine exit flags are set, and the flow passes to the Deceleration and Delay subroutines and programs returns to the main program flow.



Flow Chart No. 10. Decelerate Carriage Stepper Motor

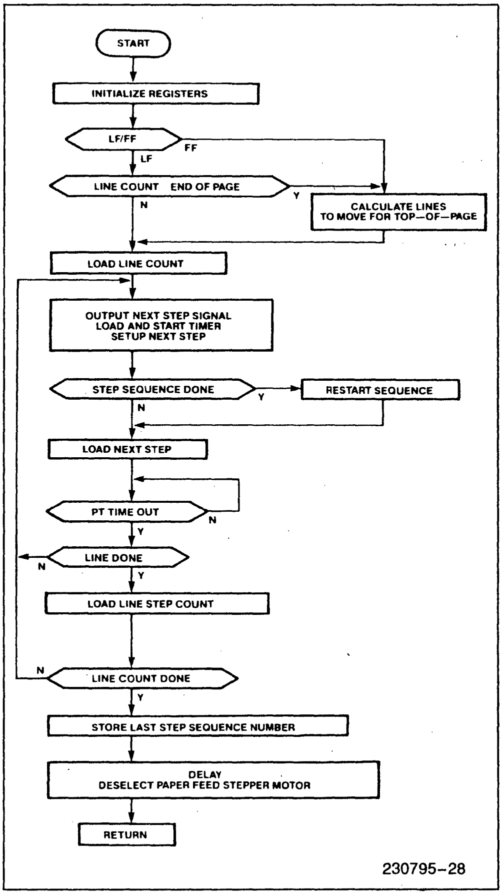
9. Decelerate Carriage Stepper Motor

The transition from the Carriage Stepper Motor Drive subroutine to the Deceleration subroutine outputs the next step signal in sequence, and then initializes the Deceleration subroutine registers; Stored Time Constants Data Memory buffer end address and size. The Stored Time Constant Buffer is a LIFO for deceleration of the carriage stepper motor. The buffer size is used as the step count. When the step count decrements to zero, the step signal output is terminated, and the last step sequence number is stored in the carriage stepper motor Next Step pseudo register. The last step sequence number is recalled, during initialization of the next carriage stepper motor drive, as the basis of the next step data signal to be output. See Flow Chart 10.

When the carriage stepper motor is decelerated, Failsafe protection and PTS monitoring are not necessary. The Deceleration subroutine acts as its own failsafe mechanism. Should the stepper motor hang-up, the subroutine would exit and deselect the motor in sufficient time to protect the motor from burnout. Since neither Failsafe nor print head solenoid firing take place during deceleration, PTS is not needed. PT is replaced by the Stored Time Constant values in Data Memory. The Deceleration subroutine determines the next step signal to output, loads the Timer with the Stored Time Constant, starts the UPI-42AH Timer, and loops until time out. The subroutine loops to output the next step until all of the Stored Time Constants have been used. The program returns to the Carriage Stepper Motor Drive subroutine and the motor is deselected following the Delay subroutine execution. The Delay subroutine is called to stabilize the stepper motor before it is deselected. During the DELAY subroutine, the IBF interrupt is enabled and characters are processed. A paperfeed is forced following the carriage stepper motor being deselected.

10. Paper Feed Stepper Motor Drive

The paper feed stepper motor subroutine outputs a predefined number of step signals to advance the paper, in one line increments, for the required number of lines. The number of step signals per line increment is a function of the defined number of lines per inch, given the distance the paper moves in one step. Figure 16 lists three step (or pulse) count and line spacing configurations, as well as the distance the paper moves in one step. Standard 6 lines per inch spacing has been implemented in this Application Note (Appendix B details how variable line spacing could be implemented). Flow Chart 11 illustrates the Paper Feed subroutine.



Flow Chart No. 11. Paper Feed Stepper Motor Drive

The number of lines the paper is to be moved is called the "Line Count." The Line Count defaults to one unless the Formfeed flag is set, or the total number of lines previously moved equals a full page. The default total lines per page for this application is 66. When the total number of lines moved equals 66, the paper is moved to the top of the next page. The Top-of-Page is set at power-on or reset.

If the Formfeed flag has been set in the Character Buffer Fill subroutine, the software calculates the number of lines needed for a top of next page paper feed. The resulting line count is loaded in the Line Count Register. The Paper Feed subroutine loops on the line count until done and then returns to the main program body.

Once the Paper Feed subroutine is complete, the software loops to test the End of File (EOF) Flag (see Flow Chart 1). If EOF is set, the print head assembly is moved to the HOME position, the program again enters the External Status Switch Test subroutine, and begins polling the external status switches. If EOF is not set, the program directly calls the External Status Switch Check subroutine, and the program repeats for the next line.

CONCLUSION

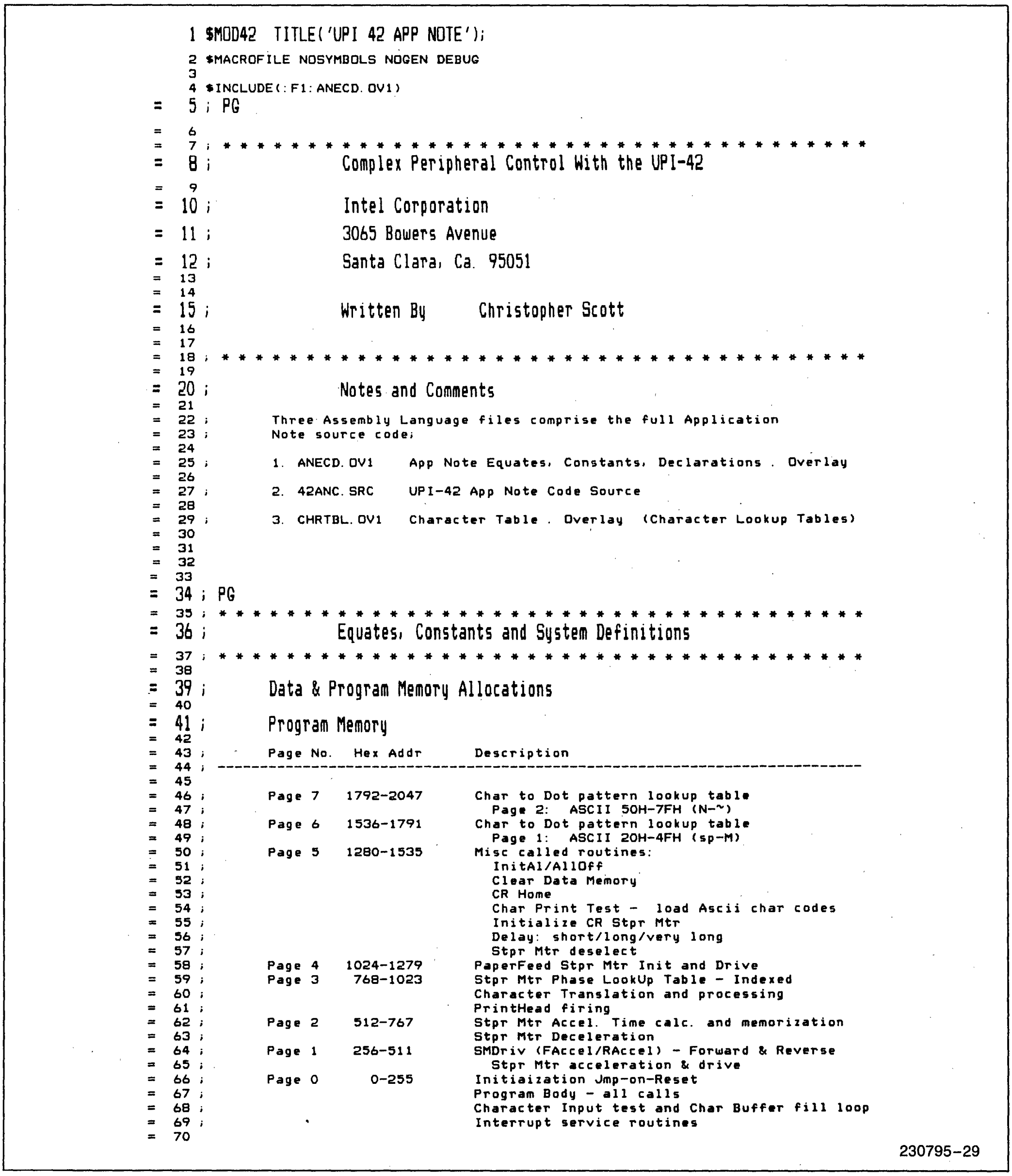
Although the full speed, 12 MHz, of the UPI-42AH was used, the actual speed required is approximately 8–9 MHz. 1400 bytes of the available 2k bytes of Program Memory were used; 500 bytes for the 95 character ASCII code dot pattern look-up table, 900 bytes for operational software. This means that the UPI-42AH has excess processing power and memory space for implementing the additional functions such as those listed below and discussed in Appendix B.

Special Characters or Symbols
Lower Case Descenders
Inline Control Codes
Different Character Formats
Variable Line Spacing

The software developed for this Application Note was not fully optimized and could be further packed by combining functions. This would require creating another status register, which could also serve to implement some of the features listed above. Since the full 16 byte stack is not used for subroutine nesting, there are 6–8 bytes of Program Stack Data Memory that could be used for this purpose. In several places, extra code was added for clarity of the Application Note. For example, each status byte flag is set with a separate instruction, using a equate label, rather than setting several flags simultaneously at the same point in the code.

This Application Note has demonstrated that the UPI-42AH is easily capable of independently controlling a complex peripheral device requiring real time event monitoring. The moderate size of the program required to implement this application attests to the effectiveness of the UPI-42AH for peripheral control.

APPENDIX A
SOFTWARE LISTING



```
  $MOD42  TITLE('UPI 42 APP NOTE');
  $MACROFILE NOSYMBOLS NOGEN DEBUG

  $INCLUDE(:F1:ANECD.OV1)
= ; PG
=
= ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
= ;              Complex Peripheral Control With the UPI-42
=
= ;              Intel Corporation
= ;              3065 Bowers Avenue
= ;              Santa Clara, Ca. 95051
=
=
= ;              Written By      Christopher Scott
=
=
= ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
=
= ;              Notes and Comments
=
= ;        Three Assembly Language files comprise the full Application
= ;        Note source code;
=
= ;        1. ANECD.OV1     App Note Equates, Constants, Declarations . Overlay
=
= ;        2. 42ANC.SRC     UPI-42 App Note Code Source
=
= ;        3. CHRTBL.OV1    Character Table . Overlay  (Character Lookup Tables)
=
=
=
=
= ; PG
= ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
= ;              Equates, Constants and System Definitions
= ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
=
= ;        Data & Program Memory Allocations
=
= ;        Program Memory
=
= ;        Page No.  Hex Addr       Description
= ; -------------------------------------------------------------------------
=
= ;        Page 7   1792-2047      Char to Dot pattern lookup table
= ;                                   Page 2:  ASCII 50H-7FH (N-~)
= ;        Page 6   1536-1791      Char to Dot pattern lookup table
= ;                                   Page 1:  ASCII 20H-4FH (sp-M)
= ;        Page 5   1280-1535      Misc called routines:
= ;                                   InitAl/AllOff
= ;                                   Clear Data Memory
= ;                                   CR Home
= ;                                   Char Print Test -  load Ascii char codes
= ;                                   Initialize CR Stpr Mtr
= ;                                   Delay: short/long/very long
= ;                                   Stpr Mtr deselect
= ;        Page 4   1024-1279      PaperFeed Stpr Mtr Init and Drive
= ;        Page 3    768-1023      Stpr Mtr Phase LookUp Table - Indexed
= ;                                Character Translation and processing
= ;                                PrintHead firing
= ;        Page 2    512-767       Stpr Mtr Accel. Time calc. and memorization
= ;                                Stpr Mtr Deceleration
= ;        Page 1    256-511       SMDriv (FAccel/RAccel) - Forward & Reverse
= ;                                   Stpr Mtr acceleration & drive
= ;        Page 0      0-255       Initiaization Jmp-on-Reset
= ;                                Program Body - all calls
= ;                                Character Input test and Char Buffer fill loop
= ;                                Interrupt service routines
=
```

230795-29

```
                   =  71 ; PG
                   =  72 ; -------------------------------------------------------------------------
                   =  73 ;        Data Memory
                   =  74 ; -------------------------------------------------------------------------
                   =  75 ;         Dec.        Hex           Description
                   =  76 ;        -------    --------       -------------------------------------------
                   =  77 ;    TOP
                   =  78 ;    ----------------------
                   =  79 ;       48-127     2F-7FH          80 Character Line Buffer
                   =  80 ;        37-47     25-2EH          Stpr Mtr Accel/Decel time, memorization
                   =  81 ;           36        24H          Unused
                   =  82 ;           35        23H          Char Print test ASCII code start tmp store
                   =  83 ;           34        22H          LF SM last Phz Inderect Addr psuedo reg
                   =  84 ;           33        21H          CR SM Forward/Reverse last Phz psuedo reg
                   =  85 ;           32        20H          Psuedo Reg:  Last Phase of stpr mtr not
                   =  86 ;                                               being driven
                   =  87 ;        24-31     18-1FH          Register Bank 1:  Character Handling
                   =  88 ;         8-23      8-17H          8 Level Stack
                   =  89 ;          0-7      0-07H          Register Bank 0:  Stpr Mtr F/R Accel/Drive
                   =  90 ;    ----------------------
                   =  91 ;    BOTTOM
                   =  92
                   =  93 ;        Data Memory Equates:
                   =  94 ; -------------------------------------------------------------------------
                   =  95
0050               =  96 CHBFSZ  EQU     50H             ;char buffer size 0-79 = 80
00D9               =  97 HlfCpl  Equ     0D9H            ;Cpl(1/2 CbBfSz) => cpl of 27H = 0D9H
                   =  98
007F               =  99 FCBfSt  Equ     7fH             ;start of char buffer
0080               = 100 FCBfIS  Equ     80H             ;init CB strt-allows xtra Dec by 1
002F               = 101 RCBfIS  Equ     2FH             ;init CB strt-allows xtra Inc by 1
0051               = 102 ChBfIS  Equ     81              ;load char cnt reg w/char bufr Init Size
                   = 103
002F               = 104 ENDBUF  EQU     2FH             ;END OF CHAR BUFFER
000B               = 105 ASBfSz  EQU     0BH             ;Accelerate stpr mtr buf count
000A               = 106 DSBfSz  Equ     0AH             ;Decelerate stpr mtr buf count
002F               = 107 SMBFST  EQU     2FH             ;STPR MTR BUFFER START
0025               = 108 SMBEnd  Equ     25H             ;Stpr Mtr Data Memory Address end
007F               = 109 DMTop   Equ     7FH             ;Data Memory Top
005D               = 110 DMSize  Equ     93              ;Data Memory Size (less two working reg's)
                   = 111
0020               = 112 LastPh  Equ     20H             ;last phz psuedo reg addr
0021               = 113 CPSAdr  EQU     21H             ;CR phz psuedo reg
0022               = 114 LPSAdr  Equ     22H             ;LF phz psuedo reg
0023               = 115 PTAscS  Equ     23H             ;Char Print Test code start tmp store
                   = 116
                   = 117 ; PG
                   = 118 ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                   = 119 ;        Register allocation
                   = 120 ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                   = 121
                   = 122 ;        All Indirect Data Memory Addressing via @Rn Inst must use
                   = 123 ;          only registers 0 & 1 of either register bank.  Any other will
                   = 124 ;          be rejected by the Assembler
                   = 125 ;        Last character in lable indicates Register Bank referenced
                   = 126
                   = 127 ;        Register Bank 0
                   = 128 ; -------------------------------------------------------------------------
0000               = 129 TmpR00  Equ     R0              ;RB0 Temporary Register
0001               = 130 TStrR0  EQU     R1              ;Store Time Register RB0
0002               = 131 GStR20  EQU     R2              ;General Status Register RB0
0003               = 132 PhzR30  EQU     R3              ;Stpr Mtr Phase Register RB0
0004               = 133 CntR40  Equ     R4              ;Count Reg. Phase count-Stpr Mtr loops
                   = 134                                 ;  Accel/Decel Count
0005               = 135 TConR0  Equ     R5              ;Time constant reg RB0
0006               = 136 LnCtR0  Equ     R6              ;Line count
                   = 137
0007               = 138 OpnR70  EQU     R7              ;available
                   = 139
                   = 140 ;        Register Bank 0 Data Memory Address
                   = 141 ; -------------------------------------------------------------------------
```

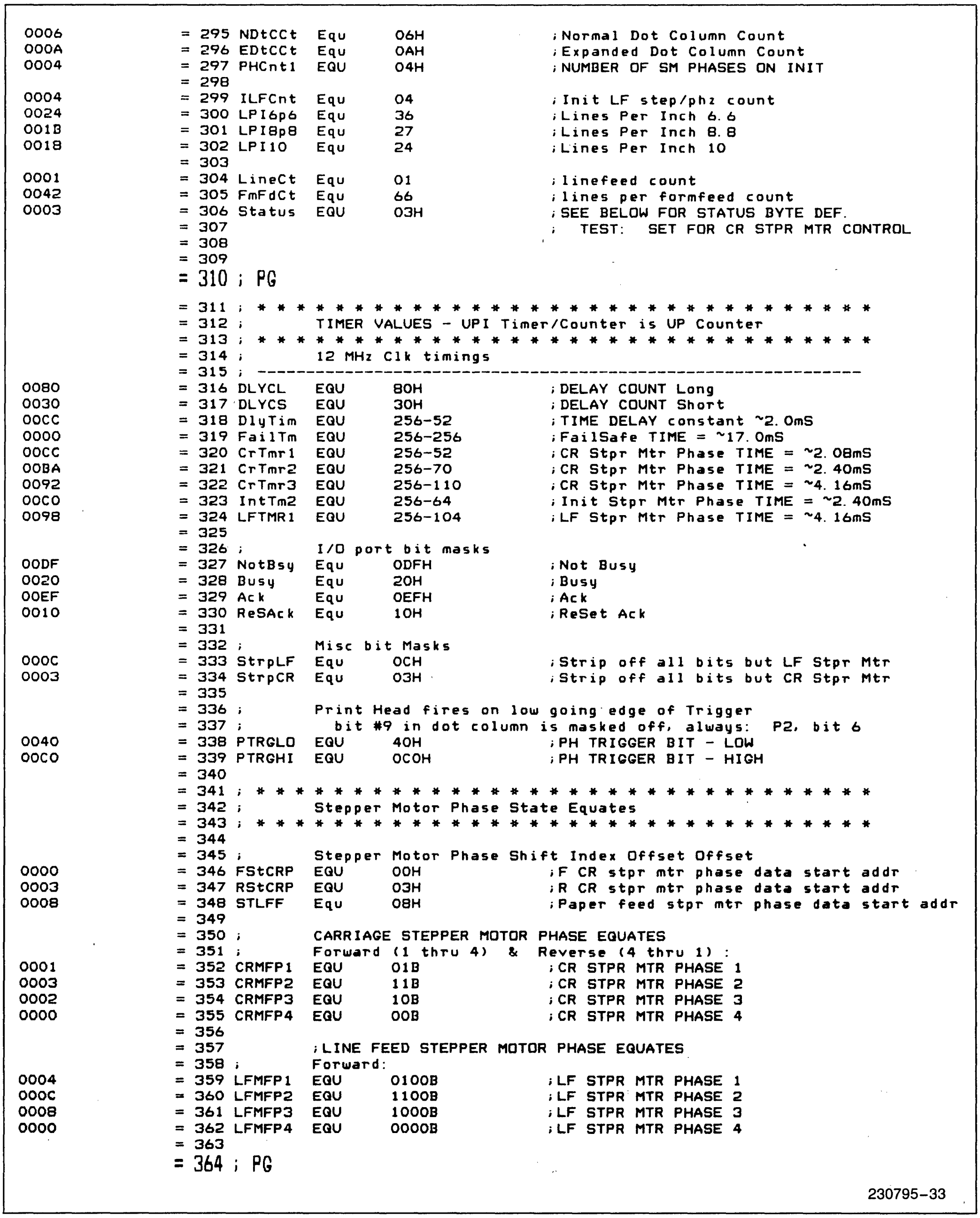
230795-30

```
0000     =  TmpA00  Equ     00H          ;Temporary Register DM address
0001     =  TStrA0  EQU     01H          ;Time Store Register  DM address
0002     =  GStRAd  Equ     02H          ;RBO Char Status Reg  DM address
0003     =  PhzA20  EQU     03H          ;Stpr Mtr Phase Register DM address
0004     =  CntRA0  Equ     04H          ;Count Reg. Phase count-Stpr Mtr loops
         =                               ;   Accel/Decel Count DM address
0005     =  TConA0  Equ     05H          ;Time constant reg DM address
0006     =  LnCtA0  Equ     06H          ;Line Count Register DM address
         =
0007     =  OpnA70  EQU     07H          ;available
         =
         =
         = ; PG
         = ; -------------------------------------------------------------------
         = ;        RBO  Status Byte Bit Definition
         = ; -------------------------------------------------------------------
         =
         = ;        Bit             Definition
         = ;        -----           ------------------------------------------
         = ;        7               Stpr Mtr Direction:  L-to-R = 1, R-to-L = 0
         = ;        6               1 = Sink / 0 = Not Sinked, Print Head Init and Fire
         = ;        5               Stpr Mtr at speed and CR not left of Home
         = ;        4               Accel/Decel Init, 1 = Done / 0 = Not Done
         =
         = ;        3               1 = FailSafe / 0 = Constant, Time Window
         = ;        2               1 = Form Feed / 0 = Line Feed
         = ;        1               1 = Do Not Print / 0 = Print
         = ;        0               FAccel/DAccel drive Ready = 1/NotRdy = 0 (exit
         = ;                          drive & decel stpr mtr)
         =
         = ;        Bit Masks:      RBO
         = ;        Stepper Motor control bit masks function on GStR10
         = ; ---------------------------------------------------------
         =
         =  LRPrnt  Equ     80H          ;Left to Right Printing (ORL)
         =  RLPrnt  Equ     7FH          ;Right to Left Printing (ANL)
         =  SnkSet  Equ     40H          ;Ready Print flag
         =  ClrSnk  Equ     0BFH         ;clear Ready to Print Bit
         =  AtSpdF  Equ     20H          ;Stpr Mtr at constant speed
         =  NAtSpd  Equ     0DFH         ;Stpr Mtr Not at speed
         =  ADIntD  Equ     10H          ;Accel/Decel Init Done
         =  ADIntN  Equ     0EFH         ;Accel/Decel Init Not Done
         =
         =  FsCTm   Equ     08H          ;FailSafe/Constant Time
         =  ClrFsC  Equ     0F7H         ;Clear FailSafe/Const time flag
         =  FrmFd   Equ     04H          ;do formfeed
         =  LineFd  Equ     0FBH         ;do line feed
         =  DoNotP  Equ     02H          ;set Do Not Print Stat bit
         =  OkPrnt  Equ     0FDH         ;Reset - Ok to Print
         =  Ready   Equ     01H          ;Ready drive Stpr Mtr
         =  NotRdy  Equ     0FEH         ;Not Ready exit Stpr Mtr drive
         =
         =
         = ; PG
         = ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         = ;      Register allocation    (cont)
         = ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         =
         =
         = ;      Register Bank 1
         = ; ---------------------------------------------------------
0000     =  TmpR10  Equ     R0
0001     =  CAdrR1  EQU     R1           ;char data memory addr register
0002     =  ChStR1  EQU     R2           ;char processing status byte register
0003     =  CDtCR1  EQU     R3           ;Char Dot count register
0004     =  CDotR1  Equ     R4           ;Char dot temp storage register
0005     =  CCntR1  Equ     R5           ;Char count temp register
0006     =  StrCR1  EQU     R6           ;Store Char Register
         =
0007     =  OpnR71  EQU     R7           ;Available
         =
         = ;        Register Bank 1 Data Memory Address
         = ; ---------------------------------------------------------
```

230795-31

```
0018          TmpA10  Equ     24              ;temporary/scratch register
0019          ChARR1  EQU     25              ;char data memory addr register
001A          ChStAd  Equ     26              ;RB1 Char Status Reg address
001B          CDtCA1  EQU     27              ;Char Dot count register
001C          CDotA1  Equ     28              ;Char dot temp storage register
001D          CCntA1  Equ     29              ;Char count temp register
001E          StrCA1  EQU     30              ;Store Char Register

001F          OpnA71  EQU     31              ;Available


              ; PG
              ; ------------------------------------------------------------------------
              ;       RB1  Status Byte Bit Definition
              ; ------------------------------------------------------------------------

              ;       Bit               Definition
              ;       -----             --------------------------------------
              ;       7                 Stpr Mtr Direction:  L-to-R = 1, R-to-L = 0
              ;       6                 Char Init, 1 =  Done / 0 = Not Done
              ;       5                 Char Lookup Table Page: 1 = Pg1, 0 = Pg2
              ;       4                 1 = Test / 0 = Normal char print/input

              ;       3                 1 = EOF / 0 = Not EOF
              ;       2                 Full = 1/Not Full = 0, Line in Char Buffer
              ;       1                 1 = CR/(LF) / 0 = Not CR/(LF)
              ;       0                 1 = Init / 0 = Do Not Init, CB registers done

              ;       Bit Masks:        RB1
              ;       Character printing bit masks function on ChStR1
              ; ----------------------------------------------------------------
0080          ChrPrn  Equ     80H             ;Stpr Mtr Direction:  L-to-R = 1
007F          ClrCPr  Equ     7FH             ;Stpr Mtr Direction:  R-to-L = 0
0040          ChIntD  Equ     040H            ;Set Char Init Done
00BF          CIntND  Equ     0BFH            ;Reset Char Init Not Done
0020          ChOnP1  Equ     20H             ;Page 1 char, set rentry bit (ORL)
00DF          ChOnP2  Equ     0DFH            ;Page 2 char, reset rentry bit (ANL)
0010          TstPrn  Equ     10H             ;Char print test
00EF          NrmPrn  Equ     0EFH            ;Normal char input

0008          EOF     Equ     08H             ;set EOF Flag
00F7          ClrEOF  Equ     0F7H            ;clear EOF flag - Not EOF
0004          CRLF    Equ     04H             ;CR/LF
00FB          ClrCR   Equ     0FBH            ;Clear CR/LF
0002          CBFLn   Equ     02H             ;Full Line in Char Buffer
00FD          NCBFLn  Equ     0FDH            ;Not Full Line in Char Buffer
0001          IntCBR  Equ     01H             ;Init of CB registers done
00FE          ClICBR  Equ     0FEH            ;Init of CB registers not done


              ; PG
              ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
              ;                   Equates  (cont)
              ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

              ;       Misc
              ; ----------------------------------------------------------------
0004          RLPShf  Equ     04H             ;R-to-L print lookup table addr shift

0020          Ascii   Equ     20H             ;hex nmbr of first Ascii Char
007F          AscLst  Equ     7FH             ;hex nmbr of last Ascii Char

00F3          CRCpl   Equ     0F3H            ;ASCII control code 2's complement
00F6          LFCpl   Equ     0F6H            ; "
00F4          FFCpl   Equ     0F4H            ; "
00E5          EscCpl  Equ     0E5H            ; "
00E0          AscCpl  Equ     0E0H            ; "
00C8          FTCpl   Equ     0C8H            ; "
000D          CR      Equ     0DH             ;Ascii code (hex)
0020          Space   Equ     20H             ;Ascii code (hex)

0081          LAsEnd  Equ     81H             ;Ascii End 2's cpl - test line start
0082          PAsEnd  Equ     82H             ;Ascii End 2's cpl - within line print
007F          AscStp  Equ     7FH             ;Ascii mask, strip off MSB
0042          PgLnCt  Equ     66              ;Page Line Count: Default = 66
00C4          PgLCpl  Equ     0C4H            ;Printed lines per page test
001B          EOFCpl  Equ     1BH             ;EOF ascii code cpl

              ;       Loop count values
              ; ----------------------------------------------------------------
```

230795-32



```
0006             = 295 NDtCCt  Equ     06H              ;Normal Dot Column Count
000A             = 296 EDtCCt  Equ     0AH              ;Expanded Dot Column Count
0004             = 297 PHCnt1  EQU     04H              ;NUMBER OF SM PHASES ON INIT
                 = 298
0004             = 299 ILFCnt  Equ     04               ;Init LF step/phz count
0024             = 300 LPI6p6  Equ     36               ;Lines Per Inch 6.6
001B             = 301 LPI8p8  Equ     27               ;Lines Per Inch 8.8
0018             = 302 LPI10   Equ     24               ;Lines Per Inch 10
                 = 303
0001             = 304 LineCt  Equ     01               ;linefeed count
0042             = 305 FmFdCt  Equ     66               ;lines per formfeed count
0003             = 306 Status  EQU     03H              ;SEE BELOW FOR STATUS BYTE DEF.
                 = 307                                  ;  TEST:  SET FOR CR STPR MTR CONTROL
                 = 308
                 = 309
                 = 310 ; PG

                 = 311 ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                 = 312 ;       TIMER VALUES - UPI Timer/Counter is UP Counter
                 = 313 ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                 = 314 ;       12 MHz Clk timings
                 = 315 ; ------------------------------------------------------------
0080             = 316 DLYCL   EQU     80H              ;DELAY COUNT Long
0030             = 317 DLYCS   EQU     30H              ;DELAY COUNT Short
00CC             = 318 DlyTim  EQU     256-52           ;TIME DELAY constant ~2.0mS
0000             = 319 FailTm  EQU     256-256          ;FailSafe TIME = ~17.0mS
00CC             = 320 CrTmr1  EQU     256-52           ;CR Stpr Mtr Phase TIME = ~2.08mS
00BA             = 321 CrTmr2  EQU     256-70           ;CR Stpr Mtr Phase TIME = ~2.40mS
0092             = 322 CrTmr3  EQU     256-110          ;CR Stpr Mtr Phase TIME = ~4.16mS
00C0             = 323 IntTm2  EQU     256-64           ;Init Stpr Mtr Phase TIME = ~2.40mS
0098             = 324 LFTMR1  EQU     256-104          ;LF Stpr Mtr Phase TIME = ~4.16mS
                 = 325
                 = 326 ;       I/O port bit masks
00DF             = 327 NotBsy  Equ     0DFH             ;Not Busy
0020             = 328 Busy    Equ     20H              ;Busy
00EF             = 329 Ack     Equ     0EFH             ;Ack
0010             = 330 ReSAck  Equ     10H              ;ReSet Ack
                 = 331
                 = 332 ;       Misc bit Masks
000C             = 333 StrpLF  Equ     0CH              ;Strip off all bits but LF Stpr Mtr
0003             = 334 StrpCR  Equ     03H              ;Strip off all bits but CR Stpr Mtr
                 = 335
                 = 336 ;       Print Head fires on low going edge of Trigger
                 = 337 ;         bit #9 in dot column is masked off, always:  P2, bit 6
0040             = 338 PTRGLO  EQU     40H              ;PH TRIGGER BIT - LOW
00C0             = 339 PTRGHI  EQU     0C0H             ;PH TRIGGER BIT - HIGH
                 = 340
                 = 341 ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                 = 342 ;       Stepper Motor Phase State Equates
                 = 343 ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                 = 344
                 = 345 ;       Stepper Motor Phase Shift Index Offset Offset
0000             = 346 FStCRP  EQU     00H              ;F CR stpr mtr phase data start addr
0003             = 347 RStCRP  EQU     03H              ;R CR stpr mtr phase data start addr
0008             = 348 STLFF   Equ     08H              ;Paper feed stpr mtr phase data start addr
                 = 349
                 = 350 ;       CARRIAGE STEPPER MOTOR PHASE EQUATES
                 = 351 ;       Forward (1 thru 4)  &  Reverse (4 thru 1) :
0001             = 352 CRMFP1  EQU     01B              ;CR STPR MTR PHASE 1
0003             = 353 CRMFP2  EQU     11B              ;CR STPR MTR PHASE 2
0002             = 354 CRMFP3  EQU     10B              ;CR STPR MTR PHASE 3
0000             = 355 CRMFP4  EQU     00B              ;CR STPR MTR PHASE 4
                 = 356
                 = 357         ;LINE FEED STEPPER MOTOR PHASE EQUATES
                 = 358 ;       Forward:
0004             = 359 LFMFP1  EQU     0100B            ;LF STPR MTR PHASE 1
000C             = 360 LFMFP2  EQU     1100B            ;LF STPR MTR PHASE 2
0008             = 361 LFMFP3  EQU     1000B            ;LF STPR MTR PHASE 3
0000             = 362 LFMFP4  EQU     0000B            ;LF STPR MTR PHASE 4
                 = 363
                 = 364 ; PG
```

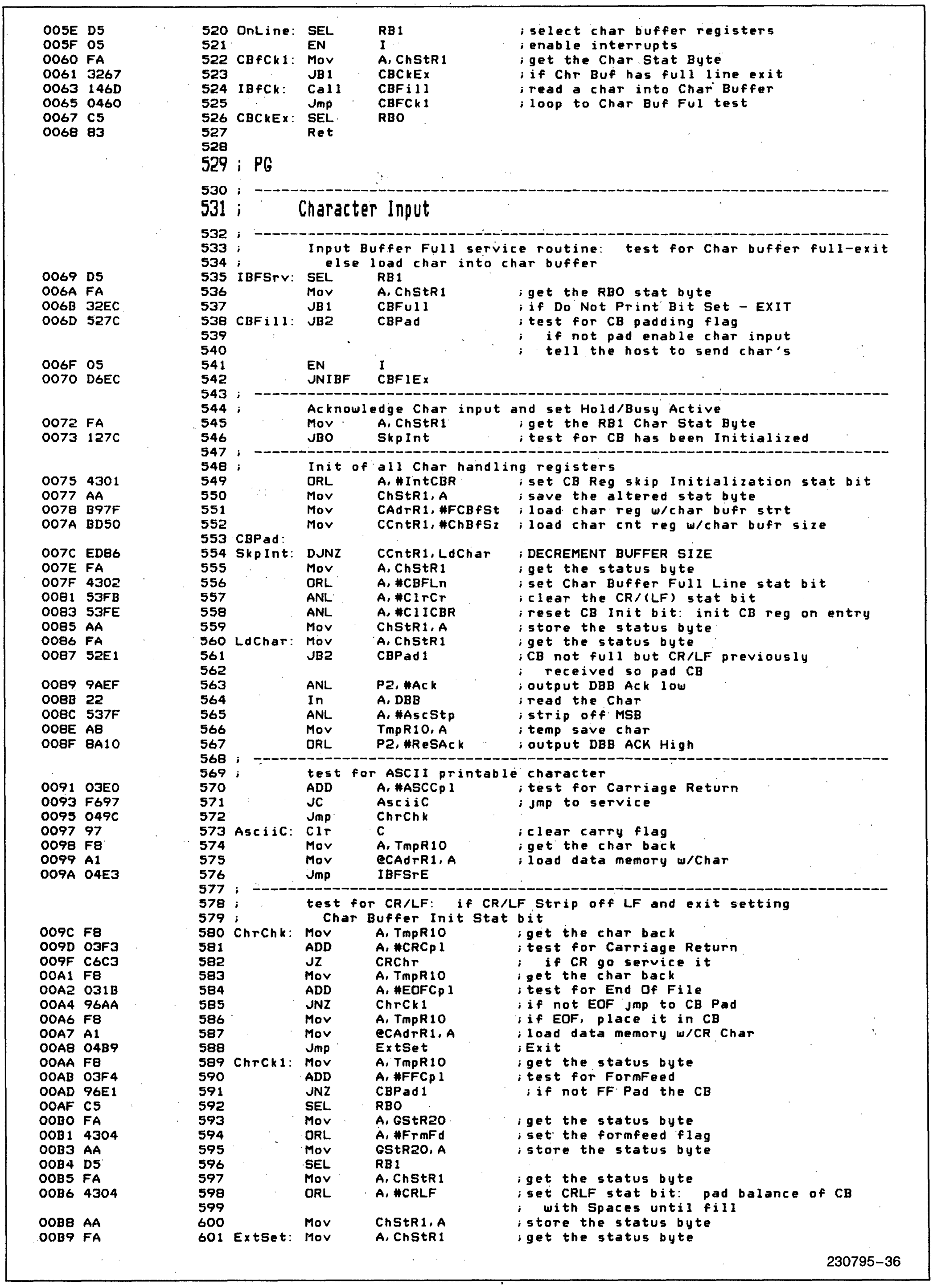
230795-33

```
           ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
           ;        STEPPER MOTOR SELECT & CONTROL [CURRENT LIMITING]
           ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

           ;        PORT BIT ASSIGNMENT:

           ;                \ \ \
           ;                S S S -
           ;                L C C
           ;                F R R
           ;                  8 1
           ;                  0 3
           ;                    2
           ;        ------------------
           ;                5 5 5 5
           ;                3 2 1 0
           ;        ------------------
           ; CODING:
           ;        SLF     0 1 1 0         06H
           ;        SCR80   1 0 0 0         0AH
           ;        SCR132  1 1 0 0         0CH
           ;        SMOFF   1 1 1 0         0EH
           ;                W/SCR80 & SCR132 '0' [BOTH SELECTED]
           ;                  DEFAULT IS TO 80 COL.
           ;                  [DO NOT KNOW WHETHER SCR80='0' WILL
           ;                  SELECT 80 COL ONLY] - REQUIRES TEST.

0008       SCR80    EQU     08H             ;SELECT CR STPR MTR - 80 COL
                                            ;  w/LF STPR MTR OFF
000C       SCR132   EQU     0CH             ;SELECT CR STPR MTR - 132 COL
                                            ;  w/LF STPR MTR OFF
0006       SLF      EQU     06H             ;SELECT LF STPR MTR ON
                                            ;  w/CR STPR MTR OFF
000E       SMOFF    EQU     0EH             ;SELECT CR & LF STPR MTR OFF


           ; PG
           ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
           ;                MAIN PROGRAM BODY
           ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

           ;        Power On / Reset Program Entry

           ;        PROGRAM START

0000                Org     00H

0000 040B  START:   JMP     RESET

           ;        INPUT BUFFER FULL INTERRUPT CALL ENTRY AND VECTOR
0003                ORG     03H
0003 1425  IBFIV:   Call    IBFIS
0005 93             RETR
           ;        TIMER OVERFLOW INTERRUPT CALL ENTRY AND VECTOR
0007                ORG     07H
0007 1429  TMRIV:   Call    TMRIS
0009 C5             SEL     RB0
000A 83             Ret

           ;                INITIALIZATION

000B 15    ReseT:   Dis     I
000C 35             Dis     TCntI
000D B40F           Call    InitAl          ;set all critical outputs inactive
000F B42F           Call    ClrDM           ;clear all data memory - 93H to 7FH
                                            ;  do not clear RB0, RB1 or Stack
0011 B44B           Call    InitCR          ;CALL CR SM POWER ON INIT
0013 9400           Call    InitLF          ;CALL LF SM POWER ON INIT

           ;                MAIN PROGRAM LOOP
           ;                All program segments are called from here

0015 B422  Home:    Call    CRHome          ;Call Home CR routine -
                                            ;  fixes logical and physical CR Home
0017 B400           Call    Defalt          ;set default register values
0019 142C  CBInpt:  Call    ESCBfF          ;Stat Switch / CB Input Service Test
                                            ; test for:  CB full/fill, LF, FF,
                                            ;            Char Prnt Test
```

230795-34

```
001B 3400   Repeat: Call   SMDriv        ;Call Forward Stpr Mtr Drive
001D 940D           Call   LFDriv        ;Call Linefeed Stpr Mtr Drive
001F D5             SEL    RB1
0020 FA             Mov    A,ChStR1      ;get the Char Status Register RB1
0021 7215           JB3    Home          ;jump to CR SM Home if EOF bit set
0023 0419           Jmp    CBInpt        ;loop to Char Buffer Input test

            ; PG
            ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
            ;               Interrupt Service Routine
            ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

            ; ----------------------------------------------------------------------------
            ;               Input Buffer Full Interrupt Service Routine
            ; ----------------------------------------------------------------------------

            IBFIS:
            ; ----------------------------------------------------------------------------
            ;       Acknowledge Char input and set Hold/Busy Active
0025 8A20           ORL    P2,#Busy      ;get & set DBB ACK/Busy Bits
0027 15             Dis    I             ;disable IBF interrupts
0028 83             Ret

            ; ----------------------------------------------------------------------------
            ;               Timer / Counter Interrupt Service Routine
            ; ----------------------------------------------------------------------------
            ;       ITF interrupt service routine disables all intr during
            ;         stpr mtr phase shifting
0029 15     TMRIS:  Dis    I             ;disable IBF interrupts
002A 35             Dis    TCntI         ;dicable ITF interrupts
002B 83             Ret

            ; PG
            ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
            ;       External Status Switch Check/Char Buffer Fill
            ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
            ESCBfF:                          ;Prep for normal character handling/input
002C D5             SEL    RB1
002D FA             Mov    A,ChStR1      ;get the character stat reg byte
002E 53EF           ANL    A,#NrmPrn     ;set normal character input
0030 AA             Mov    ChStR1,A      ;store the stat byte
0031 C5             SEL    RB0

            ;       Test External Status Port
0032 0F             MovD   A,P7          ;get the stat switch port bits
0033 123D           JB0    FormFd        ;  service Formfeed
0035 3245           JB1    LinFd         ;  service Linefeed
0037 5249           JB2    ChrTst        ;  service Character TEST
0039 725E           JB3    OnLine        ;  service Char Buffer Check/Fill
003B 042C           Jmp    ESCBfF        ;Loop
            ; ----------------------------------------------------------------------------
003D FA     FormFd: Mov    A,GStR20      ;get the status byte
003E 4304           ORL    A,#FrmFd      ;set the formfeed stat flag
0040 AA             Mov    GStR20,A      ;store trhe status byte
0041 940D           Call   LfDriv        ;do a formfeed
0043 042C           Jmp    ESCBfF
            ; ----------------------------------------------------------------------------
0045 940D   LinFd:  Call   LfDriv         ;do a line drive
0047 042C           Jmp    ESCBfF
            ; ----------------------------------------------------------------------------
0049 D5     ChrTst: SEL    RB1
004A FA             Mov    A,ChStR1      ;get the character stat reg byte
004B 4310           ORL    A,#TstPrn     ;set character test flag
004D AA             Mov    ChStR1,A      ;store the stat byte
004E B823           Mov    TmpR10,#PTAscS ;load the psuedo Ascii code tmp reg addr
0050 F0             Mov    A,@TmpR10     ;get the inc'd ascii code
0051 0381           ADD    A,#LAsEnd     ;test for code end
0053 9657           JNZ    AscCLd        ;if not code end jmp to load
                                         ;if end restart ascii at begining
0055 B020           Mov    @TmpR10,#Ascii ;store the ascii code start
0057 F0     AscCLd: Mov    A,@TmpR10     ;get the ascii code again
0058 AF             Mov    OpnR71,A      ;place in the empty register
0059 10             Inc    @TmpR10       ;Inc start ASCII char in data memory
005A B439           Call   PrnTst        ;call the DM load procedure
005C C5             SEL    RB0           ;reselect reg bank 0
005D 83             Ret
            ; ----------------------------------------------------------------------------
```

230795-35



```
005E D5     OnLine:  SEL    RB1              ;select char buffer registers
005F 05              EN     I                ;enable interrupts
0060 FA     CBfCk1:  Mov    A,ChStR1         ;get the Char Stat Byte
0061 3267            JB1    CBCkEx           ;if Chr Buf has full line exit
0063 146D   IBfCk:   Call   CBFill           ;read a char into Char Buffer
0065 0460            Jmp    CBFCk1           ;loop to Char Buf Ful test
0067 C5     CBCkEx:  SEL    RB0
0068 83              Ret

            ; PG
            ; ---------------------------------------------------------------------------
            ;       Character Input
            ; ---------------------------------------------------------------------------
            ;        Input Buffer Full service routine:  test for Char buffer full-exit
            ;          else load char into char buffer
0069 D5     IBFSrv:  SEL    RB1
006A FA              Mov    A,ChStR1         ;get the RB0 stat byte
006B 32EC            JB1    CBFull           ;if Do Not Print Bit Set - EXIT
006D 527C   CBFill:  JB2    CBPad            ;test for CB padding flag
                                             ;  if not pad enable char input
                                             ;  tell the host to send char's
006F 05              EN     I
0070 D6EC            JNIBF  CBF1Ex
            ; ---------------------------------------------------------------------------
            ;        Acknowledge Char input and set Hold/Busy Active
0072 FA              Mov    A,ChStR1         ;get the RB1 Char Stat Byte
0073 127C            JB0    SkpInt           ;test for CB has been Initialized
            ; ---------------------------------------------------------------------------
            ;        Init of all Char handling registers
0075 4301            ORL    A,#IntCBR        ;set CB Reg skip Initialization stat bit
0077 AA              Mov    ChStR1,A         ;save the altered stat byte
0078 B97F            Mov    CAdrR1,#FCBfSt   ;load char reg w/char bufr strt
007A BD50            Mov    CCntR1,#ChBfSz   ;load char cnt reg w/char bufr size
            CBPad:
007C ED86   SkpInt:  DJNZ   CCntR1,LdChar    ;DECREMENT BUFFER SIZE
007E FA              Mov    A,ChStR1         ;get the status byte
007F 4302            ORL    A,#CBFLn         ;set Char Buffer Full Line stat bit
0081 53FB            ANL    A,#ClrCr         ;clear the CR/(LF) stat bit
0083 53FE            ANL    A,#ClICBR        ;reset CB Init bit: init CB reg on entry
0085 AA              Mov    ChStR1,A         ;store the status byte
0086 FA     LdChar:  Mov    A,ChStR1         ;get the status byte
0087 52E1            JB2    CBPad1           ;CB not full but CR/LF previously
                                             ;  received so pad CB
0089 9AEF            ANL    P2,#Ack          ;output DBB Ack low
008B 22              In     A,DBB            ;read the Char
008C 537F            ANL    A,#AscStp        ;strip off MSB
008E A8              Mov    TmpR10,A         ;temp save char
008F 8A10            ORL    P2,#ReSAck       ;output DBB ACK High
            ; ---------------------------------------------------------------------------
            ;        test for ASCII printable character
0091 03E0            ADD    A,#ASCCpl        ;test for Carriage Return
0093 F697            JC     AsciiC           ; jmp to service
0095 049C            Jmp    ChrChk
0097 97     AsciiC:  Clr    C                ;clear carry flag
0098 F8              Mov    A,TmpR10         ;get the char back
0099 A1              Mov    @CAdrR1,A        ;load data memory w/Char
009A 04E3            Jmp    IBFSrE
            ; ---------------------------------------------------------------------------
            ;        test for CR/LF:  if CR/LF Strip off LF and exit setting
            ;          Char Buffer Init Stat bit
009C F8     ChrChk:  Mov    A,TmpR10         ;get the char back
009D 03F3            ADD    A,#CRCpl         ;test for Carriage Return
009F C6C3            JZ     CRChr            ;  if CR go service it
00A1 F8              Mov    A,TmpR10         ;get the char back
00A2 031B            ADD    A,#EOFCpl        ;test for End Of File
00A4 96AA            JNZ    ChrCk1           ;if not EOF jmp to CB Pad
00A6 F8              Mov    A,TmpR10         ;if EOF, place it in CB
00A7 A1              Mov    @CAdrR1,A        ;load data memory w/CR Char
00A8 04B9            Jmp    ExtSet           ;Exit
00AA F8     ChrCk1:  Mov    A,TmpR10         ;get the status byte
00AB 03F4            ADD    A,#FFCpl         ;test for FormFeed
00AD 96E1            JNZ    CBPad1           ;if not FF Pad the CB
00AF C5              SEL    RB0
00B0 FA              Mov    A,GStR20         ;get the status byte
00B1 4304            ORL    A,#FrmFd         ;set the formfeed flag
00B3 AA              Mov    GStR20,A         ;store the status byte
00B4 D5              SEL    RB1
00B5 FA              Mov    A,ChStR1         ;get the status byte
00B6 4304            ORL    A,#CRLF          ;set CRLF stat bit:  pad balance of CB
                                             ;  with Spaces until fill
00B8 AA              Mov    ChStR1,A         ;store the status byte
00B9 FA     ExtSet:  Mov    A,ChStR1         ;get the status byte
```

230795-36

```
00BA 4302                 ORL     A,#CBFLn        ;set Char Buffer Full Line stat bit
00BC 53FB                 ANL     A,#ClrCr        ;clear the CR/(LF) stat bit
00BE 53FE                 ANL     A,#ClICBR       ;reset CB Init bit: init CB reg on entry
00C0 AA                   Mov     ChStR1,A        ;store the status byte
00C1 04EC                 Jmp     CBFlEx          ;Exit
                ; ----------------------------------------------------------------
                ;         Store CR char read in LF char (assume its always there) and ignor it
00C3 F8         CRChr:    Mov     A,TmpR10        ;get the char back
00C4 A1                   Mov     @CAdrR1,A       ;load data memory w/CR Char
00C5 C5                   SEL     RB0
00C6 1E                   INC     LnCtR0          ;inc the line count
00C7 FE                   Mov     A,LnCtR0        ;get the line count
00C8 03C4                 Add     A,#PgLCpl       ;test for page feed ln cnt
                                                  ;  if LnCt => PgLnCt set formfeed flag
00CA E6D0                 JNC     NoFmFd          ;if not at end of page skip
00CC FA                   Mov     A,GStR20        ;get the status byte
00CD 4304                 ORL     A,#FrmFd        ;set the form feed status flag
00CF AA                   Mov     GStR20,A        ;save the status byte
00D0 D5         NoFmFd:   SEL     RB1
00D1 05                   En      I               ;enable the IBF service
00D2 9ADF                 ANL     P2,#NotBsy      ;output a not busy to Host
00D4 D6D4       LFTest:   JNIBF   LFTest          ;loop to next char
00D6 9AEF                 ANL     P2,#Ack         ;output DBB Ack low
00D8 22                   In      A,DBB           ;get next Char - assume it's a LF
                                                  ;  and ignor it (LF is forced upon
                                                  ;  detection of CR at print time)
00D9 FA         SetPad:   Mov     A,ChStR1        ;get the status byte
00DA 4304                 ORL     A,#CRLF         ;set CRLF stat bit:  pad balance of CB
                                                  ;  with Spaces until fill
00DC AA                   Mov     ChStR1,A        ;store the status byte
00DD 8A10                 ORL     P2,#ReSAck      ;output DBB ACK High
00DF 04E3                 Jmp     IBfSrE          ;jmp to addr step & exit
                ; ----------------------------------------------------------------
                ;         fill Char Buffer with space
00E1 B120       CBPad1:   Mov     @CAdrR1,#Space  ;load data memory w/Char
                ; ----------------------------------------------------------------
                ;         step the char address test for CB full &/or pad
00E3 C9         IBFSrE:   DEC     CAdrR1          ;Decrement dat memory location
00E4 FA                   Mov     A,ChStR1        ;get the status byte
00E5 32EC                 JB1     CBFull          ;test for CB Full
00E7 52EC                 JB2     CBFlEx          ;test for CB pad - exit w/Busy set
                ; ----------------------------------------------------------------
                ;         Set Busy Line Low - Not Busy
00E9 05                   EN      I
00EA 9ADF                 ANL     P2,#NotBsy      ;output a not busy to Host
                ; ----------------------------------------------------------------
                ;         exit w/ Busy Still set high
                CBFull:
00EC 83         CBFlEx:   Ret

                ; PG

                ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                ;         L-to-R/R-to-L Carriage Stepper Motor Drive
                ;              and Line Printing
                ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

0100                      ORG     100H

0100 3622       SMDriv:   JTO     RAccel          ;if Print Head at left drive right
                                                  ;  else drive left
                ; F=============================================================
                FAccel:                           ;L-to-R Accelerate Stepper Motor
                ;         Set the Forward acceleration/drive Entry status bits
0102 FA                   Mov     A,GStR20        ;get the status byte
0103 53BF                 ANL     A,#ClrSnk       ;set not at speed flag = 0
0105 53DF                 ANL     A,#NAtSpd       ;set Not At Speed flag = 0
0107 4380                 ORL     A,#LRPrnt       ;set L-to-R prnt stat bit = 1
0109 4301                 ORL     A,#Ready        ;set stpr mtr ready - Drive On
010B 53EF                 ANL     A,#ADIntN       ;set A/D Init Not Done
010D AA                   Mov     GStR20,A        ;store the status byte
010E D5         CBRDir:   SEL     RB1
010F FA                   Mov     A,ChStR1        ;get the Char Stat Reg Data Mem Addr
0110 4380                 ORL     A,#LRPrnt       ;Set L-to-R print bit
0112 AA                   Mov     ChStR1,A        ;save the Char Stat byte
0113 C5                   SEL     RB0

                ;         Restore the phase register index addresses
0114 B821                 Mov     TmpR00,#CPSAdr  ;get Phz Storage Addr psuedo reg
0116 F0                   Mov     A,@TmpR00       ;  get stored CR last phase index addr
0117 AB                   Mov     PhzR30,A        ;place last LF phase index addr in Phz Reg

```

230795-37

```
           ;        Set up for next phase bit output before entering timing loops
0118 1B             INC     PhzR30          ;STEP PHASE DB ADDRESS
0119 FB             MOV     A,PhzR30        ;CHECK THE PHASE COUNT REG
011A 521E           JB2     IAFZrP          ;CHK FOR COUNT BIT ROLLOVER
011C 2440           JMP     SMDflt          ;skip adr index reset
011E BB00  IAFZrP:  MOV     PhzR30,#FStCRP  ;ZERO CR SM PHASE REGISTER
0120 2440           Jmp     SMDflt

           ; R=============================================================
           RAccel:                  ;R-to-L Accelerate Stepper Motor
           ; --------------------------------------------------------------
           ;        Set the Reverse acceleration/drive Entry status bits
0122 FA             Mov     A,GStR20        ;get the status byte
0123 53BF           ANL     A,#ClrSnk        ;clear Print Ready bit
0125 53DF           ANL     A,#NAtSpd       ;set Not At Speed flag = 0
0127 537F           ANL     A,#RLPrnt       ;set R-to-L prnt status bit
0129 4301           ORL     A,#Ready        ;set stpr mtr ready - Drive On
012B 53EF           ANL     A,#ADIntN       ;set A/D Init Not Done
012D AA             Mov     GStR20,A        ;store the status byte
012E D5    RCBRDr:  SEL     RB1
012F FA             Mov     A,ChStR1        ;get the Char Stat Reg Data Mem Addr
0130 537F           ANL     A,#RLPrnt       ;Set R-to-L print bit
0132 AA             Mov     ChStR1,A        ;save the Char Stat byte .
0133 C5             SEL     RB0
           ; --------------------------------------------------------------
           ;        Restore the  phase register index address
0134 B821           Mov     TmpR00,#CPSAdr   ;get Phz Storage Addr psuedo reg
0136 F0             Mov     A,@TmpR00       ;  get stored CR last phase index addr
0137 AB             Mov     PhzR30,A        ;place last LF phase index addr in Phz Reg
           ;        Set up for next phase bit output before entering timing loops
0138 CB             Dec     PhzR30          ;STEP PHASE DB ADDRESS
0139 FB             MOV     A,PhzR30        ;CHECK THE PHASE COUNT REG
013A 523E           JB2     IARZrP          ;CHK FOR COUNT BIT ROLLOVER
013C 2440           JMP     SMDflt
013E BB03  IARZrP:  MOV     PhzR30,#RStCRP  ;ZERO CR SM PHASE REGISTER

           SMDflt:
           ; --------------------------------------------------------------
           ;        for stablization of unused stpr mtr during CR stpr mtr drive,
           ;          store the unused stpr mtr current phase bits
0140 B822           Mov     TmpR00,#LPSAdr   ;get the CR phz storeage addr
0142 F0             Mov     A,@TmpR00       ;get the byute stored there
0143 E3             MovP3   A,@A            ;get the phz data byte
0144 B820           Mov     TmpR00,#LastPh  ;load Last Phz psuedo reg to Temp Reg
0146 A0             Mov     @TmpR00,A       ;store Last Phase bits - inderect

           ;        SetUp Stpr Mtr Time Constant
0147 BDBA           MOV     TConR0,#CrTmr2  ;Load time constant Reg

           Select:                          ;Select the Stpr Mtr
0149 2308           MOV     A,#SCRB0        ;GET CR SM SELECT BITS
014B 3D             MOVD    P5,A            ;SELECT SM [SCRB0]
           ; ------------------------------------------------------------
           ;        SetUp Stpr Mtr Phase Shift index address register
           ;        Output next phase and init timer to Std Time constant
014C FD    STRTT:   MOV     A,TConR0        ;get time constant from reg
014D 62             MOV     T,A             ;load the timer
014E FB             MOV     A,PhzR30        ;get the phz reg indirect addr index
014F E3             MovP3   A,@A            ;do indirect get of phz bits

           ; ------------------------------------------------------------
           ;        patch together the CR last and LF next phase bits
0150 B820           Mov     TmpR00,#LastPh  ;load Last Phz psuedo reg to Temp Reg
0152 40             ORL     A,@TmpR00        ;patch together CR existing & new LF
0153 3C             MOVD    P4,A            ;OUTPUT BITS
0154 55             STRT    T               ;START TIMER

           ;        At start of timing loop do all Stpr Mtr Accel/Decel or
           ;          Character SetUp overhead
0155 740C           Call    ADPTst          ;call Accel/Decel/Print Test

           ;        Set up for next phase bit output before entering timing loops
           PNRdy1:  ;test for forward / reverse phase start indirect index to load
0157 FA             Mov     A,GStR20        ;store stat byte
0158 F264           JB7     AclF2

           ; reverse:
           ;        Set up for next phase bit output before entering timing loops
015A CB             Dec     PhzR30          ;STEP PHASE DB ADDRESS
015B FB             MOV     A,PhzR30        ;CHECK THE PHASE COUNT REG
015C 5260           JB2     ARZroP          ;CHK FOR COUNT BIT ROLLOVER
015E 2462           JMP     ARNxtP
0160 BB03  ARZroP:  MOV     PhzR30,#RStCRP  ;ZERO CR SM PHASE REGISTER
0162 246C  ARNxtP:  Jmp     ANxtPh
```

230795–38

```

                     ; forward:
                     ;         Set up for next phase bit output before entering timing loops
0164 1B              AclF2:    INC     PhzR30           ;STEP PHASE DB ADDRESS
0165 FB                        MOV     A,PhzR30         ;CHECK THE PHASE COUNT REG
0166 526A                      JB2     AFZroP           ;CHK FOR COUNT BIT ROLLOVER
0168 246C                      JMP     ANxtPh           ;skip adr index reset
016A BB00            AFZroP:   MOV     PhzR30,#FStCRP   ;ZERO CR SM PHASE REGISTER
                     ANxtPh:
                     ; ----------------------------------------------------------------
                     ;         stage one timer loop - T occurs before Std timeout
                     ;           wait for time out
016C 1682            TLOOP2:   JTF     FAILSF           ;JMP ON TIME OUT-t DOES NOT OCCUR 1ST
016E 5672                      JT1     tCHK1            ;IS T HIGH-JMP TO tCHK
0170 246C                      JMP     TLOOP2           ;LOOP FOR JT1 OR JTF
0172 00              tCHK1:    NOP                      ;delay, then double check T signal
0173 5677                      JT1     tTruW1           ;JUMP T TEST TRUE-WAIT FOR JTF
0175 246C                      JMP     TLOOP2
                     tTruW1:
                     ;         test for Print Ready bit - was Print Head Fire Setup Done?
                     ;         insert acceleration time/store time count done/notdone flag bit
0177 FA                        Mov     A,GStR20         ;get the status byte - prep for prnt
0178 D27C                      JB6     RdyPr2           ;if Ready Print bit set call PHFire
017A 247E                      Jmp     SkpPHF           ;  else skip Print Head Fire
017C 74CA            RdyPr2:   Call    PHFire           ;print head solenoid fire routine
                     PNRdy2:
                     SkpPHF:
017E 1698            tTruW2:   JTF     NXTPHZ           ;JUMP TO SM ERROR
0180 247E                      JMP     tTruW2           ; LOOP TO TLOOP3
                     ; ----------------------------------------------------------------
                     ;         Step into failsafe/startup timer setup - T does not
                     ;           occurs before Std Time timeout, load failsafe SM protection
                     ;           time and wait for failsafe timeout or T.  If T occurs
                     ;           output phase immediately after T verify.
0182 2300            FAILSF:   MOV     A,#FailTm        ;LOAD TIMER W/~15.0mS
0184 62                        MOV     T,A              ;  SM PROTECTION TIMEOUT
0185 55                        STRT    T                ;START TIMER
                     ; ----------------------------------------------------------------
                     ;         set the Status bit for Store time test
0186 FA                        Mov     A,GStR20         ;get the status byte
0187 4308                      ORL     A,#FSCTm         ;set Failsafe/constant time flag
0189 AA                        Mov     GStR20,A         ;store the status byte
018A 5690            TLOOP3:   JT1     tCHK2            ;IS T HIGH
018C 16AC                      JTF     DSLECT           ;IF TIME OUT GO SM ERROR
018E 248A                      JMP     TLOOP3           ;LOOP UNTIL T HIGH OR T-OUT
0190 00              tCHK2:    NOP                      ;WAIT
0191 5695                      JT1     StrTm1           ;jump out and store elapsed time
0193 248A                      JMP     TLOOP3           ;  JMP TO FAILSF LOOP
0195 65              StrTm1:   Stop    TCnt             ;stop the failSafe Timer
0196 42                        Mov     A,T              ;read the timer
0197 A1                        MOV     @TStrR0,A        ;Store the time read in indexed addr
                                                        ; - next entry to A/D Memorize Time
                                                        ; routine will add time constant to it

                     ;    Test is CR Stpr Mtr Drive is finished prior to next phase output
                     ; -------------------------------------------------------------------
                     NXTPHZ:
                     ;         test for forward / reverse phase start indirect index to load
0198 FA                        Mov     A,GStR20         ;store stat byte
0199 F2A7                      JB7     FDrive
                     ; Reverse --  test for Reverse Stpr Mtr Drive procedure exit
                     ;         ALWAYS drive the CR to the left most HOME position
019B 26AC                      JNTO    EOLn             ;test if home position jmp stop
019D FA                        Mov     A,GStR20         ;get the status byte
019E 124C                      JBO     StrtT            ;test Ready stat bit:
                                                        ;  if bit 0 = 1 then Print More
01A0 4302                      ORL     A,#DoNotP        ;set the do not print flag
01A2 53BF                      ANL     A,#ClrSnk        ;clear Print Ready bit
01A4 AA                        Mov     GStR20,A         ;save the status byte
01A5 244C                      Jmp     StrtT            ;continue CR SM drive
                                                        ;  - only exit is HR
                     ; Forward  --  test for Forward Stpr Mtr Drive procedure exit
                     FDrive:
01A7 FA                        Mov     A,GStR20         ;get the status byte
01A8 124C                      JBO     StrtT            ;test Ready stat bit:
                                                        ;  if bit 0 = 1 then Print More
01AA 24AC                      Jmp     EOLn             ;  else jmp to End Of Line exit
                                                        ;jump to start timer again
                     DSLECT:
01AC 5437            EOLn:     Call    DeclSM           ;call Sptr Mtr Deceleration
                     ; -------------------------------------------------------------------
                     ;         test for forward / reverse phase start indirect index to load
01AE FA                        Mov     A,GStR20         ;store stat byte
01AF F2B3                      JB7     FDrvFS           ;jmp to f drive flag set
```

230795-39

```
01B1 53FD                    ANL     A,#OkPrnt        ;reset print flag - Ok Print
                                                      ; only if printing R-to-L

                   ;         update the status byte
01B3 53BF          FDrvFS:   ANL     A,#ClrSnk        ;clear Print Ready bit
                                                      ;set the Status bit for Store time test
01B5 53DF                    ANL     A,#NAtSpd        ;Clear At Print Speed Bit
01B7 AA                      Mov     GStR20,A         ;save the status byte
01B8 83                      RET

                   ; PG
                   ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                   ;         Stepper Motor Accel. Time Storeage
                   ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

0200                         ORG     200H
                                                      ;  Entry has Gen Stat Byte in A
0200 920C          ADMmTS:   JB4     DADInt           ;is A/D init done - then jmp

                   ;         1st Entry initializes the A/D Time store working registers
0202 B92F                    Mov     TStrR0,#SMBfSt   ;Load the Stpr Mtr Buffer Start Addr
0204 BC0B                    Mov     CntR40,#ASBfSz   ;Load the Buffer Size
0206 FA                      Mov     A,GStR20         ;get the status byte
0207 4310                    ORL     A,#ADIntD        ;set not 1st Accel Entry Flag
0209 AA                      Mov     GStR20,A         ;store the status byte
020A 4436                    Jmp     ADExit           ;exit - 1st entry has not generated
                                                      ;  a closed time window

                   ;         Step the A/D Store count
020C EC26          DADInt:   DJNZ    CntR40,StorCt    ;dec Times to store count
                                                      ;if not 0 store the count
                                                      ;else at end-set done flag
020E FA                      Mov     A,GStR20         ;get the status byte
020F 4320                    ORL     A,#AtSpdF        ;set at speed/no more to store flag
0211 AA                      Mov     GStR20,A         ;store the status byte

                   ;         Initialize Char Print Registers:  if printing enabled
0212 3226                    JB1     StorCt           ;if Do Not Print stat bit set
                                                      ;  Skip the Char register init

                   ;         Initialize all Char Reg's
                   ;         Test for L-to-R (forward) or R-to-L (reverse) printing
0214 D5                      SEL     RB1
0215 FA                      Mov     A,ChStR1         ;get the status byte
0216 4340                    ORL     A,#CHIntD        ;set Char Init Done flag - bypass
0218 AA                      Mov     ChStR1,A         ;save the status byte
0219 F21F                    JB7     LdCBR1           ;test Chr Stat Byte Returned
                                                      ;  if bit 7 = 1 then Print L-to-R
021B B92F          LdCBR:    Mov     CAdrR1,#RCBfIS   ;load char reg w/char bufr strt R-to-L
021D 4421                    Jmp     LdCBR2

021F B980          LdCBR1:   Mov     CAdrR1,#FCBfIS   ;load char reg w/char bufr strt L-to-R

0221 BD51          LdCBR2:   Mov     CCntR1,#ChBfIS   ;load char cnt reg w/char bufr size
0223 BB01                    Mov     CDtCR1,#01         ;set the chr dot column cnt
0225 C5                      SEL     RB0

                   ;         Test for t > Tc or t < Tc
0226 722C          StorCt:   JB3     FailST           ;test for failsafe time switch

                   ;         t < Tc = store Time Constant in use
0228 FD                      Mov     A,TConR0         ;Get time constant currently in use
0229 A1                      Mov     @TStrR0,A        ;Memorize/Store the time - indirect addr
022A 4435                    Jmp     ADPRet

                   ;         t > Tc = store Time Constant + FailSafe Time Elapsed
                   ;         [see Accel/Cnst Speed/Decel WaveForm]
                   ;         equation is:    Trd - FailSafe Time = Tx
                   ;                      => Trd + Cpl(FailSafe Time) = Tx
                   ;                         Tx + Tcnst = T
                   ;                         Store/Memorize T

022C F1            FailST:   Mov     A,@TStrR0        ;get the stored time
022D 03C8                    Add     A,#FTCpl         ;2's cpl add
022F 6D                      Add     A,TConR0         ;Add: Time stored + Time constant
                                                      ;  currently in use
0230 A1                      Mov     @TStrR0,A        ;Memorize/Store the time
                   ;         Reset the Status bit for Store time test

0231 FA                      Mov     A,GStR20         ;get the status byte
0232 53F7                    ANL     A,#ClrFSC        ;reset Failsafe/constant time flag
                                                      ;  assumes entry via constant time
```

230795-40

```
0234 AA                  Mov     GStR20,A        ;store the status byte
0235 C9        ADPRet:   Dec     TStrRO          ;step the A/D time data store addr
0236 83        ADExit:   Ret

               ; PG
               ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
               ;         Carriage Stepper Motor Deceleration
               ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

               DeclSM:
               ;         SetUp the Deceleration registers
0237 B925                Mov     TStrRO,#SMBEnd  ;Load the Stpr Mtr Buffer End Addr
0239 BC0A                Mov     CntR40,#DSBfSz  ;Load the Buffer Size
023B FB                  MOV     A,PhzR30        ;get phase index address
023C E3                  MovP3   A,@A            ;get phase from indexed address
               ;         patch together the CR last and LF next phase bits
023D B820                Mov     TmpROO,#LastPh  ;load Last Phz psuedo reg to Temp Reg
023F 40                  ORL     A,@TmpROO        ;patch together CR existing & new LF
0240 3C                  MOVD    P4,A            ;OUTPUT BITS
0241 F1        StrtTD:   MOV     A,@TStrRO       ;get time from indexed data memory
0242 62                  MOV     T,A             ;load timer
0243 55                  STRT    T               ;START TIMER
0244 19                  Inc     TStrRO          ;step the Memorized time addr index reg
               ;         test for forward / reverse phase start indirect index to load
0245 FA                  Mov     A,GStR20        ;store stat byte
0246 F252                JB7     DclF2

               ; reverse:
               ;         Set up for next phase bit output before entering timing loops
0248 CB                  Dec     PhzR30          ;decrement the phase addr
0249 FB                  MOV     A,PhzR30        ;Get the phz data addr
024A 524E                JB2     DRZroP          ;CHK FOR COUNT BIT ROLLOVER
024C 445A                JMP     DNxtPh
024E BB03      DRZroP:   MOV     PhzR30,#RStCRP  ;ZERO CR SM PHASE REGISTER
0250 445A                Jmp     DclR2

               ; forward:
               ;         Set up for next phase bit output before entering timing loops
0252 1B        DclF2:    Inc     PhzR30          ;increment the phase addr
0253 FB                  MOV     A,PhzR30        ;Get the phz data addr
0254 5258                JB2     DZroPh          ;CHK FOR COUNT BIT ROLLOVER
0256 445A                JMP     DNxtPh          ;skip adr index reset
0258 BB00      DZroPh:   MOV     PhzR30,#FStCRP  ;ZERO CR SM PHASE REGISTER
               DNxtPh:                           ;set up for next phase shift
025A FB        DclR2:    MOV     A,PhzR30        ;get phase index address
025B E3                  MovP3   A,@A            ;get phase from indexed address
               ;         patch together the CR last and LF next phase bits
025C B820                Mov     TmpROO,#LastPh  ;load Last Phz psuedo reg to Temp Reg
025E 40                  ORL     A,@TmpROO        ;patch together CR existing & new LF
025F 1663      TLoopD:   JTF     NxtPD2          ;JMP ON TIME OUT TO NEXT PH
0261 445F                JMP     TLoopD          ;LOOP UNTIL TIME OUT
0263 3C        NxtPD2:   MOVD    P4,A            ;OUTPUT BITS
0264 EC41                DJNZ    CntR40,StrtTD   ;Exit Test

               ;         Set Storeage of next phase data in psuedo addr. This insures
               ;         next phase is sequence correct for stpr mtr drive direction
0266 B821      SetRN:    Mov     TmpROO,#CPSAdr   ;get Phz Storeage Addr psuedo reg
0268 FB                  MOV     A,PhzR30        ;get Phz data
0269 A0                  Mov     @TmpROO,A       ;store CR Next phase index addr
026A B478      DMExit:   Call    DlyLng
026C B490                Call    DeS1SM
026E 83                  RET

               ; PG
               ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
               ;        Stepper Motor Phase Shift Definitions
               ;            All program procedures call this data.
               ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

0300                     ORG     300H

               ;         DEFINE PHASE ADDRESSES:
               ;         THE PHASE DATA IS ENCODED TO THE ADDRESS CALLED DURING THE
               ;         STPR MTR ENERGIZE SEQUENCE CORRESPONDING TO THE NEXT PHASE
               ;         OF THE SEQUENCE REQUIRED.

               ;         CARRAGE MOTOR ENCODING: FORWARD  -  LEFT-to-RIGHT
               ;                                 REVERSE  -  RIGHT-to-LEFT

```

230795-41

```
               ;       Reverse direction ENCODING is the same bytes accessed in
               ;         reverse direction

0300 01                DB      CRMFP1
0301 03                DB      CRMFP2
0302 02                DB      CRMFP3
0303 00                DB      CRMFP4

               ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

               ;       LF MOTOR PHASE ENCODE & DECODE: FORWARD (CLOCKWISE)
               ;       Forward direction ENCODING:
               ;       -------------------------------------------------

0308                   ORG     308H

0308 04                DB      LFMFP1
0309 0C                DB      LFMFP2
030A 08                DB      LFMFP3
030B 00                DB      LFMFP4


               ; PG
               ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
               ;       Accel/Decel / Character Handling Test
               ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
               ;       TEST  >  Is CR Stpr Mtr At Speed ??
               ;         Yes - SetUp do Character Processing
               ;         No  - Calculate / Store the Acceleration Phase Shift Time (11)
               ; ---------------------------------------------------------------------------

030C FA        ADPTst: Mov     A,GStR20        ;get the status byte
030D B211              JB5     PHFSet          ;test if Stpr Mtr At Speed
                                               ;  jmp to Prnt Head Fire Setup
030F 4400              Jmp     ADMmTS          ;else Call Accel/Decel Memory Time Store

               ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
               ;        Process Characters for Printing
               ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

               ;       Character dot matrix - normal char
               ;       d = Dot Column
               ;       b = Blank Column

               ;       b d d d d d
               ;       (Char Matrix)
               ;       0 0 0 0  b
               ;       0 0 0 1  d
               ;       0 0 1 0  d
               ;       0 0 1 1  d
               ;       0 1 0 0  d
               ;       0 1 0 1  d
               ; ---------------------------------------------------------------------------

0311 2668      PHFSet: JNTO    Retrn           ;if R=0 not ready to print-exit
0313 326A              JB1     NPRet           ;if Do Not Print stat bit set - EXIT
0315 D21B              JB6     SinkSt          ;if bit previously set-skip setting it
0317 FA                Mov     A,GStR20        ;get the status byte
0318 4340              ORL     A,#SnkSet       ;set Prnt Ready Sink bit
031A AA                Mov     GStR20,A        ;save the status byte
031B D5        SinkSt: SEL     RB1
031C FA                Mov     A,ChStR1        ;get char status register addr
031D D23A              JB6     PageCk          ;test Char Init Done, 1 = Print Dot
                                               ;   0 = Get Char

               ; PG
               ; --------------------------------------------------------------------------
               ;       Call for Individual character processing:  mid line test if CR/(LF)
               ; --------------------------------------------------------------------------
               GetChr:
               ;       test for CR/(LF) if it is the test position in the line
031F F1        CRChCk: Mov     A,@CAdrR1       ;get character
0320 03F3              ADD     A,#CRCpl        ;test for Carriage Return
0322 C626              JZ      CrLnCk          ;  if CR go service it
0324 6437              Jmp     AsciC1          ;if not CR Insert Space Char
0326 FA        CRLnCk: Mov     A,ChStR1        ;get char status register addr
0327 F22B              JB7     HlfLn           ;test Chr Stat Byte Returned
                                               ;  if bit 7 = 1 then Print L-to-R
0329 6432              Jmp     SpFill          ;if R-to-L print skip exit upon CR detect
               ; ------------------------------------------------------------------------
```

230795-42

```
                ;         if L-to-R printing exit the line if less than 1/2 line printed
032B FD         HlfLn:    Mov     A,CCntR1          ;load char cnt reg w/char bufr size
032C 03D9                 ADD     A,#HlfCpl         ;add the 2's cpl of 1/2 chr buf size
032E F632                 JC      LnPad             ;if CB>1/2 full set CR/LF stat bit for pad
                                                    ;If CB<1/2 set buffer full stat bit
0330 648A                 Jmp     MdLnEx            ;mid-line exit
                SpFill:
0332 97         LnPad:    Clr     C                 ;clear carry flag
0333 2320                 Mov     A,#Space          ;insert a space char
0335 6438                 Jmp     ChIsrt            ;char inserted jmp over get char
                ; -------------------------------------------------------------------
0337 F1         AsciC1:   Mov     A,@CAdrR1         ;get character
0338 7498       ChIsrt:   Call    GChar1            ;call the char lookup/trns table
                ; -------------------------------------------------------------
                ;         fetch the char dot column data
                PageCk:                             ;page test for balance of char
033A FA                   Mov     A,ChStR1          ;get the status byte
033B B241                 JB5     FxJmp1            ;fix jmp over page boundries
033D F4EB                 Call    ChrPg2            ;Ascii char 50 - 7F Hex
033F 6443                 Jmp     MtxTst            ;jump to Matrix Test
0341 D4F0       FxJmp1:   Call    ChrPg1            ;Ascii char 20 - 4F Hex
                                                    ;  fall thru to print matrix
                                                    ;  and CB count tests

                ; PG
                ; ----------------------------------------------------------------------
                ;         test the Char dot column print matrix count and Char buffer count
                ; ----------------------------------------------------------------------
0343 EB61       MtxTst:   DJNZ    CDtCR1,PrntDt     ;test for dot col or blank
                                                    ;status byte in A upon entry here
0345 FA                   Mov     A,ChStR1          ;get the status byte
0346 53BF                 ANL     A,#CIntND         ;set Char Init NotDone stat Flag
0348 AA                   Mov     ChStR1,A          ;store the status byte
0349 ED58                 DJNZ    CCntR1,NotLCh     ;dec char cnt-jmp if Not Last Char
034B 53FD                 ANL     A,#NCBFln         ;if 0 reset stat bit Not CB Full Line
034D 53FE                 ANL     A,#C1ICBR         ;reset CB Reg Init Flag - do Init
034F AA                   Mov     ChStR1,A          ;save the status byte

0350 C5                   SEL     RB0
0351 FA                   Mov     A,GStR20          ;get Gen Status register addr
0352 53FE                 ANL     A,#NotRdy         ;clear the ready bit
0354 AA                   Mov     GStR20,A          ;store the General Status Byte
0355 D5                   SEL     RB1
0356 6468                 Jmp     Retrn             ;EXit

                ;         Test for L-to-R (forward) or R-to-L (reverse) printing
                ;           (see GChar1 ASCII char code translation procedure)
                ; ----------------------------------------------------------------------
                NotLCh:                             ;A contains LR/RL bit properly set
0358 FA                   Mov     A,ChStR1          ;get char status register addr
0359 F25E                 JB7     StpCh2            ;test Chr Stat Byte Returned
                                                    ;  if bit 7 = 1 then Print L-to-R
035B 19         StpCh1:   Inc     CAdrR1            ;Increment char data memory addr.
035C 6468                 Jmp     Retrn
035E C9         StpCh2:   Dec     CAdrR1            ;Decrement char data memory addr.
035F 6468                 Jmp     Retrn             ;  fall thru to Get Char


                ; ----------------------------------------------------------------------
                ;         Re-Entry Exit point for same char:
                ;           (before returning step the matrix)
                ; ----------------------------------------------------------------------
                ;         Test for L-to-R (forward) or R-to-L (reverse) printing
                ;           (see GChar1 ASCII char code translation procedure)
                ; ----------------------------------------------------------------------

                PrntDt:
0361 FA         PrnDir:   Mov     A,ChStR1          ;get char status byte
0362 F267                 JB7     StpCD2            ;test Chr Stat Byte Returned
                                                    ;  if bit 7 = 1 then Print L-to-R
0364 CC         StpCD1:   Dec     CDotR1            ;reverse step char dot col index
                                                    ;  addr if R-to-L print
0365 6468                 Jmp     Retrn             ;skip over L-to-R print addr inc
0367 1C         StpCD2:   INC     CDotR1            ;forward step char dot col index
                                                    ;  addr if L-to-R print
                                                    ;EXIT

                ; PG
```

230795-43

```
             ; -------------------------------------------------------------------
             ;       Character Print SetUp Exit Procedures
             ; -------------------------------------------------------------------
             ;       Clean Standard Exit
             ; -------------------------------------------------------------------
0368 C5      Retrn:  SEL     RB0
0369 83              Ret                         ;EXIT - return w/ Reg Bank 0 Reset

             ;       Do Not Print exit:  set Stpr Mtr drive routine count loop
036A D5      NPRet:  SEL     RB1
036B FA              Mov     A,ChStR1            ;get the status byte
036C F27C            JB7     SkpNPI              ;test print direction
             ; Reverse
036E C5              SEL     RB0
036F FA              Mov     A,GStR20            ;get the status byte
0370 53BF            ANL     A,#ClrSnk           ;reset the print ready bit- skips PHFire call
0372 83              Ret
             ; Forward
0373 D27C            JB6     SkpNPI              ;test for first PHFSet entry reg init
             ;       Initialize register variables upon first entry
             ;         end of count clears char to print bit in status byte
0375 4340            ORL     A,#ChIntD           ;set Char Reg Init Done stat bit
0377 AA              Mov     ChStR1,A            ;save the status byte
0378 B807            Mov     TmpR10,#07H         ;load CR stpr mtr count during NoPrnt
037A 6488            Jmp     NPExit
037C E888    SkpNPI: DJNZ    TmpR10,NPExit
037E FA              Mov     A,ChStR1            ;get the status byte
037F 53BF            ANL     A,#CIntND           ;reset - char init not done
0381 AA              Mov     ChStR1,A            ;save the status byte
0382 C5              SEL     RB0
0383 FA              Mov     A,GStR20            ;get Gen Status register addr
0384 53FE            ANL     A,#NotRdy           ;clear the ready bit
0386 AA              Mov     GStR20,A            ;store the General Status Byte
0387 83      NSetEx: Ret
0388 C5      NPExit: SEL     RB0
0389 83              Ret

             ;       Mid-Line Exit
             ; -------------------------------------------------------------------
             ;       EXIT - if CR and not > 1/2 line done during L-to-R print
038A FA      MdLnEx: Mov     A,ChStR1            ;get the status byte
038B 53FD            ANL     A,#NCBFln           ;if 0 reset stat bit Not CB Full Line
038D 53FE            ANL     A,#ClICBR           ;reset CB Reg Init Flag - do Init
038F AA              Mov     ChStR1,A            ;save the status byte
0390 C5              SEL     RB0
0391 FA              Mov     A,GStR20            ;get the RB0 status byte
0392 4302            ORL     A,#DoNotP           ;set the Do Not Print Flag(for RAccel)
0394 53BF            ANL     A,#ClrSnk           ;reset the print ready bit-exit FAccel
0396 AA              Mov     GStR20,A            ;save the status byte
0397 83              Ret

             ; PG

             ; ---------------------------------------------------------------
             ;       Character Dot Generator Math
             ;         Look-up Table Page Vectoring
             ;         Print Head Firing
             ; ---------------------------------------------------------------

0398 AE      GCHAR1: MOV     StrCR1,A                   ;STORE THE CHAR

             ;   screen for printable char [char +(cpl 20 Hex + 1 = E0 Hex)]
0399 03E0            ADD     A,#0E0H
039B F69F            JC      PrntCh
039D 64C9            jmp     CntlCh              ;jmp to control char lookup table
039F 97      PrntCh: Clr     C                   ;clear carry flag
03A0 FE              Mov     A,StrCR1            ;get the char again

             ;   screen for char page [char +(cpl 50 Hex + 1 = B0 Hex)]
             ;      if carry char on page 2 else page 1
03A1 03B0            ADD     A,#0B0H
03A3 F6AE            JC      Page2

             ;       Page 1 Character -- ASCII 20 Hex thru 4F Hex
             ;         Correct offset for lookup table page
             ;         {(char + E0 Hex)*5 = Page 1 index addr}
             ; ---------------------------------------------------------------
03A5 FA      Page1:  Mov     A,ChStR1            ;get the status byte
03A6 4320            OrL     A,#ChOnP1           ;set the page rentry flag bit
03A8 AA              Mov     ChStR1,A            ;store the status byte
03A9 FE              Mov     A,StrCR1            ;get the char agian
03AA 03E0            ADD     A,#0E0H             ;set page 1 relative 00 offset
03AC 64B8            Jmp     Multi5              ;jump to address math function
```

230795-44

```
                ;       Page 2 Character -- ASCII 20 Hex thru 4F Hex
                ;         Correct offset for lookup table page two's complement
                ;         of ASCII chr code LookUp Table page base char of 50H plus
                ;         char * 5 ((char + B0 Hex)*5 = Page 2 index addr)
                ; ------------------------------------------------------------------
03AE 97         Page2:  Clr     C               ;clear carry flag
03AF FA                 Mov     A,ChStR1        ;get the status byte
03B0 53DF               AnL     A,#ChOnP2       ;set the page rentry flag bit
03B2 AA                 Mov     ChStR1,A        ;store the status byte
03B3 FE                 Mov     A,StrCR1        ;get the char agian
03B4 03B0               ADD     A,#0B0H         ;set page 2 relative 00 offset
03B6 64B8               Jmp     Multi5          ;fall thru to address math function

                ;       Compute character page offset dot pattern index address
03B8 AE         MULTI5: Mov     StrCR1,A        ;store the zero offset char
03B9 E7                 RL      A               ;MULTIPLY CHR BY 5 TO
03BA E7                 RL      A               ;  FIND THE ADDRESS
03BB 6E                 ADD     A,StrCR1        ;ADD 1 TO COMPLETE 5X
03BC AC                 MOV     CDotR1,A        ;SAVE THE ADDRESS

                ;       Test for L-to-R (forward) or R-to-L (reverse) printing
                ;         (see GChar1 ASCII char code translation procedure)
                ; ------------------------------------------------------------------
03BD FA                 Mov     A,ChStR1        ;get char status byte
03BE F2C4               JB7     LRPrn           ;test Chr Stat Byte Returned
                                                ;  if bit 7 = 1 then Print L-toR
03C0 FC                 MOV     A,CDotR1        ;get the char index addr
03C1 0304               ADD     A,#RLPShf       ;add char offset - start at end
                                                ;of char, print it R-to-L
03C3 AC                 MOV     CDotR1,A        ;SAVE THE ADDRESS

                ;       Set the status byte for Character SetUp done
                ; ------------------------------------------------------------------
03C4 FA         LRPrn:  Mov     A,ChStR1        ;get the status byte
03C5 4340               ORL     A,#ChIntD       ;set 1st char col test bit = 0
03C7 AA                 Mov     ChStR1,A        ;store the status byte
03C8 83                 Ret                     ;return w/status byte in A
                ;       test for non printable characters goes here
03C9 83         CntlCh: Ret

                ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                ;       Print Head Fire
                ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                ;       Entry point for print head solenoid firing
                ;         - test for status byte for dot/blank column position
03CA D5         PHFire: SEL     RB1
03CB FB                 Mov     A,CDtCR1        ;set the chr dot column cnt
03CC 96D2               JNZ     Fire            ;if char cnt not 0 - Fire Head Sol.
                                                ;if Chr Dot Cnt 0, reset the
03CE BB06       SetCnt: Mov     CDtCR1,#NDtCCt  ;  char dot column count
03D0 64D8               Jmp     Retrn1          ;skip PH Fire
03D2 2340       Fire:   MOV     A,#PTrgLo       ;get the Prnt Head Trigger byte
03D4 3A                 OUTL    P2,A            ;FIRE PRINT HEAD
03D5 23C0               MOV     A,#PTrgHi       ;get the Prnt Head Trigger byte
03D7 3A                 OUTL    P2,A            ;FIRE PRINT HEAD
03D8 C5         Retrn1: SEL     RB0
03D9 83                 Ret                     ;EXIT - return w/ Reg Bank 0 Reset

                ; PG
                ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                ;       PaperFeed Stpr Mtr Drive
                ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

0400                    ORG     400H

                ;       Init psuedo register with LF inderect addr start - subsequent
                ;         exchanges of the psuedo register will yield correct value
0400 BC04       InitLF: MOV     CntR40,#ILFCNT  ;INIT PHASE COUNT REG
0402 B822               Mov     TmpR00,#LPSAdr  ;get Phz Inderect Addr psuedo reg
0404 2308               MOV     A,#StLFF        ;get LF starting addr
0406 A0                 Mov     @TmpR00,A       ;store LF phase index addr start
                                                ;  in psuedo register
0407 BE01               Mov     LnCtR0,#LineCt  ;set line count reg for 1 ln
                                                ; enables exit following LF SM init
0409 841B               Jmp     LfDrv1          ;jump over line/form feed amd variable
                                                ;  line spacing tests & setups

                ;       LineFeed / FormFeed Drive
```

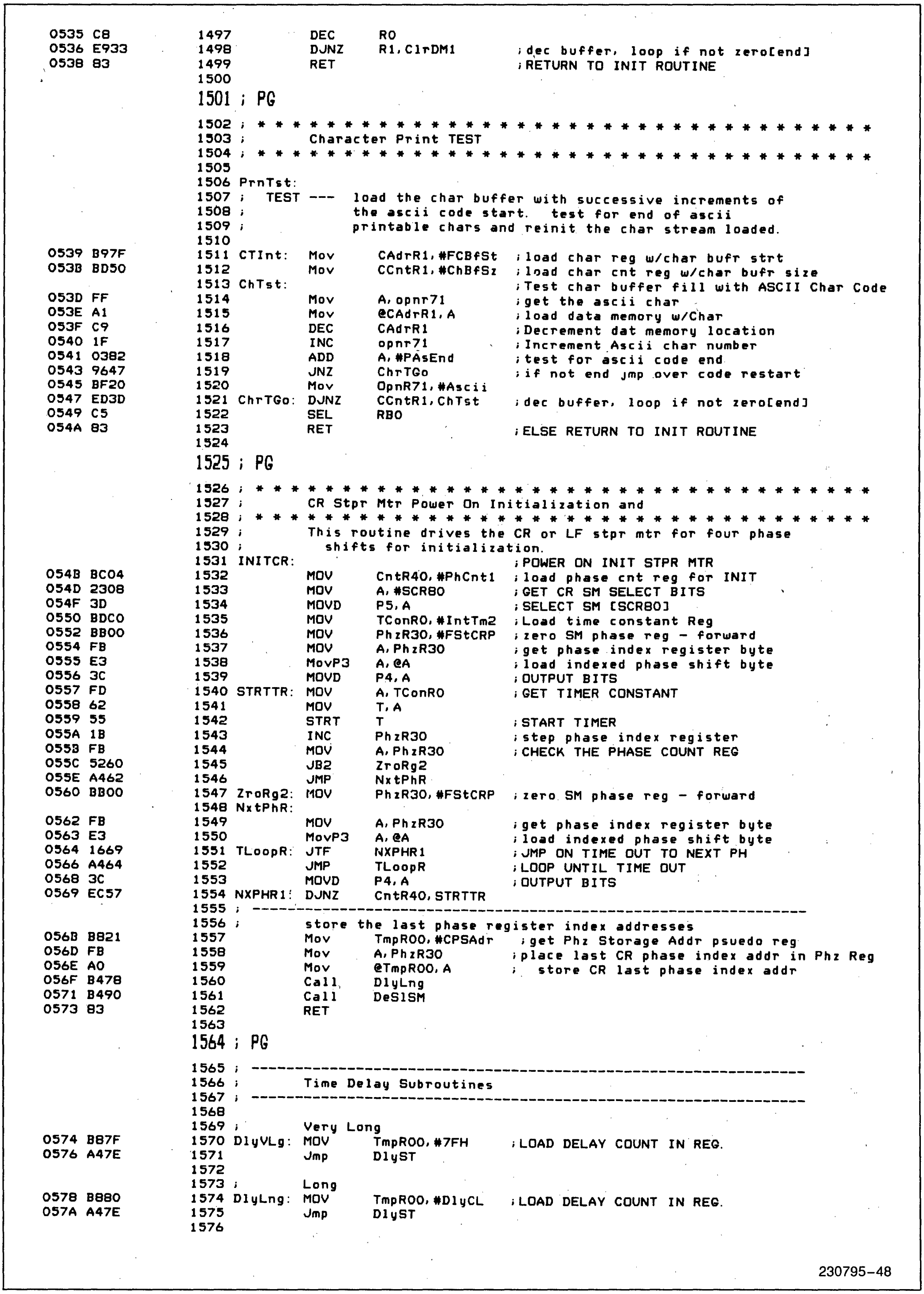
230795-45

```
                ; ------------------------------------------------------------------------

                ;        load step count constant for standard line spacing
                ; ------------------------------------------------------------------------
                ;        test for various line/inch spacing would go here
                ;          (and removal of constant setup below)
040B BC1B                MOV     CntR40,#LPI8p8  ;init cnt reg for standard line feed

                ;        LineFeed/FormFeed Test
040D FA         LfDriv:  Mov     A,GStR20        ;get the status byte
040E 5214                JB2     FmFd            ;if linefeed jmp to cnt load
0410 BE01       LnCtLd:  Mov     LnCtR0,#LineCt  ;set line count reg for 1 line
0412 841B                Jmp     LfDrv1          ;jmp to Start of Drive
0414 FE         FmFd:    Mov     A,LnCtR0        ;get the line count
0415 37                  Cpl     A               ;2's cpl Line Count
0416 0301                Add     A,#01
0418 0342                Add     A,#PgLnCt       ;Add 2's cpl for Paging
                                                 ;  PgLnCt - LnCt = n Lines to move
                                                 ;  PgLnCt+(cpl(LnCt) = n lines to move
041A AE                  Mov     LnCtR0,A        ;set the line count for FF

                ;        for stablization of unused stpr mtr during CR stpr mtr drive,
                ;          store the unused stpr mtr current phase bits
041B B821       LFDrv1:  Mov     TmpR00,#CPSAdr  ;get the CR phz storeage addr
041D F0                  Mov     A,@TmpR00       ;get the byute stored there
041E E3                  MovP3   A,@A            ;get the phz data byte
041F B820                Mov     TmpR00,#LastPh  ;load Last Phz psuedo reg to Temp Reg
0421 A0                  Mov     @TmpR00,A       ;store Last Phase bits - inderect
                ;        exchange/store the phase register index addresses
0422 B822                Mov     TmpR00,#LPSAdr  ;get Phz Inderect Addr psuedo reg
0424 F0                  Mov     A,@TmpR00       ;get LF last phase index addr
0425 AB                  Mov     PhzR30,A        ;place last LF phase index addr in Phz Reg
0426 BD98                MOV     TConR0,#LFTMR1  ;Load time constant Reg

                ;        Select the Stpr Mtr
0428 2306                MOV     A,#SLF          ;GET CR SM SELECT BITS
042A 3D                  MOVD    P5,A            ;SELECT SM [SCR80]

                ; ------------------------------------------------------------------------
                ;        LineFeed / FormFeed Drive Loop
                ; ------------------------------------------------------------------------
042B FB                  MOV     A,PhzR30        ;get the phz reg indirect addr index
042C E3                  MovP3   A,@A            ;do indirect get of phz bits
                ;        patch together the CR last and LF next phase bits
042D B820                Mov     TmpR00,#LastPh  ;load Last Phz psuedo reg to Temp Reg
042F 40                  ORL     A,@TmpR00       ;patch together CR existing & new LF
                ;        start timer and step motor
0430 3C                  MOVD    P4,A            ;OUTPUT BITS

                StrtLF:
0431 FD         STRLFT:  MOV     A,TConR0        ;get time constant from reg
0432 62                  MOV     T,A             ;load the timer
0433 55                  STRT    T               ;START TIMER
                ;        setup the next phase to output
0434 1B                  INC     PhzR30          ;STEP PHASE DB ADDRESS
0435 FB                  MOV     A,PhzR30        ;get the phase index address
0436 523A                JB2     ZROPHL          ;test phase
0438 843C                JMP     NXTPHL
043A BB08       ZROPHL:  MOV     PhzR30,#STLFF   ;re-init phase register

043C FB         NXTPHL:  MOV     A,PhzR30        ;get the phz reg indirect addr index
043D E3                  MovP3   A,@A            ;do indirect get of phz bits
                ;        patch together the CR last and LF next phase bits
043E B820                Mov     TmpR00,#LastPh  ;load Last Phz psuedo reg to Temp Reg
0440 40                  ORL     A,@TmpR00       ;patch together CR existing & new LF

0441 1645       TLoopL:  JTF     NXPHLF          ;jmp on time out to output nxt phz
0443 8441                JMP     TLOOPL          ;loop until timer times out

0445 3C         NXPHLF:  MOVD    P4,A            ;step motor - OUTPUT BITS
0446 EC31                DJNZ    CntR40,StrLFT   ;test for end of phase count for line
                                                 ;prep for next line

                ;        test for various line/inch spacing would go here
0448 BC1B                MOV     CntR40,#LPI8p8  ;init cnt reg for standard line feed
044A EE31                DJNZ    LnCtR0,StrtLF   ;test for end of line count

044C FA                  Mov     A,GStR20        ;Get the status byte
044D 53FB                ANL     A,#LineFd       ;reset for line feed
044F AA                  Mov     GStR20,A        ;save the status byte

                ;        store the phase register index addresses
                ;        Set LineFeed Stpr Mtr Next Phase index address
0450 B822       SetLRN:  Mov     TmpR00,#LPSAdr  ;get Phz Storage Addr psuedo reg
```

230795-46

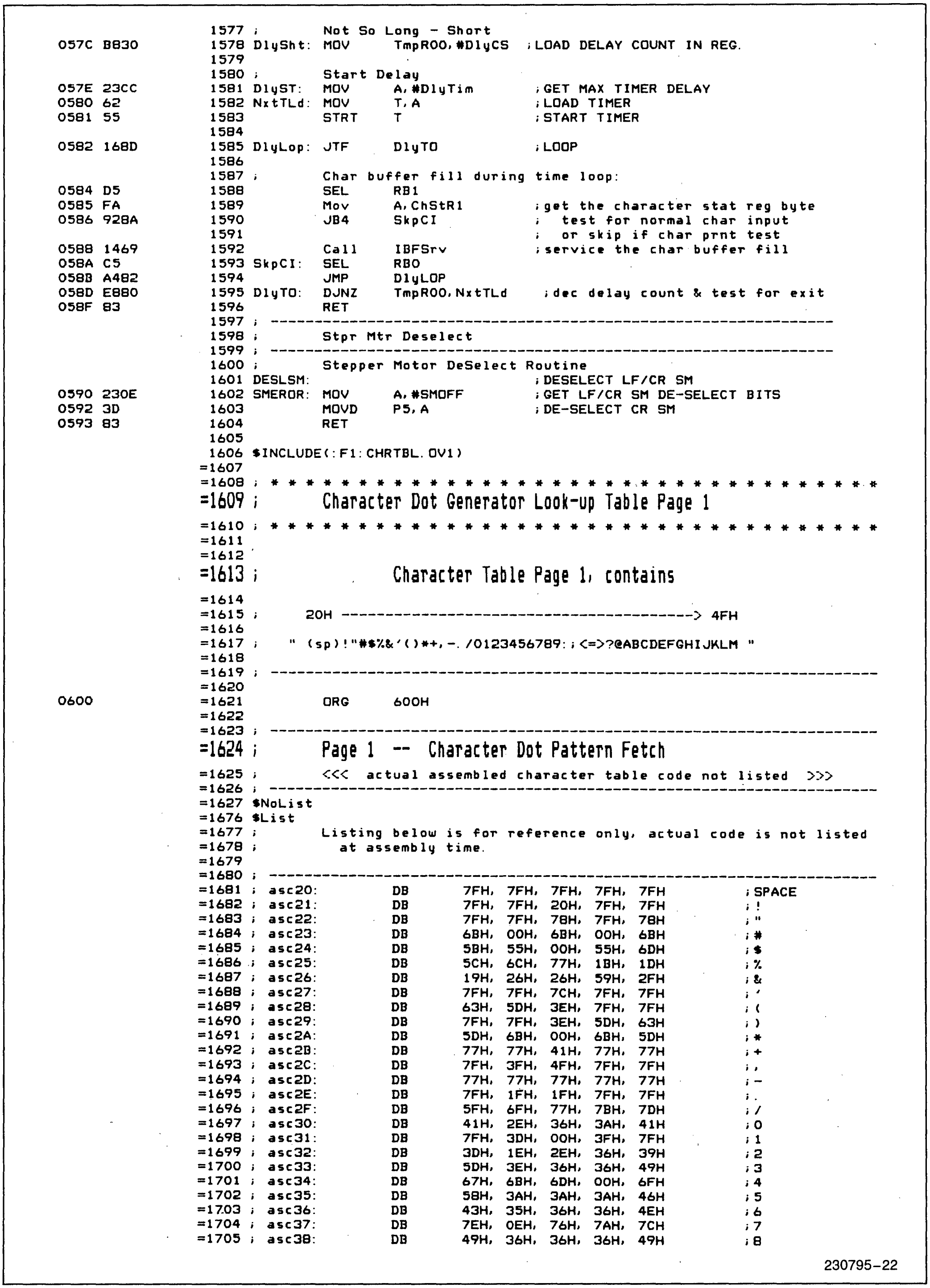
```
0452 FB                       Mov     A,PhzR30         ;get the phase index address
0453 A0                       Mov     @TmpR00,A        ;store LF Next phase index addr
0454 B478                     Call    DlyLng
0456 B490                     Call    DeS1SM

          ;        Check if Char Buffer contains full line (80 char or nChar & CR)
          ;          exit otherwise continue to read in characters
          ;         Mov     A,GStR20         ;get the stat byte
          ;         JB1     ByPas1           ;if Do Not Print Bit Set - EXIT
          ;         Call    CBFck
0458 83   ByPas1:  Ret

          ; PG
          ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
          ;        Minor Software Subroutines
          ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

0500                          ORG     500H

          ; --------------------------------------------------------------------
          ;        System initialization subroutines
          ; --------------------------------------------------------------------
          Defalt:
          ; --------------------------------------------------------------------
          ;        reset/set EOF status flag bit = 0
0500 D5                       SEL     RB1
0501 FA                       Mov     A,ChStR1         ;get the char status byte
0502 53F7                     ANL     A,#ClrEOF        ;clear the EOF flag bit
0504 AA                       Mov     ChStR1,A         ;store the char status byte
0505 B823                     Mov     TmpR10,#PTAscS   ;get the Ascii code tmp store addr
0507 B020                     Mov     @TmpR10,#Ascii   ;load the tmp stor reg w/ascii start
0509 C5                       SEL     RB0
          ; --------------------------------------------------------------------
          ;        reset/set Ok-to-Print status flag bit = 0
050A FA                       Mov     A,GStR20         ;get the status byte
050B 53FD                     ANL     A,#OkPrnt        ;reset print flag - Ok Print
050D AA                       Mov     GStR20,A         ;save the status byte
050E 83                       RET
          InitAl:
          AllOff:
          ; --------------------------------------------------------------------
          ;        CLEAR all outputs
050F C5                       SEL     RB0
0510 230F                     MOV     A,#0FH           ;FORCE PORT HI - R/ OF 555
0512 3E                       MOVD    P6,A
0513 23FF                     MOV     A,#0FFH          ;TURN ALL PRNT SOL's OFF
0515 39                       OUTL    P1,A
0516 23C0                     MOV     A,#PTRGHI        ;print head fire tirgger inactive
0518 3A                       OUTL    P2,A
0519 8A03                     ORL     P2,#03           ;set comm hdsk to ACK hi/Busy hi
051B BA00                     Mov     GStR20,#00H      ;clear the status registers
051D D5                       SEL     RB1
051E BA00                     Mov     ChStR1,#00H
0520 C5                       SEL     RB0
0521 83                       RET                      ;RETURN TO INIT ROUTINE

          ; PG
          ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
          ;       Home Carriage / Print Head Assembly
          ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

0522 FA   CRHome:  Mov     A,GStR20         ;get the status byte
0523 4302          ORL     A,#DoNotP        ;set the do not print flag
0525 AA            Mov     GStR20,A         ;save the status byte
0526 362A          JT0     RtoL             ;test for position of PH assembly
                                            ;  drive accordingly
0528 3402          Call    FAccel           ;drive CR Stpr Mtr
052A 3422 RtoL:    Call    RAccel           ;find the logical left home CR position
                                            ;delay a long time before continuing
052C B474          Call    DlyVLg
052E 83            Ret

          ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
          ;       Clear Data Memory
          ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

          ;       At PowerUp or Reset, following CR & LF Stpr Mtr Init, this
          ;       procedure clears data memory above RB0, Stack and RB1.
052F B87F ClrDM:   MOV     R0,#DMTop        ;GET BUFFER START LOCATION [HEX]
0531 B95D          MOV     R1,#DMSIZE
0533 B000 ClrDM1:  MOV     @R0,#00H         ;ZERO MEMORY LOCATION
```

230795-47



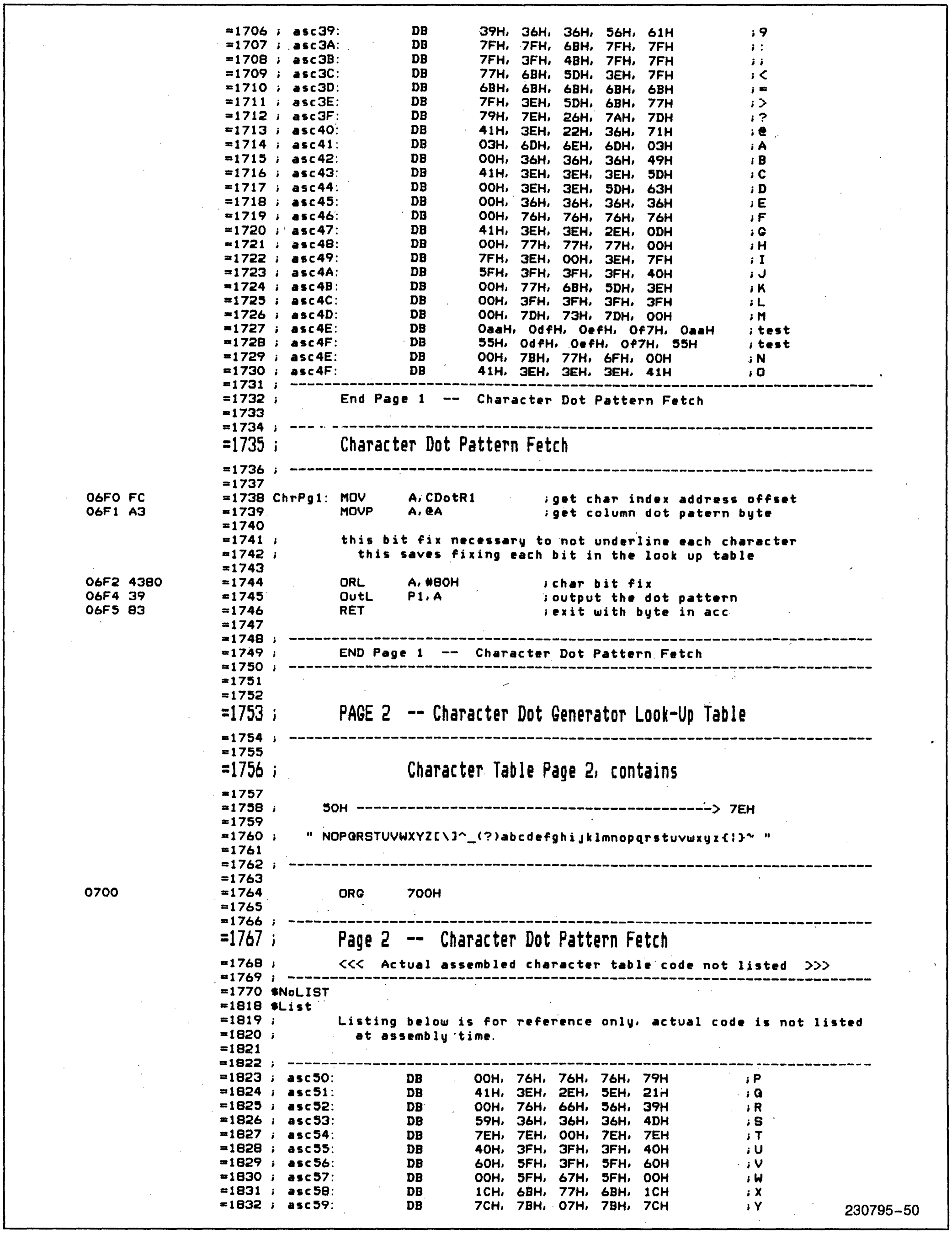
```
0535 C8                    DEC     R0
0536 E933                  DJNZ    R1,ClrDM1         ;dec buffer, loop if not zero[end]
0538 83                    RET                       ;RETURN TO INIT ROUTINE

                 ; PG
                 ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                 ;         Character Print TEST
                 ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                 PrnTst:
                 ;  TEST ---  load the char buffer with successive increments of
                 ;            the ascii code start.  test for end of ascii
                 ;            printable chars and reinit the char stream loaded.

0539 B97F        CTInt:    Mov     CAdrR1,#FCBfSt    ;load char reg w/char bufr strt
053B BD50                  Mov     CCntR1,#ChBfSz    ;load char cnt reg w/char bufr size
                 ChTst:                              ;Test char buffer fill with ASCII Char Code
053D FF                    Mov     A,opnr71          ;get the ascii char
053E A1                    Mov     @CAdrR1,A         ;load data memory w/Char
053F C9                    DEC     CAdrR1            ;Decrement dat memory location
0540 1F                    INC     opnr71            ;Increment Ascii char number
0541 0382                  ADD     A,#PAsEnd         ;test for ascii code end
0543 9647                  JNZ     ChrTGo            ;if not end jmp over code restart
0545 BF20                  Mov     OpnR71,#Ascii
0547 ED3D        ChrTGo:   DJNZ    CCntR1,ChTst      ;dec buffer, loop if not zero[end]
0549 C5                    SEL     RB0
054A 83                    RET                       ;ELSE RETURN TO INIT ROUTINE

                 ; PG
                 ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                 ;         CR Stpr Mtr Power On Initialization and
                 ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                 ;         This routine drives the CR or LF stpr mtr for four phase
                 ;           shifts for initialization.
                 INITCR:                             ;POWER ON INIT STPR MTR
054B BC04                  MOV     CntR40,#PhCnt1    ;load phase cnt reg for INIT
054D 2308                  MOV     A,#SCRB0          ;GET CR SM SELECT BITS
054F 3D                    MOVD    P5,A              ;SELECT SM [SCRB0]
0550 BDC0                  MOV     TConR0,#IntTm2    ;Load time constant Reg
0552 BB00                  MOV     PhzR30,#FStCRP    ;zero SM phase reg - forward
0554 FB                    MOV     A,PhzR30          ;get phase index register byte
0555 E3                    MovP3   A,@A              ;load indexed phase shift byte
0556 3C                    MOVD    P4,A              ;OUTPUT BITS
0557 FD          STRTTR:   MOV     A,TConR0          ;GET TIMER CONSTANT
0558 62                    MOV     T,A
0559 55                    STRT    T                 ;START TIMER
055A 1B                    INC     PhzR30            ;step phase index register
055B FB                    MOV     A,PhzR30          ;CHECK THE PHASE COUNT REG
055C 5260                  JB2     ZroRg2
055E A462                  JMP     NxtPhR
0560 BB00        ZroRg2:   MOV     PhzR30,#FStCRP    ;zero SM phase reg - forward
                 NxtPhR:
0562 FB                    MOV     A,PhzR30          ;get phase index register byte
0563 E3                    MovP3   A,@A              ;load indexed phase shift byte
0564 1669        TLoopR:   JTF     NXPHR1            ;JMP ON TIME OUT TO NEXT PH
0566 A464                  JMP     TLoopR            ;LOOP UNTIL TIME OUT
0568 3C                    MOVD    P4,A              ;OUTPUT BITS
0569 EC57        NXPHR1:   DJNZ    CntR40,STRTTR
                 ; ----------------------------------------------------------------
                 ;         store the last phase register index addresses
056B B821                  Mov     TmpR00,#CPSAdr    ;get Phz Storage Addr psuedo reg
056D FB                    Mov     A,PhzR30          ;place last CR phase index addr in Phz Reg
056E A0                    Mov     @TmpR00,A         ;   store CR last phase index addr
056F B478                  Call    DlyLng
0571 B490                  Call    DeS1SM
0573 83                    RET

                 ; PG
                 ; ----------------------------------------------------------------
                 ;         Time Delay Subroutines
                 ; ----------------------------------------------------------------

                 ;         Very Long
0574 B87F        DlyVLg:   MOV     TmpR00,#7FH       ;LOAD DELAY COUNT IN REG.
0576 A47E                  Jmp     DlyST

                 ;         Long
0578 B880        DlyLng:   MOV     TmpR00,#DlyCL     ;LOAD DELAY COUNT IN REG.
057A A47E                  Jmp     DlyST

```

230795-48



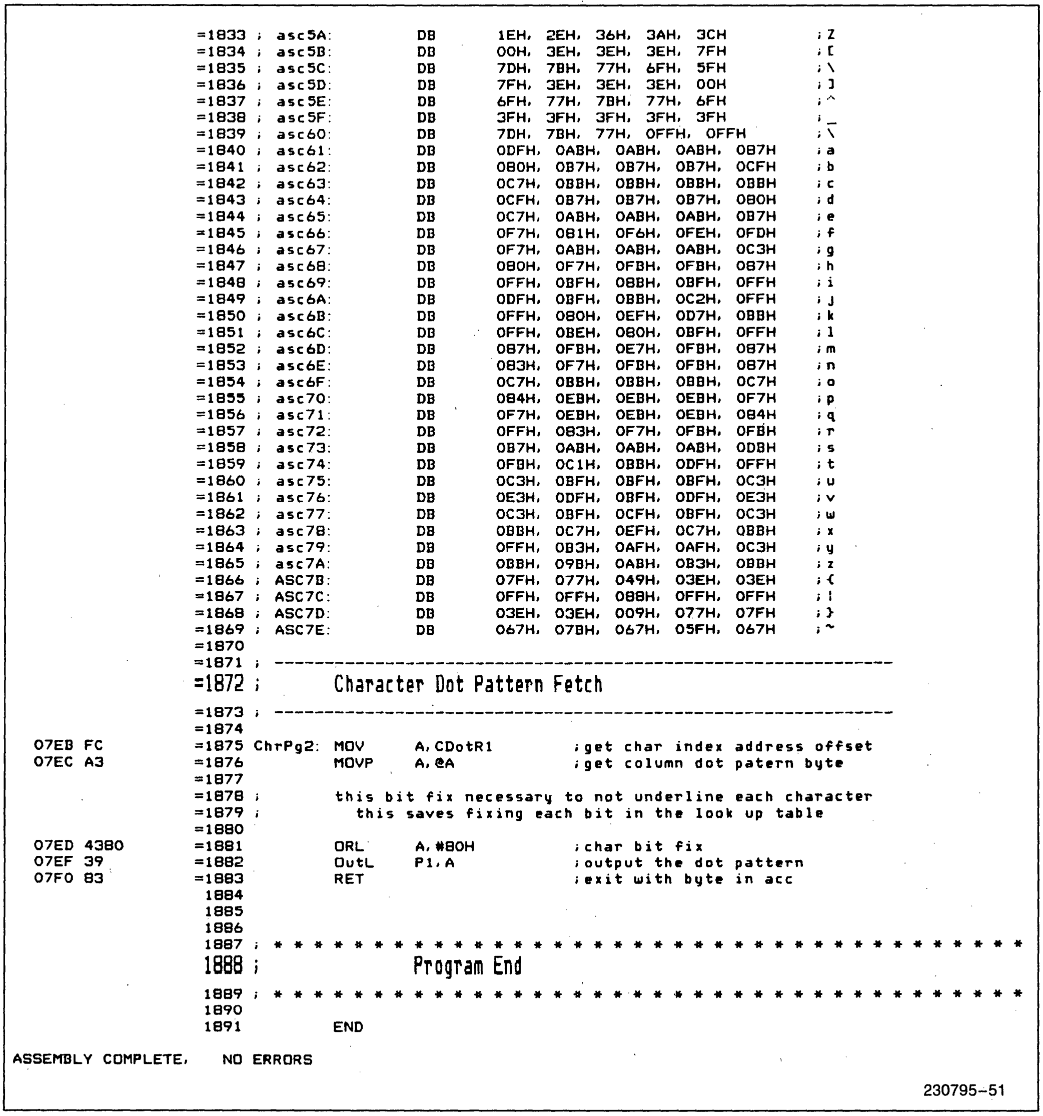
```
                  ;        Not So Long - Short
057C B830      DlySht: MOV      TmpROO,#DlyCS  ;LOAD DELAY COUNT IN REG.

                  ;        Start Delay
057E 23CC      DlyST:  MOV      A,#DlyTim      ;GET MAX TIMER DELAY
0580 62        NxtTLd: MOV      T,A            ;LOAD TIMER
0581 55                STRT     T              ;START TIMER

0582 168D      DlyLop: JTF      DlyTO          ;LOOP

                  ;        Char buffer fill during time loop:
0584 D5                SEL      RB1
0585 FA                Mov      A,ChStR1       ;get the character stat reg byte
0586 928A              JB4      SkpCI          ;  test for normal char input
                                               ;  or skip if char prnt test
0588 1469              Call     IBFSrv         ;service the char buffer fill
058A C5        SkpCI:  SEL      RBO
058B A482              JMP      DlyLOP
058D E880      DlyTO:  DJNZ     TmpROO,NxtTLd   ;dec delay count & test for exit
058F 83                RET
                  ; ------------------------------------------------------------------
                  ;        Stpr Mtr Deselect
                  ; ------------------------------------------------------------------
                  ;        Stepper Motor DeSelect Routine
               DESLSM:                         ;DESELECT LF/CR SM
0590 230E      SMEROR: MOV      A,#SMOFF       ;GET LF/CR SM DE-SELECT BITS
0592 3D                MOVD     P5,A           ;DE-SELECT CR SM
0593 83                RET

               $INCLUDE(:F1:CHRTBL.OV1)

                  ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                  ;        Character Dot Generator Look-up Table Page 1
                  ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


                  ;                Character Table Page 1, contains

                  ;      20H --------------------------------------> 4FH

                  ;    " (sp)!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLM "

                  ; ----------------------------------------------------------------------

0600                   ORG      600H

                  ; ----------------------------------------------------------------------
                  ;        Page 1 -- Character Dot Pattern Fetch
                  ;        <<<  actual assembled character table code not listed  >>>
                  ; ----------------------------------------------------------------------
               $NoList
               $List
                  ;        Listing below is for reference only, actual code is not listed
                  ;          at assembly time.

                  ; ----------------------------------------------------------------------
                  ; asc20:        DB      7FH, 7FH, 7FH, 7FH, 7FH        ;SPACE
                  ; asc21:        DB      7FH, 7FH, 20H, 7FH, 7FH        ;!
                  ; asc22:        DB      7FH, 7FH, 78H, 7FH, 78H        ;"
                  ; asc23:        DB      6BH, 00H, 6BH, 00H, 6BH        ;#
                  ; asc24:        DB      5BH, 55H, 00H, 55H, 6DH        ;$
                  ; asc25:        DB      5CH, 6CH, 77H, 1BH, 1DH        ;%
                  ; asc26:        DB      19H, 26H, 26H, 59H, 2FH        ;&
                  ; asc27:        DB      7FH, 7FH, 7CH, 7FH, 7FH        ;'
                  ; asc28:        DB      63H, 5DH, 3EH, 7FH, 7FH        ;(
                  ; asc29:        DB      7FH, 7FH, 3EH, 5DH, 63H        ;)
                  ; asc2A:        DB      5DH, 6BH, 00H, 6BH, 5DH        ;*
                  ; asc2B:        DB      77H, 77H, 41H, 77H, 77H        ;+
                  ; asc2C:        DB      7FH, 3FH, 4FH, 7FH, 7FH        ;,
                  ; asc2D:        DB      77H, 77H, 77H, 77H, 77H        ;-
                  ; asc2E:        DB      7FH, 1FH, 1FH, 7FH, 7FH        ;.
                  ; asc2F:        DB      5FH, 6FH, 77H, 7BH, 7DH        ;/
                  ; asc30:        DB      41H, 2EH, 36H, 3AH, 41H        ;0
                  ; asc31:        DB      7FH, 3DH, 00H, 3FH, 7FH        ;1
                  ; asc32:        DB      3DH, 1EH, 2EH, 36H, 39H        ;2
                  ; asc33:        DB      5DH, 3EH, 36H, 36H, 49H        ;3
                  ; asc34:        DB      67H, 6BH, 6DH, 00H, 6FH        ;4
                  ; asc35:        DB      58H, 3AH, 3AH, 3AH, 46H        ;5
                  ; asc36:        DB      43H, 35H, 36H, 36H, 4EH        ;6
                  ; asc37:        DB      7EH, 0EH, 76H, 7AH, 7CH        ;7
                  ; asc38:        DB      49H, 36H, 36H, 36H, 49H        ;8
```

230795-22



```
                ; asc39:        DB      39H, 36H, 36H, 56H, 61H         ;9
                ; asc3A:        DB      7FH, 7FH, 6BH, 7FH, 7FH         ;:
                ; asc3B:        DB      7FH, 3FH, 4BH, 7FH, 7FH         ;;
                ; asc3C:        DB      77H, 6BH, 5DH, 3EH, 7FH         ;<
                ; asc3D:        DB      6BH, 6BH, 6BH, 6BH, 6BH         ;=
                ; asc3E:        DB      7FH, 3EH, 5DH, 6BH, 77H         ;>
                ; asc3F:        DB      79H, 7EH, 26H, 7AH, 7DH         ;?
                ; asc40:        DB      41H, 3EH, 22H, 36H, 71H         ;@
                ; asc41:        DB      03H, 6DH, 6EH, 6DH, 03H         ;A
                ; asc42:        DB      00H, 36H, 36H, 36H, 49H         ;B
                ; asc43:        DB      41H, 3EH, 3EH, 3EH, 5DH         ;C
                ; asc44:        DB      00H, 3EH, 3EH, 5DH, 63H         ;D
                ; asc45:        DB      00H, 36H, 36H, 36H, 36H         ;E
                ; asc46:        DB      00H, 76H, 76H, 76H, 76H         ;F
                ; asc47:        DB      41H, 3EH, 3EH, 2EH, 0DH         ;G
                ; asc48:        DB      00H, 77H, 77H, 77H, 00H         ;H
                ; asc49:        DB      7FH, 3EH, 00H, 3EH, 7FH         ;I
                ; asc4A:        DB      5FH, 3FH, 3FH, 3FH, 40H         ;J
                ; asc4B:        DB      00H, 77H, 6BH, 5DH, 3EH         ;K
                ; asc4C:        DB      00H, 3FH, 3FH, 3FH, 3FH         ;L
                ; asc4D:        DB      00H, 7DH, 73H, 7DH, 00H         ;M
                ; asc4E:        DB      0aaH, 0dfH, 0efH, 0f7H, 0aaH    ;test
                ; asc4F:        DB      55H, 0dfH, 0efH, 0f7H, 55H      ;test
                ; asc4E:        DB      00H, 7BH, 77H, 6FH, 00H         ;N
                ; asc4F:        DB      41H, 3EH, 3EH, 3EH, 41H         ;O
                ; ------------------------------------------------------------------
                ;         End Page 1  --  Character Dot Pattern Fetch

                ; ------------------------------------------------------------------
                ;         Character Dot Pattern Fetch
                ; ------------------------------------------------------------------

06F0 FC         ChrPg1: MOV     A,CDotR1        ;get char index address offset
06F1 A3                 MOVP    A,@A            ;get column dot patern byte

                ;       this bit fix necessary to not underline each character
                ;         this saves fixing each bit in the look up table

06F2 4380               ORL     A,#80H          ;char bit fix
06F4 39                 OutL    P1,A            ;output the dot pattern
06F5 83                 RET                     ;exit with byte in acc

                ; ------------------------------------------------------------------
                ;         END Page 1  --  Character Dot Pattern Fetch
                ; ------------------------------------------------------------------


                ;         PAGE 2 -- Character Dot Generator Look-Up Table
                ; ------------------------------------------------------------------

                ;                 Character Table Page 2, contains

                ;     50H --------------------------------------------> 7EH

                ;   " NOPQRSTUVWXYZ[\]^_(?)abcdefghijklmnopqrstuvwxyz{!}~ "

                ; ------------------------------------------------------------------

0700                    ORG     700H

                ; ------------------------------------------------------------------
                ;         Page 2 -- Character Dot Pattern Fetch
                ;         <<< Actual assembled character table code not listed >>>
                ; ------------------------------------------------------------------
                $NoLIST
                $List
                ;       Listing below is for reference only, actual code is not listed
                ;         at assembly time.

                ; ------------------------------------------------------------------
                ; asc50:        DB      00H, 76H, 76H, 76H, 79H         ;P
                ; asc51:        DB      41H, 3EH, 2EH, 5EH, 21H         ;Q
                ; asc52:        DB      00H, 76H, 66H, 56H, 39H         ;R
                ; asc53:        DB      59H, 36H, 36H, 36H, 4DH         ;S
                ; asc54:        DB      7EH, 7EH, 00H, 7EH, 7EH         ;T
                ; asc55:        DB      40H, 3FH, 3FH, 3FH, 40H         ;U
                ; asc56:        DB      60H, 5FH, 3FH, 5FH, 60H         ;V
                ; asc57:        DB      00H, 5FH, 67H, 5FH, 00H         ;W
                ; asc58:        DB      1CH, 6BH, 77H, 6BH, 1CH         ;X
                ; asc59:        DB      7CH, 7BH, 07H, 7BH, 7CH         ;Y
```

230795-50



```
             ;  asc5A:        DB        1EH,  2EH,  36H,  3AH,  3CH               ; Z
             ;  asc5B:        DB        00H,  3EH,  3EH,  3EH,  7FH               ; [
             ;  asc5C:        DB        7DH,  7BH,  77H,  6FH,  5FH               ; \
             ;  asc5D:        DB        7FH,  3EH,  3EH,  3EH,  00H               ; ]
             ;  asc5E:        DB        6FH,  77H,  7BH,  77H,  6FH               ; ^
             ;  asc5F:        DB        3FH,  3FH,  3FH,  3FH,  3FH               ; _
             ;  asc60:        DB        7DH,  7BH,  77H,  0FFH,  0FFH             ; \
             ;  asc61:        DB        0DFH,  0ABH,  0ABH,  0ABH,  087H          ; a
             ;  asc62:        DB        080H,  0B7H,  0B7H,  0B7H,  0CFH          ; b
             ;  asc63:        DB        0C7H,  0BBH,  0BBH,  0BBH,  0BBH          ; c
             ;  asc64:        DB        0CFH,  0B7H,  0B7H,  0B7H,  080H          ; d
             ;  asc65:        DB        0C7H,  0ABH,  0ABH,  0ABH,  0B7H          ; e
             ;  asc66:        DB        0F7H,  081H,  0F6H,  0FEH,  0FDH          ; f
             ;  asc67:        DB        0F7H,  0ABH,  0ABH,  0ABH,  0C3H          ; g
             ;  asc68:        DB        080H,  0F7H,  0FBH,  0FBH,  087H          ; h
             ;  asc69:        DB        0FFH,  0BFH,  08BH,  0BFH,  0FFH          ; i
             ;  asc6A:        DB        0DFH,  0BFH,  0BBH,  0C2H,  0FFH          ; j
             ;  asc6B:        DB        0FFH,  080H,  0EFH,  0D7H,  0BBH          ; k
             ;  asc6C:        DB        0FFH,  0BEH,  080H,  0BFH,  0FFH          ; l
             ;  asc6D:        DB        087H,  0FBH,  0E7H,  0FBH,  087H          ; m
             ;  asc6E:        DB        083H,  0F7H,  0FBH,  0FBH,  087H          ; n
             ;  asc6F:        DB        0C7H,  0BBH,  0BBH,  0BBH,  0C7H          ; o
             ;  asc70:        DB        084H,  0EBH,  0EBH,  0EBH,  0F7H          ; p
             ;  asc71:        DB        0F7H,  0EBH,  0EBH,  0EBH,  084H          ; q
             ;  asc72:        DB        0FFH,  083H,  0F7H,  0FBH,  0FBH          ; r
             ;  asc73:        DB        0B7H,  0ABH,  0ABH,  0ABH,  0DBH          ; s
             ;  asc74:        DB        0FBH,  0C1H,  0BBH,  0DFH,  0FFH          ; t
             ;  asc75:        DB        0C3H,  0BFH,  0BFH,  0BFH,  0C3H          ; u
             ;  asc76:        DB        0E3H,  0DFH,  0BFH,  0DFH,  0E3H          ; v
             ;  asc77:        DB        0C3H,  0BFH,  0CFH,  0BFH,  0C3H          ; w
             ;  asc78:        DB        0BBH,  0C7H,  0EFH,  0C7H,  0BBH          ; x
             ;  asc79:        DB        0FFH,  0B3H,  0AFH,  0AFH,  0C3H          ; y
             ;  asc7A:        DB        0BBH,  09BH,  0ABH,  0B3H,  0BBH          ; z
             ;  ASC7B:        DB        07FH,  077H,  049H,  03EH,  03EH          ; {
             ;  ASC7C:        DB        0FFH,  0FFH,  088H,  0FFH,  0FFH          ; ¦
             ;  ASC7D:        DB        03EH,  03EH,  009H,  077H,  07FH          ; }
             ;  ASC7E:        DB        067H,  07BH,  067H,  05FH,  067H          ; ~

             ; --------------------------------------------------------------
             ;         Character Dot Pattern Fetch
             ; --------------------------------------------------------------

07EB FC      ChrPg2:  MOV     A,CDotR1            ;get char index address offset
07EC A3               MOVP    A,@A                ;get column dot patern byte

             ;        this bit fix necessary to not underline each character
             ;          this saves fixing each bit in the look up table

07ED 4380             ORL     A,#80H              ;char bit fix
07EF 39               OutL    P1,A                ;output the dot pattern
07F0 83               RET                         ;exit with byte in acc



             ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
             ;              Program End
             ; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                      END

ASSEMBLY COMPLETE,    NO ERRORS
```

230795-51

APPENDIX B
SOFTWARE PRINTER ENHANCEMENTS

This section describes several software enhancements which could be implemented as additions to the software developed for this Application Note. Space is available for most of the items described. Approximately 5 bytes of Data Memory would be required to implement most of the features. Two bytes would be used for status flags, and two bytes for temporary data or count storage. It is possible to use less than five bytes, but this would require the duplicate use of some flags, or other Data Memory storage, which will significantly complicate the software coding and debug tasks.

Special Characters or Symbols

Dot matrix printing lends itself well to the creation of custom characters and symbols. There are two aspects to implementing special characters. First, a character look-up table, and second, additional software for decoding and processing the special characters or symbols. Special characters might be scientific notation, mathematical symbols, unique language characters, or block and line graphics characters.

The character look-up table could be an additional page of Program Memory dedicated to the special characters, or replace part, or all, of the existing look-up tables. If an additional look-up table is used, a third page test would be needed at the beginning of the Character Translation subroutine. There is fundamentally no difference between the processing of special characters and standard ASCII printable characters. If the characters require the same 5 x 7 dot matrix, the balance of the software would remain the same. If, however, the special characters require a different matrix, or the manipulation of the matrix, the software becomes more complex.

In general, the major software modification required to implement special characters is the size of the dot matrix printed or the dot matrix configuration used. In the case of scientific characters, it would often be necessary to shift the 5 x 7 matrix pattern within the available 9 x 9 matrix. Block or line graphics characters, on-the-other-hand, would require using the entire 9 x 9 print head matrix and printing during normally blank dot columns. This would require suspending the blank column blanking mechanism implemented in this Application Note. This would be the most complex aspect of implementing special characters. It would possibly change the number of required instructions, and thus the timing between PTS detection and print head solenoid trigger firing. This could cause the dot columns to be misaligned within a printed line and between lines.

In the case of a matrix change, two approaches are possible: dynamically changing the matrix, in line, as standard ASCII characters are being printed, or isolating the special characters to a separate processing flow where special characters are handled as a unique and complete line of characters only. A discussion of in line matrix changes for special characters is beyond the scope of this Appendix. It is sufficient to say that the changes would require the conditions setting the EOLN flag, character count, and dot column count software be modified during character processing and printing.

Lower Case Descenders

The general principle of implementing lower case descenders is to shift the 5 x 7 character dot matrix within the available 9 x 9 print head solenoid matrix. Implementing lower case descenders requires two software modifications and the creation of status flag for the purpose. First, the detection of characters needing descenders and setting a dedicated status flag during the character code to dot pattern translation subroutine. Second, the character dot column data output to the print head solenoids must be shifted for each dot column of the character. At the end of the character, the flag would be reset.

Inline Control Codes

Inline control codes are two to three character sequences, which indicate special hardware conditions or software flow control and branching. The first character indicates that the control code sequence is beginning and is typically an ASCII Escape Character (ESC), 1BH. Termination of the inline code sequence would be indicated by a default number of code sequence characters. This would decrease the buffer size available for characters. Full 80 character line buffering would require loading the Character Buffer with a received character as a character is removed from it and processed.

The Inline Control Code test would be performed in two places: in the Character Buffer Fill subroutine and in the Character Processing (translation) subroutine. The test would be performed in the same manner that a Carriage Return (CR) character code test is implemented. Examples are horizontal tabs and expanded or condensed character fonts. In the case of horizontal tabs, 20H (Space Character) would have to be placed in the Character Buffer for inline processing during character processing and printing. Unless fixed position tabs are used, a minimum of a nibble of Data Memory would be required to maintain a "spaces-to-tab" count. Fixed tab positions could be set via another inline control code, by default of the print software, or through the use of external hardware switch settings. The control code method of setting the tab positions is the most desirable, but the most complex to implement.

Different Character Formats

Condensed and enlarged characters are variations in either the number of dots and/or the space used to print them. Thus, each character is a variation of the stepper motor and/or print head solenoid trigger timings. It is possible to print each in bold face by printing each dot twice per dot column position. This would require little software modification, but would require a status flag. Again, care must be used to ensure that the delay in retriggering the solenoids is precisely the same for each type of event. Without this precise timing the dot column alignment will not be accurate. The software modifications needed to implement enlarged or condensed characters is essentially the same. The carriage and print head solenoid firing software flow is the same, but the timing for each changes. For condensed characters, the step Time Constant is doubled to approximately 4.08 ms, and the solenoids are fired four times within each step time. The step rate actually becomes a multiple of the solenoid firing time, and a counter incrementing once for each solenoid firing would be needed. At the count of four, the carriage stepper motor is stepped and the counter reset.

In the case of condensed characters, PTS does not play the same roll as in standard or enlarged character printing. PTS is not used to indicate the optimum print head solenoid firing time. Solenoid firing is purely a time function for condensed characters. PTS would only be used for Failsafe protection.

Enlarged characters would require the solenoids be fired twice per dot column data, in two sequential dot columns, at the same rate as standard characters. The character dot column data and dot column count would not be incremented at each output but at every other output. A flag could be used for this purpose.

When printing either condensed or enlarged characters, the maximum character count would have to compensate for the increased or decreased characters per line count. When printing enlarged characters, the maximum characters per line would be 40.

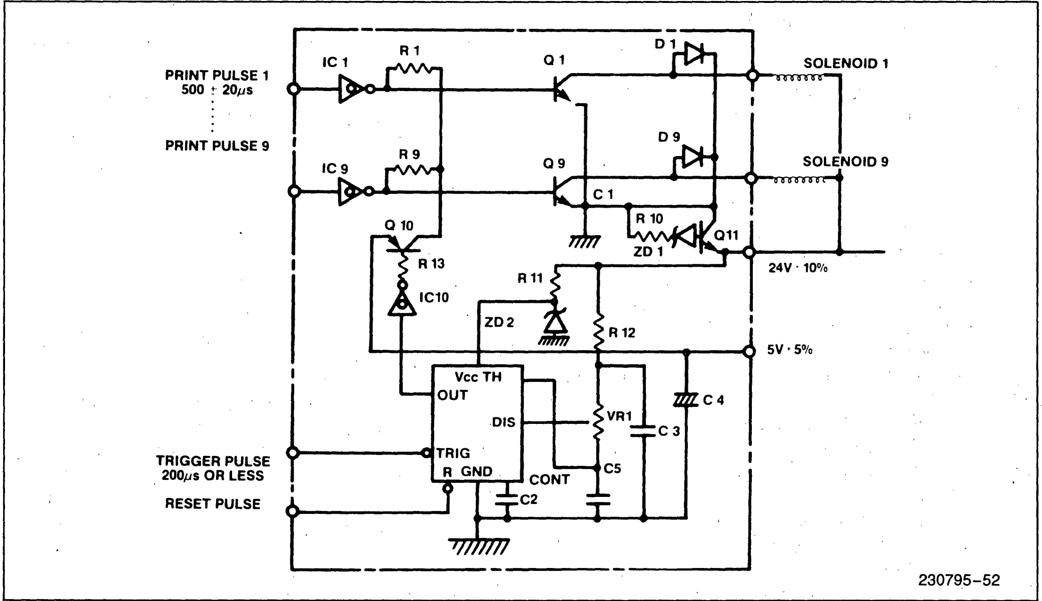
The Character Buffer could hold two complete lines of characters. But, condensed characters presents a quite different situation. The avalable character per line increases to 132, well beyond the 80 character Character Buffer size. The solution is to re-initialize the Character Buffer Size Count register count during condensed character processing. This will effectively inhibit the carriage stepper motor drive EOLN detection.

Two status flags would be required; one for standard or enlarged characters, and the second for condensed characters. A third status flag would be required to implement bold face printing. Activating one of the alternate character fonts could be either through the use of external status switches or through inline control code sequences, as detailed above. Note, that if the alternate character fonts are implemented in such a way that format changing is to occur dynamically during any single line being printed, the same control code problems described above also apply. In addition, the effect on the timing and dot column alignment must also be investigated.

Variable Line Spacing

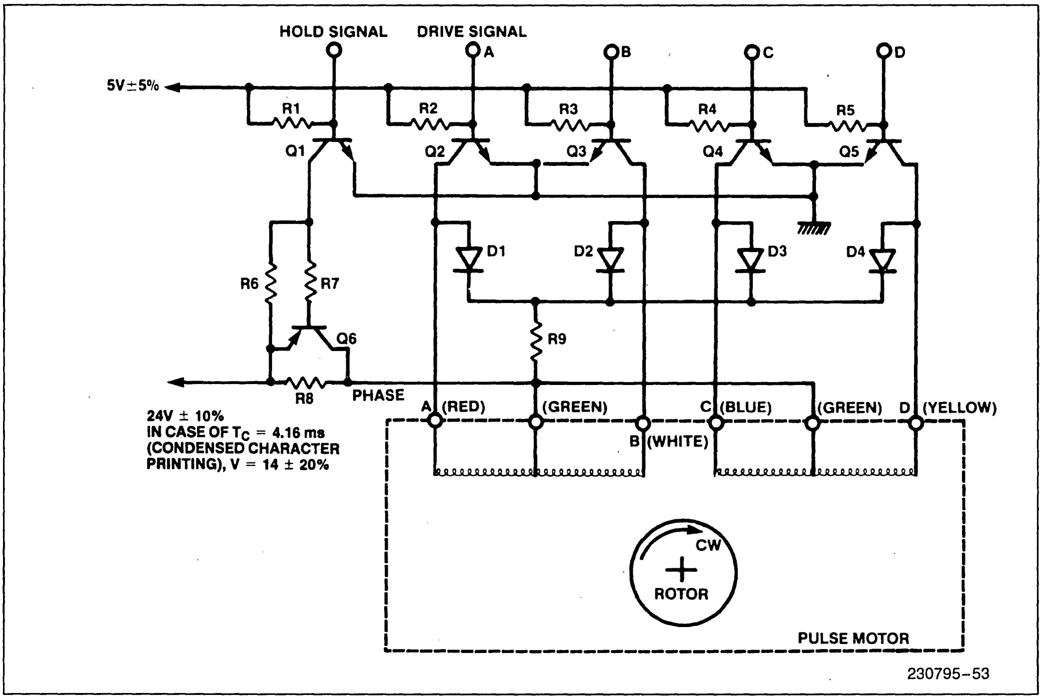
Variable line spacing is another feature which could be implemented either through the use of external status switches or inline control codes. The line spacing is a function of the number of steps the stepper motor rotates for a given line. Figure 15, Paper Feed Stepper Motor Predetermined Time Constants, in the Background section, lists the Time Constants required for three different line spacings; 6, 8, and 10 lines per inch. At the beginning of the Paper Feed Stepper Motor Drive subroutine, the default line step count is loaded. The software required is a conditional load for the line spacing, indicated by a status flag set in the External Status Switch Check subroutine or the Character Buffer Fill subroutine. Implementing the three different line spacings would require two additional status flags.

APPENDIX C PRINTER MECHANISM DRIVE CIRCUIT



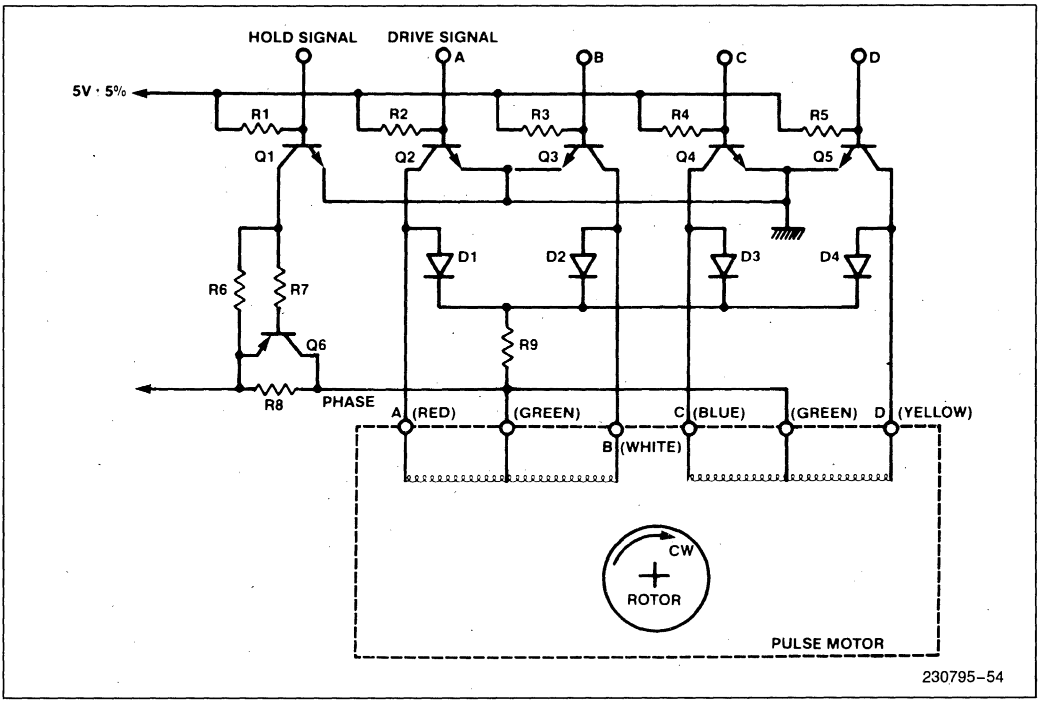
C1. Recommended Solenoid Drive Circuit

Parts No.		Type	Maker
IC1 ~ IC10		SN7406	TI
IC11		μA555	Fairchild
D1 ~ D9	Diode	S5277B	Toshiba
Q1 ~ 9	Transistor	2SD986	NEC
Q10	Transistor	2SA1015	Toshiba
Q11	Transistor	2SD633	Toshiba
R1 ~ R9	Resistor	1.2 kΩ ¼	
R10	Resistor	22Ω ¼	
R11	Resistor	580Ω 2	
R12	Resistor	15 kΩ ¼ Carbon fil =	
R13	Resistor	1.2 kΩ ¼	
VR1	Variable Resistor	20 kΩ ¼	
C1	Capacitor	1 μF 100V	
C2	Capacitor	0.01 μF	
C3	Capacitor	0.001 μF	
C4	Capacitor	10 μF 16V	
C5	Capacitor	0.1 μF fil =	
ZD1	Zenor Diode	HZ24	Hitachi
ZD2	Zenor Diode	HZ5C1	Hitachi



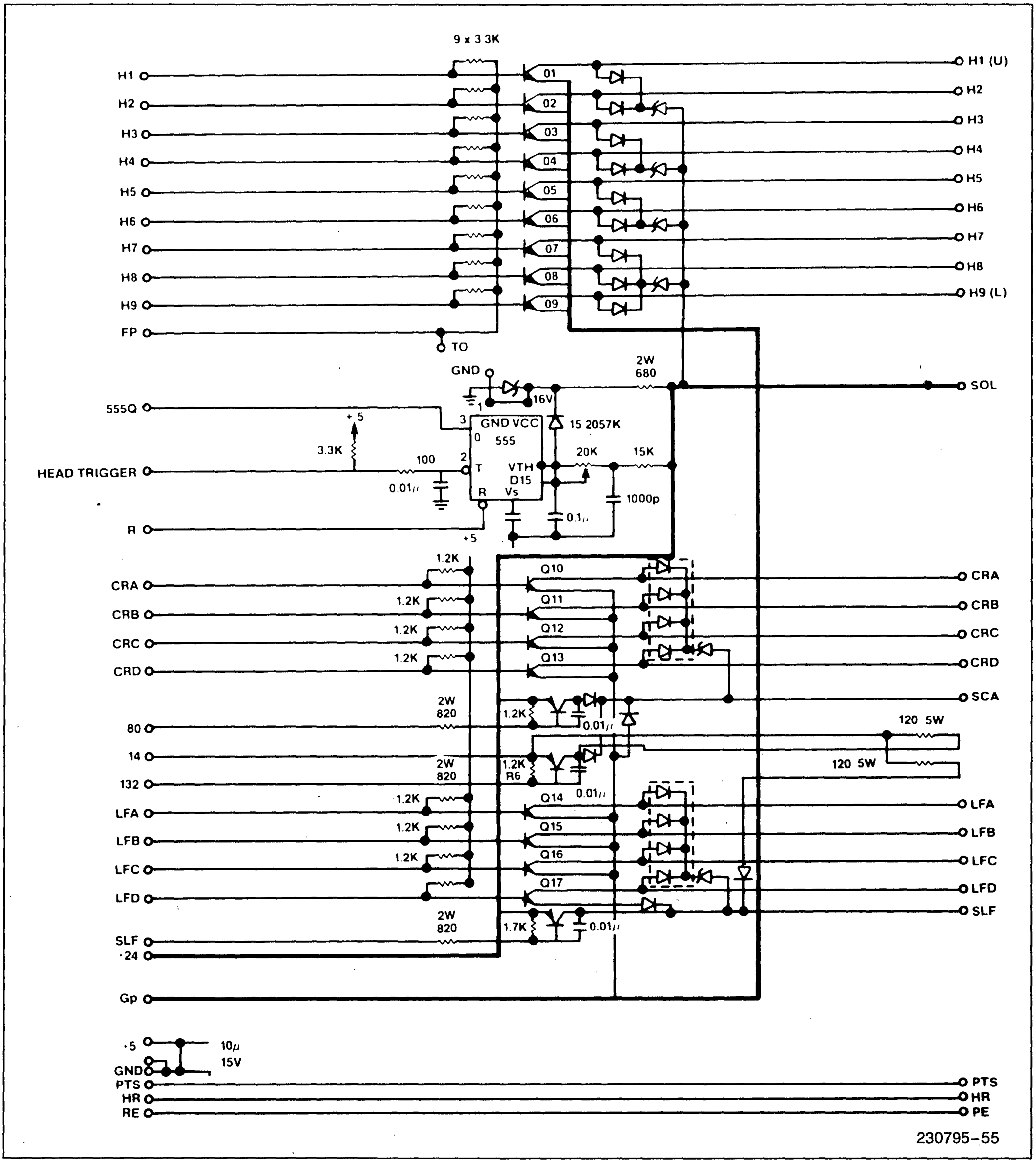
C2. Recommended Carriage Motor Drive Circuit

Parts No.		Type	Maker	Qty
R1	Resistor	1 kΩ ± 10% 1/4		1
R2-R5	Resistor	220Ω ± 10% 1/4		4
R6	Resistor	10 kΩ ± 10% 1/4		1
R7	Resistor	470Ω ± 10% 3		1
R8	Resistor	130Ω ± 10% 7		1
R9	Resistor	330Ω ± 10% 3		1
Q1	Transistor	2SC1815	Toshiba	1
Q2 ~ Q5	Transistor	2SD526-Y	Toshiba	4
Q6	Transistor	2SB669	Matsushita	1
D1 ~ D4	Diode	1S954	NEC	4



C3. Recommended Paper Feed Motor Drive Circuit

Parts No.		Type	Maker	Qty
R1	Resistor	1 kΩ ± 10% ¼		1
R2-R5	Resistor	220Ω ± 10% ¼		4
R6	Resistor	10 kΩ ± 10% ¼		1
R7	Resistor	470Ω ± 10% 3		1
R8	Resistor	130Ω ± 10% 7		1
R9	Resistor	330Ω ± 10% 3		1
Q1	Transistor	2SC1815	Toshiba	1
Q2 ~ Q5	Transistor	2SD526-Y	Toshiba	4
Q6	Transistor	2SB669	Matsushita	1
D1 ~ D4	Diode	1S954	NEC	4


9 x 3 3K
H1
H2
H3
H4
H5
H6
H7
H8
H9
FP
01
02
03
04
05
06
07
08
09
TO
H1 (U)
H2
H3
H4
H5
H6
H7
H8
H9 (L)
GND
2W
680
SOL
1
16V
555Q
3
GND VCC
0
555
15 2057K
+5
3.3K
100
2
20K
15K
HEAD TRIGGER
T
VTH
D15
R
Vs
0.01
1000p
0.1
R
+5
1.2K
Q10
Q11
Q12
Q13
CRA
CRB
CRC
CRD
SCA
2W
820
1.2K
0.01
80
14
132
120 5W
120 5W
1.2K
R6
0.01
Q14
Q15
Q16
Q17
LFA
LFB
LFC
LFD
SLF
1.7K
0.01
SLF
+24
Gp
+5
10μ
15V
GND
PTS
HR
RE
PTS
HR
PE

230795-55

C4

intel®

APPLICATION NOTE

AP-90

November 1986

An 8741AH/8041A Digital Cassette Controller

JOHN BEASTON, JIM KAHN
PERIPHERAL APPLICATIONS

Order Number: 231314-002

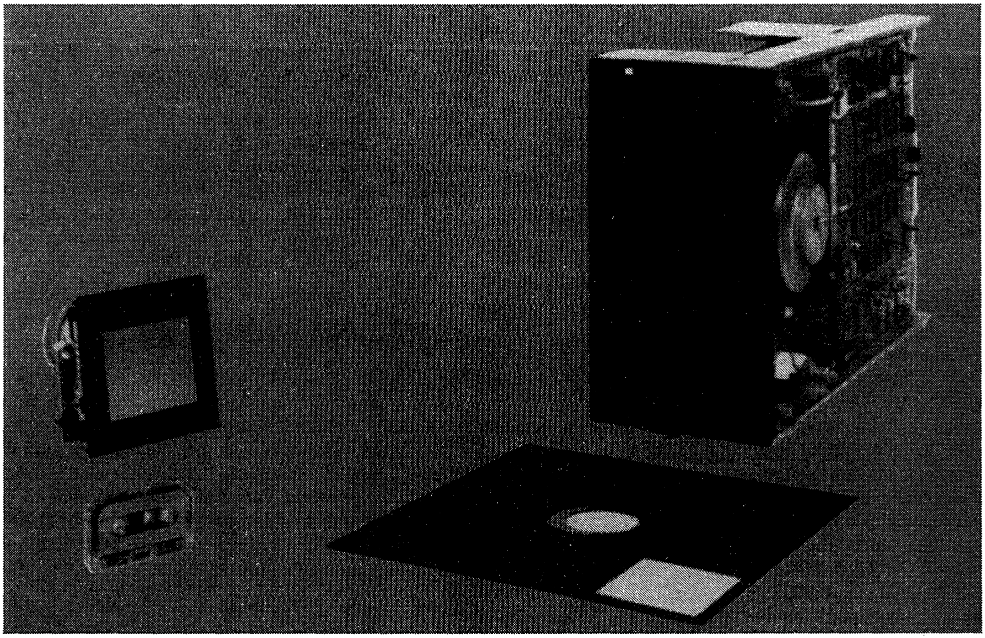
INTRODUCTION

The microcomputer system designer requiring a low-cost, non-volatile storage medium has a difficult choice. His options have been either relatively expensive, as with floppy discs and bubble memories, or non-transportable, like battery backed-up RAMs. The full-size digital cassette option was open but many times it was too expensive for the application. Filling this void of low-cost storage is the recently developed digital mini-cassette. These mini-cassettes are similar to, but not compatible with, dictation cassettes. The mini-cassette transports are inexpensive (well under $100 in quantity), small (less than 25 cu. in.), low-power (one watt), and their storage capacity is a respectable 200K bytes of unformatted data on a 100-foot tape. These characteristics make the mini-cassette perfect for applications ranging from remote datalogging to program storage for hobbyist systems.

The only problem associated with mini-cassette drives is controlling them. While these drives are relatively easy to interface to a microcomputer system, via a parallel I/O port, they can quickly overburden a CPU if other concurrent or critical real-time I/O is required. The cleanest and probably the least expensive solution in terms of development cost is to use a dedicated single-chip controller. However, a quick search through the literature turns up no controllers compatible with these new transports. What to do? Enter the UPI-41AH family of Universal Peripheral Interfaces.

The UPI-41AH family is a group of two user-programmable slave microcomputers plus a companion I/O expander. The 8741AH is the "flag-chip" of the line. It is a complete microcomputer with 1024 bytes of EPROM program memory, 64 bytes of RAM data memory, 16 individually programmable I/O lines, an 8-bit event counter and timer, and a complete slave peripheral interface with two interrupts and Direct Memory Access (DMA) control. The 8041A is the masked ROM, pin compatible version of the 8741AH. Figure 2 shows a block diagram common to both parts. The 8243 I/O port expander completes the family. Each 8243 provides 16 programmable I/O lines.

Using the UPI concept, the designer can develop a custom peripheral control processor for his particular I/O problem. The designer simply develops his peripheral control algorithm using the UPI-41AH assembly language and programs the EPROM of the 8741AH. Voila!



231314–1

Figure 1. Comparison of Mini-Cassette and Floppy Disk Transports and Media

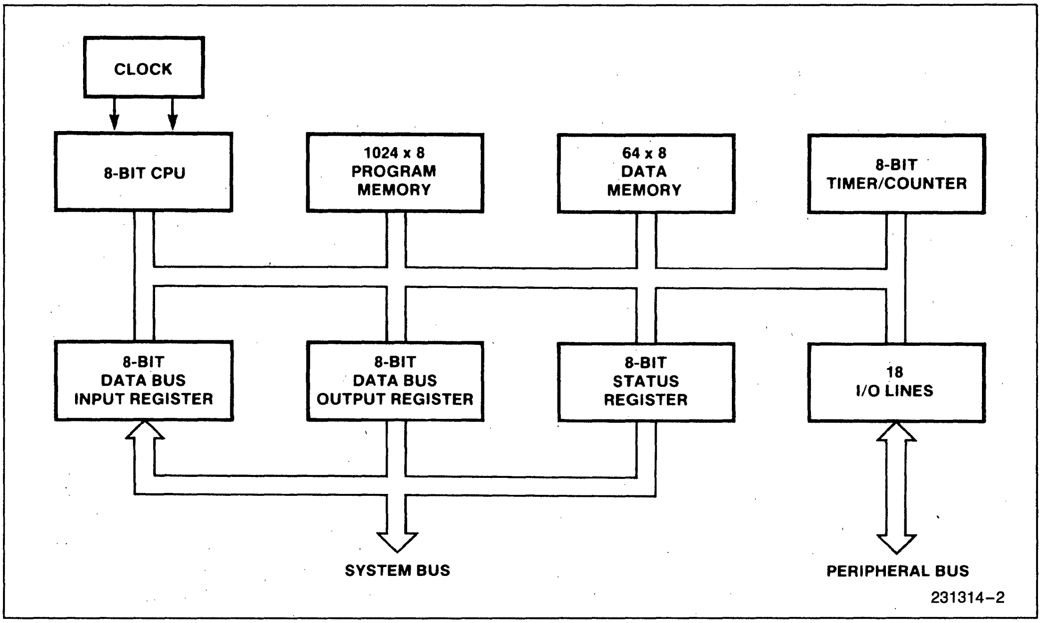


Figure 2. 8741AH/8041A Block Diagram

He has a single-chip dedicated controller. Testing may be accomplished using either an ICE-41A or the Single-Step mode of the 8741AH. UPI-41AH peripheral interfaces are being used to control printers, keyboards, displays, custom serial interfaces, and data encryption units. Of course, the UPI family is perfect for developing a dedicated controller for digital mini-cassette transports. To illustrate this application for the UPI family let's consider the job of controlling the Braemar CM-600 Mini-Dek*.

THE CM-600 MINI-DEK*

The Braemar CM-600 is representative of digital mini-cassette transports. It is a single-head, single-motor transport which operates entirely from a single 5V power supply. Its power requirements, including the motor, are 200 mA for read or write and 700 mA for rewind. Tape speeds are 3 inches per second (IPS) during read or write, 5 IPS fast forward, and 15 IPS rewind. With these speeds and a maximum recording density of 800 bits per inch (BPI), the maximum data rate is 2400 bits per second (BAUD). The data capacity using both sides of a 100-foot tape is 200K bytes. On top of this, the transport occupies only 22.5 cubic inches (3″x3″x2.5″).

All I/O for the CM-600 is TTL-compatible and can be divided into three groups: motor control, data control, and cassette status. The motor group controls are GO/STOP, FAST/SLOW, and FORWARD/REVERSE. The data controls are READ/WRITE, DATA IN, and DATA OUT. The remaining group of outputs give the transport's status: CLEAR LEADER, CASSETTE PRESENCE, FILE PROTECT, and SIDE SENSOR. These signals, shown schematically in Figure 3 and Table 1, give the pin definition of the CM-600 16-pin I/O connector.

RECORDING FORMAT

The CM-600 does not provide either encoding or decoding of the recorded data; that task is left for the peripheral interface. A multitude of encoding techniques from which the user may choose are available. In this single-chip dedicated controller application, a "self-clocking" phase encoding scheme similar to that used in floppy discs was chosen. This scheme specifies that a logic "0" is a bit cell with no transition; a cell with a transition is a logic "1."

*MiniDek is a registered trademark of Braemar.

Table 1. CM-600 I/O Pin Definition

Pin	I/O	Function
1	—	Index pin—not used
2	—	Signal ground
3	O	Cassette side (0—side B, 1—side A)
4	I	Data input (0—space, 1—mark)
5	O	Cassette presence (0—cassette, 1—no cassette)
6	I	Read/Write (0—read, 1—write)
7	O	File protect (0—tab present, 1—tab removed)
8	—	+5V motor power
9	—	Power ground
10	—	Chassis ground
11	I	Direction (0—forward, 1—rewind)
12	I	Speed (0—fast, 1—slow)
13	O	Data output (0—space, 1—mark)
14	O	Clear leader (0—clear leader, 1—off clear leader)
15	I	Motion (0—go, 1—stop)
16	—	+5V logic power

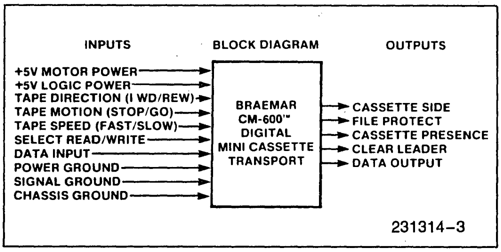


Figure 3. Braemar CM-600 Block Diagram

Figure 4 illustrates the encoding of the character 3AH assuming the previous data ended with the data line high. (The least significant bit is sent first.) Notice that there is always a "clocking" transition at the beginning of each cell. Decoding is simply a matter of triggering on this "clocking" transition, waiting $^3/_4$ of a bit cell time, and determining whether a mid-cell transition has occurred. Cells with no mid-cell transitions are data 0's; cells with transitions are data 1's. This encoding technique has all the benefits of Manchester encoding with the added advantage that the encoded data may be "decoded by eyeball:" long cells are always 0's, short cells are always 1's.

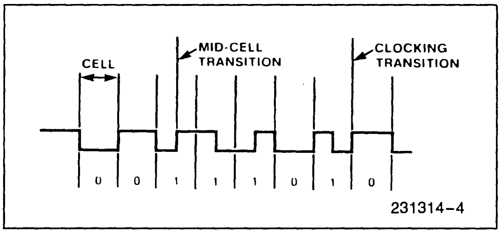


Figure 4. Modified Phase Encoding of Character 3A Hex

Besides the encoding scheme, the data format is also up to the user. This controller uses a variable byte length, checksum protected block format. Every block starts and ends with a SYNC character (AAH), and the character immediately preceding the last SYNC is the checksum. The checksum is capable of catching 2 bit errors. The number of data characters within a block is limited to 64K bytes. Blocks are separated by an Inter-Record Gap (IRG). The IRG is of such a length that the transport can stop and start within an IRG, as illustrated in the data block timing, Figure 5. Braemar specifies a maximum start or stop time of 150 ms for the transport, thus the controller uses 450 ms for the IRG. This gives plenty of margin for controlling the transport and also for detecting IRGs while skipping blocks.

THE UPI-41AH™ CONTROLLER

The goal of the UPI software design for this application was to make the UPI-41AH microcomputer into an intelligent cassette control processor. The host processor (8086, 8088, 8085A, etc.) simply issues a high-level command such as READ-a-block or WRITE-a-block. The 8741AH accepts the command, performs the requested operation, and returns to the host system a result code telling the outcome of the operation, eg. Good-Completion, Sync Error, etc. Table 2 shows the command and result code repertoire. The 8741AH completely manages all the data transfers for reading and writing.

As an example, consider the WRITE-a-block command. When this command is issued, the UPI-41AH expects a 16-bit number from the host telling how many data bytes to write (up to 64K bytes per block). Once this number is supplied in the form of two bytes, the host is free to perform other tasks; a bit in the UPI's STATUS register or an interrupt output will notify the host when a data transfer is required. The 8741AH then checks the transport's status to be sure that a cassette is present and not file protected. If either is false, a result code is returned to the host; otherwise the transport is started. After the peripheral controller checks to make sure that the tape is off the clear leader and past the hole in the tape, it writes a 450 ms IRG, a SYNC character, the block of data, the checksum, and the final SYNC character. (The tape has a clear leader at both ends and a small hole 6 inches from the end of each leader.) The data transfers from the host to the

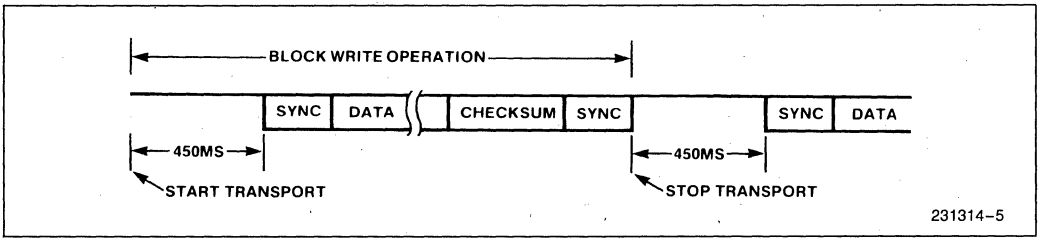


Figure 5. IRG/Block Timing Diagram (not to scale)

Table 2. Controller Command/Result Code Set

Command	Result
Read (01H)	Good-Completion (00H) Buffer Overrun Error (41H) Bad Synch1 Error (42H) Bad Synch2 Error (43H) Checksum Error (44H) Command Error (45H) End of Tape Error (46H)
Rewind (04H)	Good-Completion (00H)
Skip (03H)	Good-Completion (00H) End of Tape Error (47H) Beginning of Tape Error (48H)
Write (02H)	Good-Completion (00H) Buffer Underrun Error (81H) Command Error (82H) End of Tape Error (83H)

UPI-41AH slave microcomputer are double buffered. The controller requests only the desired number of data bytes by keeping track of the count internally.

If nothing unusual happened, such as finding clear leader while writing, it returns a Good-Completion result code to the host. If clear leader was encountered, the transport is stopped immediately and an End-of-Tape result code is returned to the host. Another possible error would be if the host is late in supplying data. If this occurs, the controller writes an IRG, stops the drive, and returns the appropriate Data-Underrun result code.

The READ-a-block command also provides error checking. Once this command is issued by the host, the controller checks for cassette presence. If present, it starts the transport. The data output from the transport is then examined and decoded continuously. If the first character is not a SYNC, that's an error and the controller returns a Bad-First-SYNC result code (42H) after advancing to the next IRG. If the SYNC is good, the succeeding characters are read into an on-chip 30 character circular buffer. This continues until an IRG is encountered. When this occurs, the transport is stopped. The controller then tests that it is the last character. If it is a SYNC, the controller then compares the accumulated internal checksum to the block's checksum, the second to the last character of the block. If they match, a Good-Completion result code (00H) is returned to the host. If either test is bad, the appropriate error result code is returned. The READ command also checks for the End-of-Tape (EOT) clear leader and returns the appropriate error result code if it is found before the read operation is complete.

The 30 character circular buffer allows the host up to 30 character times of response time before the host must collect the data. All data transfers take place thru the UPI-41AH Data Bus Buffer Output register (DBBOUT). The controller continually monitors the status of this register and moves characters from the circular buffer to the register whenever it is empty.

The SKIP-n-blocks command allows the host to skip the transport forward or backward up to 127 blocks. Once the command is issued, the controller expects one data byte specifying the number of blocks to skip. The most significant bit of this byte selects the direction of the skip (0 = forward, 1 = reverse). SKIP is a dual-speed operation in the forward direction. If the number of blocks to skip is greater than 8, the controller uses fast-forward (5 IPS until it is within 8 blocks of the desired location. Once within 8 blocks, the controller switches to the normal read speed (3 IPS) to allow accurate placement of the tape. The reverse skip uses only the rewind speed (15 IPS). Like the READ and WRITE commands, SKIP also checks for EOT and beginning-of-tape (BOT) depending upon the tape's direction. An error result code is returned if either is encountered before the number of blocks skipped is complete.

The REWIND command simply rewinds the tape to the BOT clear leader. The ABORT command allows the termination of any operation in progress, except a REWIND. All commands, including ABORT, always leave the tape positioned on an IRG.

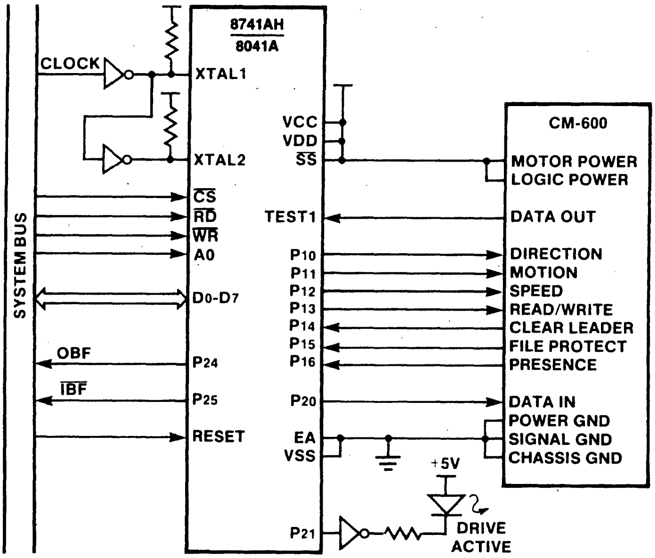
THE HARDWARE INTERFACE

There's hardly any hardware design effort required for the controller and transport interface in Figure 6. Since the CM-600 is TTL compatible, it connects directly to the I/O ports of the UPI controller. If the two are separated (i.e. on different PC cards), it is recommended that TTL buffers be provided. The only external circuitry needed is an LED driver for the DRIVE ACTIVE status indicator.

The 8741AH-to-host interface is equally straightforward. It has a standard asynchronous peripheral interface: 8 data lines (D_0–D_7), read (RD), write (WR), register select (AO), and chip select (CS). Thus it connects directly to an 8086, 8088, 8085A, 8080, or 8048 bus structure. Two interrupt outputs are provided for data transfer requests if the particular system is interrupt-driven. DMA transfer capability is also available. The clock input can be driven from a crystal directly or with the system clock (6 MHz max). The UPI-41AH clock may be asynchronous with respect to other clocks within the system.

This application was developed on an Intel iSBC 80/30 single board computer. The iSBC 80/30 is controlled by an 8085A microprocessor, contains 16K bytes of dual-ported dynamic RAM and up to 8K bytes of either EPROM or ROM. Its I/O complement consists of an 8255A Programmable Parallel Interface, an 8251A Programmable Communications Interface, an 8253 Programmable Interval Timer, and an 8259A Programmable Interrupt Controller. The iSBC 80/30 is especially convenient for UPI development since it contains an uncommitted socket dedicated to either an 8041A or 8741AH, complete with buffering for its I/O ports. The iSBC 80/30 to 8741AH interface is reflected in Figure 8. (Optionally, an iSBC 569 Digital Controller board could be used. The iSBC 569 board contains three uncommitted UPI sockets with an interface similar to that in Figure 8.)

Looking at the host-to-controller interface, the host sees the 8741AH as three registers in the host's I/O address space: the data register, the command register, and the status register. The decoding of these registers is shown in Figure 7. All data and commands for the controller are written into the Data Bus Buffer Input register (DBBIN). The state of the register select input, AO, determines whether a command or data is written. (Writes with AO set to 1 are commands by convention.) All data and results from the controller are read by the host from the Data Bus Buffer Output register (DBBOUT).

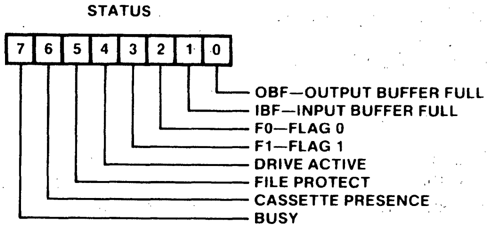


231314-6

Figure 6. Controller/Transport System Schematic

$\overline{\text{CS}}$	$\overline{\text{RD}}$	$\overline{\text{WR}}$	A0	Register
O	O	I	O	DBBOUT
O	O	I	I	STATUS
O	I	O	O	DBBIN (DATA)
O	I	O	I	DBBIN (COMMAND)
I	X	X	X	NONE

Figure 7. 8741AH/8041A Interface Register Decoding



231314-7

Figure 8. Status Register Bit Definition

The Status register contains flags which give the host the status of various operations within the controller. Its format is given in Figure 8. The Input Buffer Full ($\overline{\text{IBF}}$) and Output Buffer Full (OBF) flags show the Status of the DBBIN and DBBOUT registers respectively. $\overline{\text{IBF}}$ indicates when the DBBIN register contains data written by the host. The host may write to DBBIN only when $\overline{\text{IBF}}$ is 0. Likewise, the host may read DBBOUT only when OBF is set to a 1. These bits are handled automatically by the UPI-41AH internal hardware. FLAG 0 (F_0) and FLAG 1 (F_1) are general purpose flags used internally by the controller which have no meaning externally.

The remaining four bits are user-definable. For this application they are DRIVE ACTIVE, FILE PROTECT, CASSETTE PRESENCE, and BUSY flags. The FILE PROTECT and CASSETTE PRESENCE flags reflect the state of the corresponding I/O lines from the transport. DRIVE ACTIVE is set whenever the transport motor is on and the controller is performing an operation. The BUSY flag indicates whether the contents of the DBBOUT register is data or a result code. The BUSY flag is set whenever a command is issued by the host and accepted by the controller. As long as BUSY is set, any character found in DBBOUT is a result code. Thus whenever the host finds OBF set, it should test the BUSY flag to determine whether the character is data or a result code.

Notice the OBF and $\overline{\text{IBF}}$ are available as interrupt outputs to the host processor, Figure 6. These outputs are self-clearing, that is, OBF is set automatically upon the controller loading DBBOUT and cleared automatically by the host reading DBBOUT. Likewise $\overline{\text{IBF}}$ is cleared to a 0 by the host writing into DBBIN: set to a 1 when the controller reads DBBIN into the accumulator.

The flow charts of Figure 9 show the flow of sample host software assuming a polling software interface between the host and the controller. The WRITE command requires two additional count bytes which form the 16-bit byte count. These extra bytes are "handshaked" into the controller using the IBF flag in the STATUS register. Once these bytes are written, the host writes data in response to IBF being cleared. This continues until the host finds OBF set indicating that the operation is complete and reads the result code from DBBOUT. No testing of BUSY is needed since only the result code appears in the DBBOUT register.

The READ command does require that BUSY be tested. Once the READ command is written into the controller, the host must test BUSY whenever OBF is set to determine whether the contents of DBBOUT is data from the tape or the result code.

THE CONTROLLER SOFTWARE

The UPI-41AH software to control the cassette can be divided up into various commands such as WRITE, READ and ABORT. In a previous version of this application note (May 1980), software was described that implemented these commands. This code however did not adequately compensate for speed variations of the motor during record and playback nor for data distortion caused by the magnetic media. Since then, a new code has been written to include these effects. This revised software is now available through the INTEL User's Library, INSITE. For more information on this software or INSITE, contact your local INTEL Sales Office.

July 1986

UPI-452 Accelerates iAPX 286 Bus Performance

CHRISTOPHER SCOTT
TECHNICAL MARKETING ENGINEER
INTEL CORPORATION

Order Number: 292018-001

INTRODUCTION

The UPI-452 targets the leading problem in peripheral to host interfacing, the interface of a slow peripheral with a fast Host or "bus utilization". The solution is data buffering to reduce the delay and overhead of transferring data between the Host microprocessor and I/O subsystem. The Intel CMOS UPI-452 solves this problem by combining a sophisticated programmable FIFO buffer and a slave interface with an MSC-51 based microcontroller.

The UPI-452 is Intel's newest Universal Peripheral Interface family member. The UPI-452 FIFO buffer enables Host—peripheral communications to be through streams or bursts of data rather than by individual bytes. In addition the FIFO provides a means of embedding commands within a stream or block of data. This enables the system designer to manage data and commands to further off-load the Host.

The UPI-452 interfaces to the iAPX 286 microprocessor as a standard Intel slave peripheral device. READ, WRITE, CS and address lines from the Host are used to access all of the Host addressable UPI-452 Special Function Registers (SFR).

The UPI-452 combines an MSC-51 microcontroller, with 256 bytes of on-chip RAM and 8K bytes of EPROM/ROM, twice that of the 80C51, a two channel DMA controller and a sophisticated 128 byte, two channel, bidirectional FIFO in a single device. The UPI-452 retains all of the 80C51 architecture, and is fully compatible with the MSC-51 instruction set.

This application note is a description of an iAPX 286 to UPI-452 slave interface. Included is a discussion of the respective timings and design considerations. This application note is meant as a supplement to the UPI-452 Advance Data Sheet. The user should consult the data sheet for additional details on the various UPI-452 functions and features.

UPI-452 iAPX 286 SYSTEM CONFIGURATION

The interface described in this application note is shown in Figure 1, iAPX 286 UPI-452 System Block Diagram. The iAPX 286 system is configured in a local bus architecture design. DMA between the Host and the UPI-452 is supported by the 82258 Advanced DMA Controller. The Host microprocessor accesses all UPI-452 externally addressable registers through address decoding (see Table 3, UPI-452 External Address Decoding). The timings and interface descriptions below are given in equation form with examples of specific calculations. The goal of this application note is a set of interface analysis equations. These equations are the tools a system designer can use to fully utilize the features of the UPI-452 to achieve maximum system performance.

HOST-UPI-452 FIFO SLAVE INTERFACE

The UPI-452 FIFO acts as a buffer between the external Host 80286 and the internal CPU. The FIFO allows the Host - peripheral interface to achieve maximum decoupling of the interface. Each of the two FIFO channels is fully user programmable. The FIFO buffer ensures that the respective CPU, Host or internal CPU, receives data in the same order as transmitted. Three slave bus interface handshake methods are supported by the UPI-452; DMA, Interrupt and Polled.

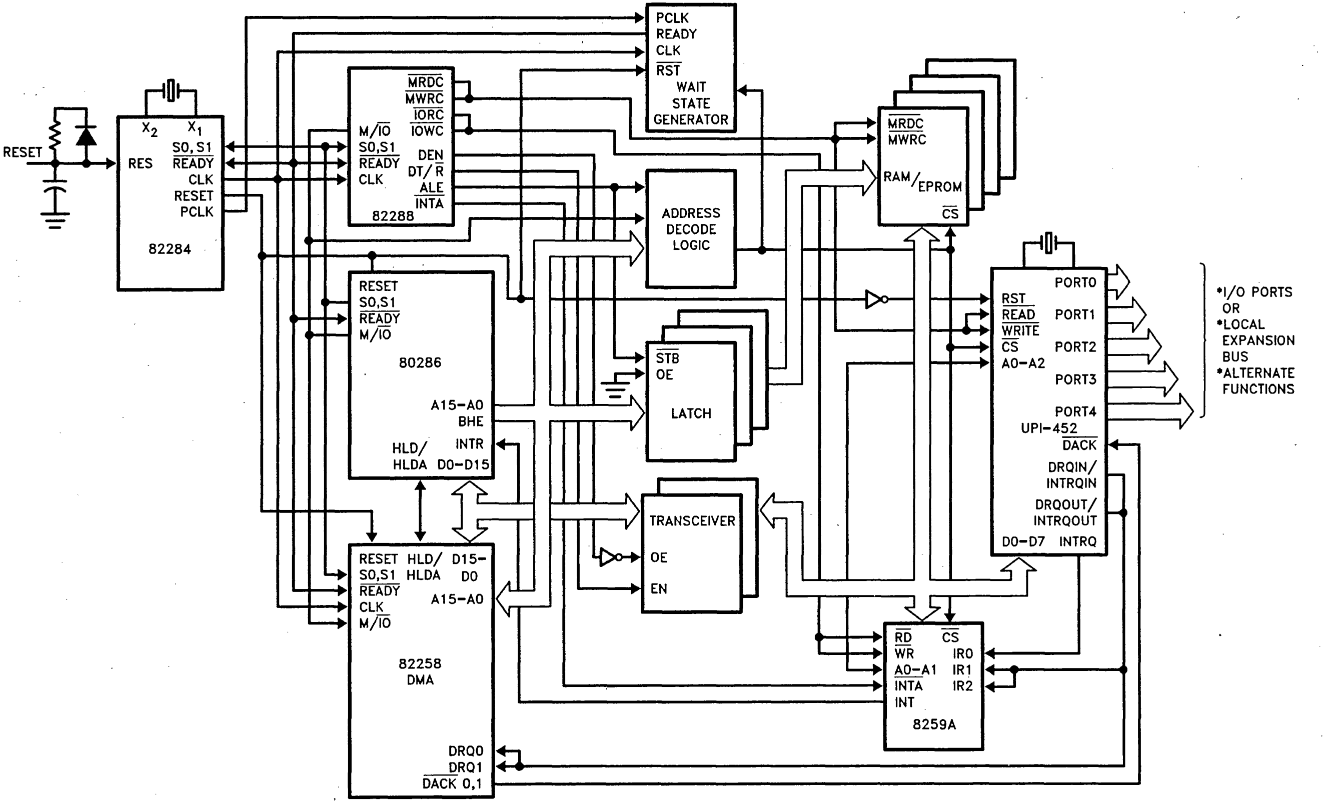
The interface between the Host 80286 and the UPI-452 is accomplished with a minimum of signals. The 8 bit data bus plus READ, WRITE, CS, and A0-2 provide access to all of the externally addressable UPI-452 registers including the two FIFO channels. Interrupt and DMA handshaking pins are tied directly to the interrupt controller and DMA controller respectively.

DMA transfers between the Host and UPI-452 are controlled by the Host processors DMA controller. In the example shown in Figure 1, the Host DMA controller is the 82258 Advanced DMA Controller. An internal DMA transfer to or from the FIFO, as well as between other internal elements, is controlled by the UPI-452 internal DMA processor. The internal DMA processor can also transfer data between Input and Output FIFO channels directly. The description that follows details the UPI-452 interface from both the Host processor's and the UPI-452's internal CPU perspective.

One of the unique features of the UPI-452 FIFO is its ability to distinguish between commands and data embedded in the same data block. Both interrupts and status flags are provided to support this operation in either direction of data transfer. These flags and interrupts are triggered by the FIFO logic independent of, and transparent to either the Host or internal CPUs. Commands embedded in the data block, or stream, are called Data Stream Commands.

Programmable FIFO channel Thresholds are another unique feature of the UPI-452. The Thresholds provide for interrupting the Host only when the Threshold number of bytes can be read or written to the FIFO buffer. This further decouples the Host UPI-452 interface by relieving the Host of polling the buffer to determine the number of bytes that can be read or written. It also reduces the chances of overrun and underrun errors which must be processed.

The UPI-452 also provides a means of bypassing the FIFO, in both directions, for an immediate interrupt of either the Host or internal CPU. These commands are called Immediate Commands. A complete description of the internal FIFO logic operation is given in the FIFO Data Structure section.



292018–1

Figure 1. iAPX 286 UPI-452 System Block Diagram

UPI-452 INITIALIZATION

The UPI-452 at power-on reset automatically performs a minimum initialization of itself. The UPI-452 notifies the Host that it is in the process of initialization by setting a Host Status SFR bit. The user UPI-452 program must release the UPI-452 from initialization for the FIFO to be accessible by the Host. This is the minimum Host to UPI-452 initialization sequence. All further initialization and configuration of the UPI-452, including the FIFO, is done by the internal CPU under user program control. No interaction or programming is required by the Host 80286 for UPI-452 initialization.

At power-on reset the UPI-452 automatically enters FIFO DMA Freeze Mode by resetting the Slave Control (SLCON) SFR FIFO DMA Freeze/Normal Mode bit to FIFO DMA Freeze Mode (FRZ = "0"). This forces the Slave Status (SSTAT) and Host Status (HSTAT) SFR FIFO DMA Freeze/Normal Mode bits to FIFO DMA Freeze Mode In Progress. FIFO DMA Freeze Mode allows the FIFO interface to be configured, by the internal CPU, while inhibiting Host access to the FIFO.

The MODE SFR is forced to zero at reset. This disables, (tri-states) the DRQIN/INTRQIN, DRQOUT/INTRQOUT and INTRQ output pins. INTRQ is inhibited from going active to reflect the fact that a Host Status SFR bit, FIFO DMA Freeze Mode, is active. If the MODE SFR INTRQ configure bit is enabled (='1'), before the Slave Control and Host Status SFR FIFO DMA Freeze/Normal Mode bit is set to Normal Mode, INTRQ will go active immediately.

The first action by the Host following reset is to read the UPI-452 Host Status SFR Freeze/Normal Mode bit to determine the status of the interface. This may be done in response to a UPI-452 INTRQ interrupt, or by polling the Host Status SFR. Reading the Host Status SFR resets the INTRQ line low.

Any of the five FIFO interface SFRs, as well as a variety of additional features, may be programmed by the internal CPU following reset. At power-on reset, the five FIFO Special Function Registers are set to their default values as listed in Table 1. All reserved location bits are set to one, all other bits are set to zero in these three SFRs. The FIFO SFRs listed in Table 1 can be programmed only while the UPI-452 is in FIFO DMA Freeze Mode. The balance of the UPI-452 SFRs default values and descriptions are listed in the UPI-452 Advance Data Sheet in the Intel Microsystems Component Handbook Volume II and Microcontroller Handbook.

The above sequence is the minimum UPI-452 internal initialization required. The last initialization instruction must always set the UPI-452 to Normal Mode. This causes the UPI-452 to exit Freeze Mode and enables Host read/write access of the FIFO. The internal CPU sets the Slave Control (SLCON) SFR FIFO DMA Freeze/Normal Mode (FRZ) bit high (= 1) to activate Normal Mode. Ths causes the Slave Status (SSTAT) and Host Status (HSTAT) SFR FIFO DMA Freeze Mode bits to be set to Normal Mode. Table 2, UPI-452 Initialization Event Sequence Example, shows a summary of the initialization events described above.

Table 1. FIFO Special Function Register Default Values

SFR Name	Label	Reset Value
Channel Boundary Pointer	CBP	40H/64D
Output Channel Read Pointer	ORPR	40H/64D
Output Channel Write Pointer	OWPR	40H/64D
Input Channel Read Pointer	IRPR	00H/0D
Input Channel Write Pointer	IWPR	00H/0D
Input Threshold	ITH	00H/0D
Output Threshold	OTH	01H/1D

Table 2. UPI-452 Initialization Event Sequence Example

Event Description	SFR/bit
Power-on Reset	
UPI-452 forces FIFO DMA Freeze Mode (Host access to FIFO inhibited)	SLCON FRZ = 0
UPI-452 forces Slave Status and Host Status SFR to FIFO DMA Freeze Mode In Progress	SSTAT SST5 = 0 HSTAT HST1 = 1
UPI-452 forces all SFRs, including FIFO SFRs, to default values.	
* UPI-452 user program enables INTRQ, INTRQ goes active, high	MODE MD4 = 1
* Host $\overline{\text{READ}}$'s UPI-452 Host Status (HSTAT) SFR to determine interrupt source, INTRQ goes low	
* UPI-452 user program initializes any other SFRs; FIFO, Interrupts, Timers/Counters, etc.	
User program sets Slave Control SFR to Normal Mode (Host access to FIFO enabled)	SLCON FRZ = 1
UPI-452 forces Slave and Host Status SFRs bits to Normal Operation	SSTAT SST5 = 1 HSTAT HST1 = 0
* Host polls Host Status SFR to determine when it can access the FIFO - or - * Host waits for UPI-452 Request for Service interrupt to access FIFO	

* user option

FIFO DATA STRUCTURES

Overview

The UPI-452 provides three means of communication between the Host microprocessor and the UPI-452 in either direction;

Data
Data Stream Commands
Immediate Commands

Data and Data Stream Commands (DSC) are transferred between the Host and UPI-452 through the UPI-452 FIFO buffer. The third, Immediate Commands, provides a means of bypassing the FIFO entirely. These three data types are in addition to direct access by either Host or Internal CPU of dedicated Status and Control Special Function Registers (SFR).

The FIFO appears to both the Host 80286 and the internal CPU as 8 bits wide. Internally the FIFO is logically nine bits wide. The ninth bit indicates whether the byte is a data or a Data Stream Command (DSC) byte; 0 = data, 1 = DSC. The ninth bit is set by the FIFO logic in response to the address specified when writing to the FIFO by either Host or internal CPU. The FIFO uses the ninth bit to condition the UPI-452 interrupts and status flags as a byte is made available for a Host or internal CPU read from the FIFO. Figures 2 and 3 show the structure of each FIFO channel and the logical ninth bit.

It is important to note that both data and DSCs are actually entered into the FIFO buffer (see Figures 2 and 3). External addressing of the FIFO determines the state of the internal FIFO logic ninth bit. Table 3 shows the UPI-452 External Address Decoding used by the Host and the corresponding action. The internal CPU interface to the FIFO is essentially identical to the external Host interface. Dedicated internal Special Function Registers provide the interface between the FIFO, internal CPU and the internal two channel DMA processor. FIFO read and write operations by the Host and internal CPU are interleaved by the UPI-452 so they appear to be occurring simultaneously.

The ninth bit provides a means of supporting two data types within the FIFO buffer. This feature enables the Host and UPI-452 to transfer both commands and data while maintaining the decoupled interface a FIFO buffer provides. The logical ninth bit provides both a means of embedding commands within a block of data and a means for the internal CPU, or external Host, to discriminate between data and commands. Data or DSCs may be written in any order desired. Data Stream Commands can be used to structure or dispatch the data by defining the start and end of data blocks or packets, or how the data following a DSC is to be processed.

A Data Stream Command (DSC) acts as an internal service routine vector. The DSC generates an interrupt to a service routine which reads the DSC. The DSC byte acts as an address vector to a user defined service routine. The address can be any program or data memory location with no restriction on the number of DSCs or address boundaries.

A Data Stream Command (DSC) can also be used to clear data from the FIFO or "FLUSH" the FIFO. This is done by appending a DSC to the end of a block of data entered in the FIFO which is less than the programmed threshold number of bytes. The DSC will cause an interrupt, if enabled, to the respective receiving CPU. This ensures that a less than Threshold number of bytes in the FIFO will be read. Two conditions force a Request for Service interrupt, if enabled, to the Host. The first is due to a Threshold number of bytes having been written to the FIFO Output channel; the second is if a DSC is written to the Output FIFO channel. If less than the Threshold number of bytes are written to the Output FIFO channel, the Host Status SFR flag will not be set, and a Request for Service interrupt will not be generated, if enabled. By appending a DSC to end of the data block, the FIFO Request for Service flag and/or interrupt will be generated.

An example of a FIFO Flush application is a mass storage subsystem. The UPI-452 provides the system interface to a subsystem which supports tape and disk storage. The FIFO size is dynamically changed to provide the maximum buffer size for the direction of transfer. Large data blocks are the norm in this application. The FIFO Flush provides a means of purging the FIFO of the last bytes of a transfer. This guarantees that the block, no matter what its size, will be transmitted out of the FIFO.

Immediate Commands allow more direct communication between the Host processor and the UPI-452 by bypassing the FIFO in either direction. The Immediate Command IN and OUT SFRs are two more unique address locations externally and internally addressable. Both DSCs and Immediate Commands have internal interrupts and interrupt priorities associated with their operation. The interrupts are enabled or disabled by setting corresponding bits in the Slave Control (SLCON), Interrupt Enable (IEC), Interrupt Priority (IPC) and Interrupt Enable and Priority (IEP) SFRs. A detailed description of each of these may be found in the UPI-452 Advance Information Data Sheet.

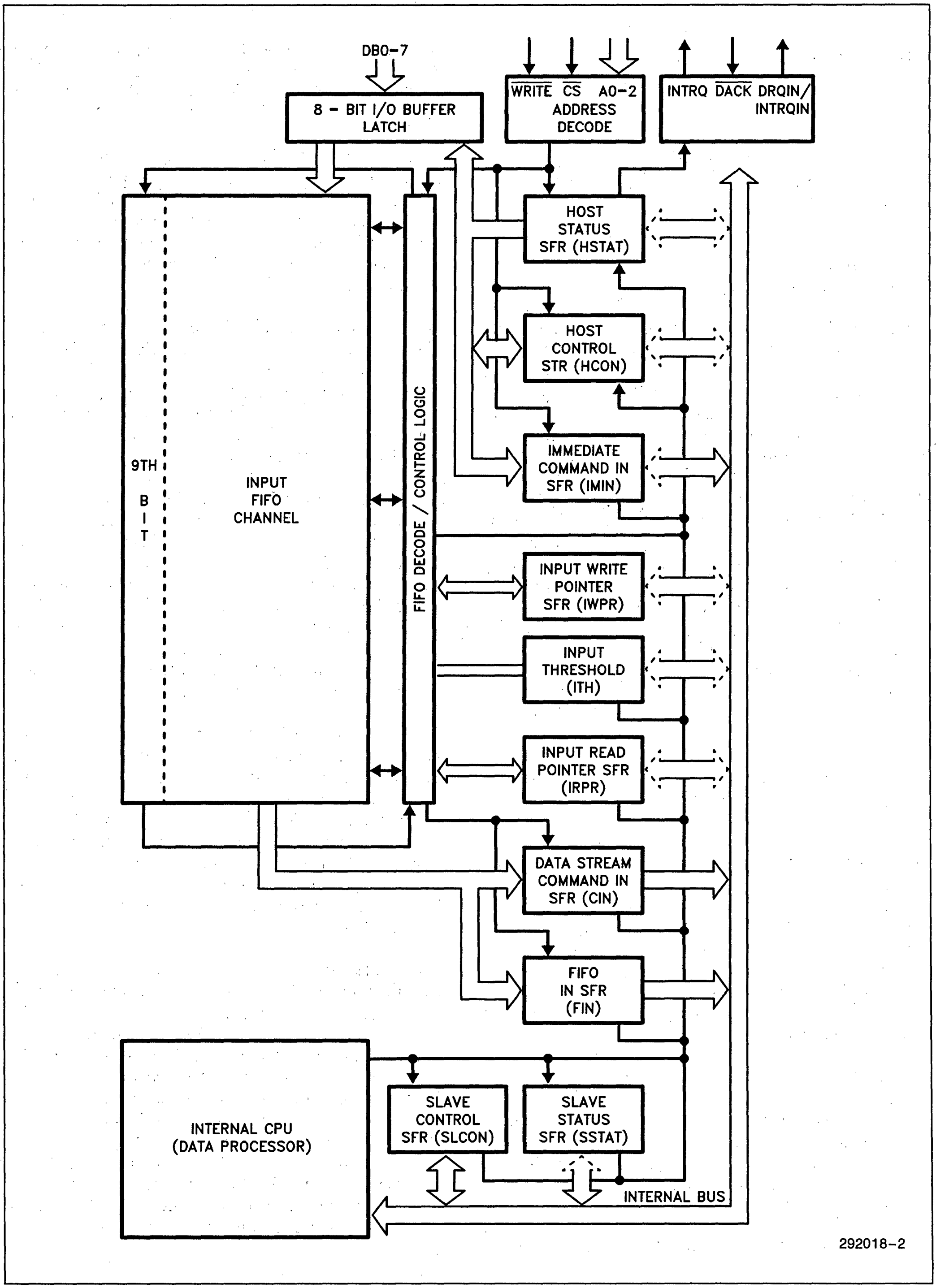

DB0-7
8 - BIT I/O BUFFER LATCH
WRITE CS A0-2 ADDRESS DECODE
INTRQ DACK DRQIN/ INTRQIN
HOST STATUS SFR (HSTAT)
HOST CONTROL STR (HCON)
IMMEDIATE COMMAND IN SFR (IMIN)
INPUT WRITE POINTER SFR (IWPR)
INPUT THRESHOLD (ITH)
INPUT READ POINTER SFR (IRPR)
DATA STREAM COMMAND IN SFR (CIN)
FIFO IN SFR (FIN)
9TH BIT
INPUT FIFO CHANNEL
FIFO DECODE / CONTROL LOGIC
INTERNAL CPU (DATA PROCESSOR)
SLAVE CONTROL SFR (SLCON)
SLAVE STATUS SFR (SSTAT)
INTERNAL BUS
292018-2

Figure 2. Input FIFO Channel Functional Diagram

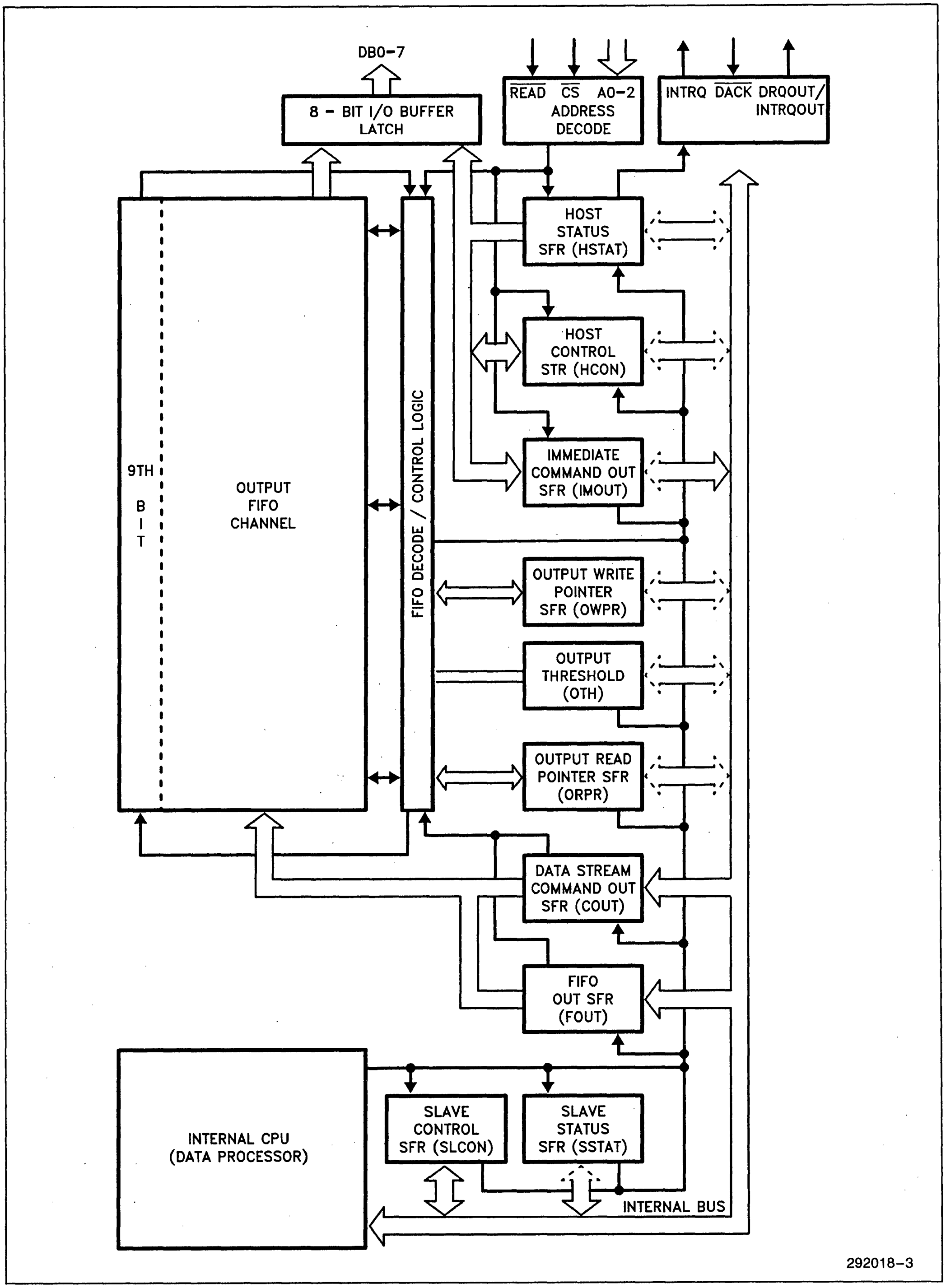

DB0–7
8 – BIT I/O BUFFER LATCH
READ CS A0–2 ADDRESS DECODE
INTRQ DACK DRQOUT/ INTRQOUT
HOST STATUS SFR (HSTAT)
HOST CONTROL STR (HCON)
IMMEDIATE COMMAND OUT SFR (IMOUT)
OUTPUT WRITE POINTER SFR (OWPR)
OUTPUT THRESHOLD (OTH)
OUTPUT READ POINTER SFR (ORPR)
DATA STREAM COMMAND OUT SFR (COUT)
FIFO OUT SFR (FOUT)
9TH B I T
OUTPUT FIFO CHANNEL
FIFO DECODE / CONTROL LOGIC
INTERNAL CPU (DATA PROCESSOR)
SLAVE CONTROL SFR (SLCON)
SLAVE STATUS SFR (SSTAT)
INTERNAL BUS
292018–3

Figure 3. Output FIFO Channel Functional Diagram

Table 3. UPI-452 External Address Decoding

$\overline{\text{DACK}}$	$\overline{\text{CS}}$	A2	A1	A0	$\overline{\text{READ}}$	$\overline{\text{WRITE}}$
1	1	X	X	X	No Operation	No Operation
1	0	0	0	0	Data or DMA from Output FIFO Channel	Data or DMA to Input FIFO Channel
1	0	0	0	1	Data Stream Command from Output FIFO Channel	Data Stream Command to Input FIFO Channel
1	0	0	1	0	Host Status SFR Read	Reserved
1	0	0	1	1	Host Control SFR Read	Host Control SFR Write
1	0	1	0	0	Immediate Command SFR Read	Immediate Command SFR Write
1	0	1	1	X	Reserved	Reserved
0	X	X	X	X	DMA Data from Output FIFO Channel	DMA Data to Input FIFO Channel

Below is a detailed description of each FIFO channel's operation, including the FIFO logic response to the ninth bit, as a byte moves through the channel. The description covers each of the three data types for each channel. The details below provide a picture of the various FIFO features and operation. By understanding the FIFO structure and operation the user can optimize the interface to meet the requirements of an individual design.

OUTPUT CHANNEL

This section covers the data path from the internal CPU to the HOST. Data Stream Command or Immediate Command processing during Host DMA Operations is covered in the DMA section.

UPI-452 Internal Write to the FIFO

The internal CPU writes data and Data Stream Commands into the FIFO through the FIFO OUT (FOUT) and Command OUT (COUT) SFRs. When a Threshold number of bytes has been written, the Host Status SFR Output FIFO Request for Service bit is set and an interrupt, if enabled, is generated to the Host. Either the INTRQ or DRQOUT/INTRQOUT output pins can be used for this interrupt as determined by the MODE and Host Control (HCON) SFR setting. The Host responds to the Request for Service interrupt by reading the Host Status (HSTAT) SFR to determine the source of the interrupt. The Host then reads the Threshold number of bytes from the FIFO. The internal CPU may continue to write to the FIFO during the Host read of the FIFO Output channel.

Data Stream Commands may be written to the Output FIFO channel at any time during a write of data bytes. The write instruction need only specify the Command Out (COUT) SFR in the direct register instruction used. Immediate Commands may also be written at any time to the Immediate Command OUT (IMOUT) SFR. The Host reads Immediate Commands from the Immediate Command OUT (IMOUT).

The internal CPU can determine the number of bytes to write to the FIFO Output channel in one of three ways. The first, and most efficient, is by utilizing the internal DMA processor which will automatically manage the writing of data to avoid Underrun or Overrun Errors. The second is for the internal CPU to read the Output FIFO channels Read and Write Pointers and compare their values to determine the available space. The third method for determining the available FIFO space is to always write the programmed channel size number of bytes to the Output FIFO. This method would use the Overrun Error flag and interrupt to halt FIFO writing whenever the available space was less than the channel size. The interrupt service routine could read the channel pointers to determine or monitor the available channel space. The time required for the internal CPU to write data to the Output FIFO channel is a function of the individual instruction cycle time and the number of bytes to be written.

Host Read from the FIFO

The Host reads data or Data Stream Commands (DSC) from the FIFO in response to the Host Status (HSTAT) SFR flags and interrupts, if enabled. All Host read operations access the same UPI-452 internal I/O Buffer Latch. At the end of the previous Host FIFO read cycle a byte is loaded from the FIFO into the I/O Buffer Latch and Host Status (HSTAT) SFR bit 5 is set or cleared (1 = DSC, 0 = data) to reflect the state of the byte's FIFO ninth bit. If the FIFO ninth bit is set (= 1) indicating a DSC, an interrupt is generated to the external Host via INTRQ pin or INTRQIN/INTRQOUT pins as determined by Host Control (HCON) SFR bit 1. The Host then reads the Host Status (HSTAT) SFR to determine the source of the interrupt.

The most efficient Host read operation of the FIFO Output channel is through the use of Host DMA. The UPI-452 fully supports external DMA handshaking. The MODE and Host Control SFRs control the configuration of UPI-452 Host DMA handshake outputs. If Host DMA is used the Threshold Request for Service interrupt asserts the UPI-452 DMA Request (DRQOUT) output. The Host DMA processor acknowledges with $\overline{\text{DACK}}$ which acts as a chip select of the FIFO channels. The DMA transfer would stop when either the threshold byte count had been read, as programmed in the Host DMA processor, or when the DRQOUT output is brought inactive by the UPI-452.

INPUT CHANNEL

This section covers the data path from the HOST to the internal CPU or internal DMA processor. The details of Data Stream Command or Immediate Command processing during internal DMA operations are covered in the DMA section below.

Host Write to the FIFO

The Host writes data and Data Stream Commands into the FIFO through the FIFO IN (FIN) and Command IN (CIN) SFRs. When a Threshold number of bytes has been read out of the Input FIFO channel by the internal CPU, the Host Status SFR Input FIFO Request for Service bit is set and an interrupt, if enabled, is generated to the Host. The Input FIFO Threshold interrupt tells the Host that it may write the next block of data into the FIFO. Either the INTRQ or DRQIN/INTRQIN output pins can be used for this interrupt as determined by the MODE and Host Control (HCON) SFR settings. The Host may continue to write to the FIFO Input channel during the internal CPU read of the FIFO. Data Stream Commands may be written to the FIFO Input channel at any time during a write of data bytes. Immediate Commands may also be written at any time to the Immediate Command IN (IMIN) SFR.

The Host also has three methods for determining the available FIFO space. Two are essentially identical to that of the internal CPU. They involve reading the FIFO Input channel pointers and using the Host Status SFR Underrun and Overrun Error flags and Request for Service interrupts these would generate, if enabled in combination. The third involves using the UPI-452 Host DMA controller handshake signals and the programmed Input FIFO Threshold. The Host would receive a Request for Service interrupt when an Input FIFO channel has a Theshold number of bytes able to be written by the Host. The Host service routine would then write the Threshold number of bytes to the FIFO.

If a Host DMA is used to write to the FIFO Input channel, the Threshold Request for Service interrupt could assert the UPI-452 DRQIN output. The Host DMA processor would assert $\overline{\text{DACK}}$ and the FIFO write would be completed by Host the DMA processor. The DMA transfer would stop when either the Threshold byte count had been written or the DRQIN output was removed by the UPI-452. Additional details on Host and internal DMA operation is given below.

Internal Read of the FIFO

At the end of an internal CPU read cycle a byte is loaded from the FIFO buffer into the FIFO IN/Command IN SFR and Slave Status (SSTAT) SFR bit 1 is set or cleared (1 = data, 0 = DSC) to reflect the state of the FIFO ninth bit. If the byte is a DSC, the FIFO ninth bit is set (= 1) and an interrupt is generated, if enabled, to the Internal CPU. The internal CPU then reads the Slave Status (SSTAT) SFR to determine the source of the interrupt.

Immediate Commands are written by the Host and read by the internal CPU through the Immediate Command IN (IMIN) SFR. Once written, an Immediate Command sets the Slave Status (SSTAT) SFR flag bit and generates an interrupt, if enabled, to the internal CPU. In response to the interrupt the internal CPU reads the Slave Status (SSTAT) SFR to determine the source of the interrupt and service the Immediate Command.

FIFO INPUT/OUTPUT CHANNEL SIZE

Host

The Host does not have direct control of the FIFO Input or Output channel sizes or configuration. The Host can, however, issue Data Stream Commands or Immediate Commands to the UPI-452 instructing the UPI-452 to reconfigure the FIFO interface by invoking FIFO DMA Freeze Mode. The Data Stream Command or Immediate Command would be a vector to a service routine which performs the specific reconfiguration.

UPI-452 Internal

The default power-on reset FIFO channel sizes are listed in the "Initialization" section and can be set only by the internal CPU during FIFO DMA Freeze Mode. The FIFO channel size is selected to achieve the optimum application performance. The entire 128 byte FIFO can be allocated to either the Input or Output channel. In this case the other channel consists of a single SFR; FIFO IN/Command IN or FIFO OUT/Command OUT SFR. Figure 4 shows a FIFO division with a portion devoted to each channel. Figure 5 shows a FIFO configuration with all 128 bytes assigned to the Output channel.

The FIFO channel Threshold feature allows the user to match the FIFO channel size and the performance of the internal and Host data transfer rates. The programmed Threshold provides an elasticity to the data transfer operation. An example is if the Host FIFO

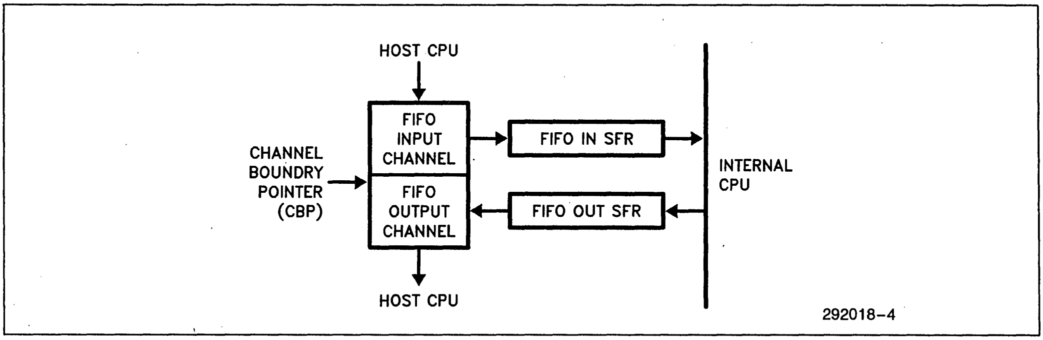


Figure 4. Full Duplex FIFO Operation

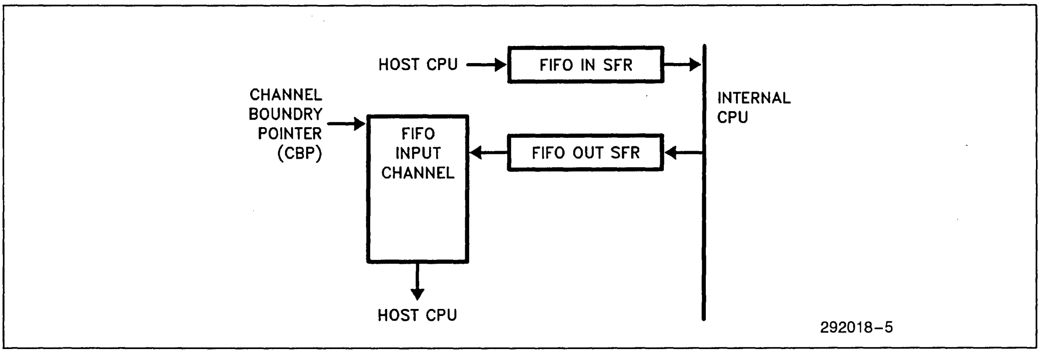


Figure 5. Entire FIFO Assigned to Output Channel

data transfer rate is twice as fast as the internal FIFO DMA data transfer rate. In this example the FIFO Input channel size is programmed to be 64 bytes and the Input channel Threshold is programmed to be 20 bytes. The Host writes the first 64 bytes to the Input FIFO. When the internal DMA processor has read 20 bytes the Threshold interrupt, or DMA request (DRQIN), is generated to signal the Host to begin writing more data to the Input FIFO channel. The internal DMA processor continues to read data from the Input channel as the Host, or Host DMA processor, writes to the FIFO. The Host can write 40 bytes to the FIFO Input channels in the time it takes for the internal DMA processor to read 20 more bytes from it. This will keep both the Host and internal DMA operating at their maximum rates without forcing one to wait for the other.

Two methods of managing the FIFO size are possible; fixed and variable channel size. A fixed channel size is one where the channel is configured at initialization and remains unchanged throughout program execution. In a variable FIFO channel size, the configuration is changed dynamically to meet the data transmission requirements as needed. An example of a variable channel size application is the mass storage subsystem described earlier. To meet the demands of a large data block transfer the FIFO size could be fully allocated to the Input or Output channel as needed. The Thresholds are also reprogrammed to match the respective data transfer rates.

An example of a fixed channel size application might be one which uses the UPI-452 to directly control a series of stepper motors. The UPI-452 manages the motor operation and status as required. This would include pulse train, acceleration, deceleration and feedback. The Host transmits motor commands to the UPI-452 in blocks of 6–10 bytes. Each block of motor command data is preceded by a command to the UPI-452 which selects a specific motor. The UPI-452 transmits blocks of data to the Host which provides motor and overall system status. The data and embedded commands structure, indicating the specific motor, is the same. In this example the default 64 bytes per channel might be adequate for both channels.

INTERRUPT RESPONSE TIMING

Interrupts enable the Host UPI-452 FIFO buffer interface and the internal CPU FIFO buffer interface to operate with a minimum of overhead on the respective CPU. Each CPU is "interrupted" to service the FIFO on an as needed basis only. In configuring the FIFO buffer Thresholds and choosing the appropriate internal DMA Mode the user must take into account the interrupt response time for both CPUs. These response times will affect the DMA transfer rates for each channel. By choosing FIFO channel Thresholds which reflect both the respective DMA transfer rate and the interrupt response time the user will achieve the maximum data throughput and system bus decoupling. This in turn will mean the overall available system bus bandwidth will increase.

The following section describes the FIFO interrupt interface to the Host and internal CPU. It also describes an analysis of sample interrupt response times for the Host and UPI-452 internal CPU. These equations and the example times shown are then used in the DMA section to further analyze an optimum Host UPI-452 interface.

HOST

Interrupts to the Host processor are supported by the three UPI-452 output pins; INTRQ, DRQIN/INTRQIN and DRQOUT/INTRQOUT. INTRQ is a general purpose Request For Service interrupt output. DRQIN/INTRQIN and DRQOUT/INRQOUT pins are multiplexed to provide two special "Request for Service" FIFO interrupt request lines when DMA is disabled. A FIFO Input or Output channel Request for Service interrupt is generated based upon the value programmed in the respective channel's Threshold SFRs; Input Threshold (ITHR), and Output Threshold

(OTHR) SFRs. Additional interrupts are provided for FIFO Underrun and Overrun Errors, Data Stream Commands, and Immediate Commands. Table 4 lists the eight UPI-452 interrupt sources as they appear in the HSTAT SFR to the Host processor.

Table 4. UPI-452 to Host Interrupt Sources

HSTAT SFR Bit	Interrupt Source
HST7	Output FIFO Underrun Error
HST6	Immediate Command Out SFR Status
HST5	Data Stream Command/Data at Output FIFO Status
HST4	Output FIFO Request for Service Status
HST3	Input FIFO Overrun Error Condition
HST2	Immediate Comamnd In SFR Status
HST1	FIFO DMA Freeze/Normal Mode Status
HST0	Input FIFO Request for Service

The interrupt response time required by the Host processor is application and system specific. In general, a typical sequence of Host interrupt response events and the approximate times associated with each are listed in Equation 1.

The example assumes the hardware configuration shown in Figure 1, iAPX 286/UPI-452 Block Diagram, with an 8259A Programmable Interrupt Controller. The timing analysis in Equation 1 also assumes the following; no other interrupt is either in process or pending, nor is the 286 in a LOCK condition. The current instruction completion time is 8 clock cycles (800 ns @ 10 MHz), or 4 bus cycles. The interrupt service routine first executes a PUSHA instruction, PUSH All General Registers, to save all iAPX 286 internal registers. This requires 19 clocks (or 2.0 μs @ 10 MHz), or 10 bus cycles (rounded to complete bus cycle). The next service routine instruction reads the UPI-452 Host Status SFR to determine the interrupt source.

It is important to note that any UPI-452 INTRQ interrupt service routine should ALWAYS mask for the Freeze Mode bit first. This will insure that Freeze Mode always has the highest priority. This will also save the time required to mask for bits which are forced inactive during Freeze Mode, before checking the Freeze Mode bit. Access to the FIFO channels by the Host is inhibited during Freeze Mode. Freeze Mode is covered in more detail below.

To initiate the interrupt the UPI-452 activates the INTRQ output. The interrupt acknowledge sequence requires two bus cycles, 400 ns (10 MHz iAPX 286), for the two INTA pulse sequence.

Equation 1. Host Interrupt Response Time

Action	Time	Bus Cycles*
Current instruction execution completion	800 ns	4
INTA sequence	400 ns	2
Interrupt service routine (time to host first $\overline{\text{READ}}$ of UPI-452)	2000 ns	10
Total Interrupt Response Time	2.3 μs	16

NOTE:
10 MHz iAPX 286 bus cycle, 200 ns each

UPI-452 Internal

The internal CPU FIFO interrupt interface is essentially identical to that of the Host to the FIFO. Three internal interrupt sources support the FIFO operation; FIFO-Slave bus Interface Buffer, DMA Channel 0 and DMA Channel 1 Requests. These interrupts provide a maximum decoupling of the FIFO buffer and the internal CPU. The four different internal DMA Modes available add flexibility to the interface.

The FIFO-Slave Bus Interface interrupt response is also similar to the Host response to an INTRQ Request for Service interrupt. The internal CPU responds to the interrupt by reading the Slave Status (SSTAT) SFR to determine the source of the interrupt. This allows the user to prioritize the Slave Status flag response to meet the users application needs.

The internal interrupt response time is dependent on the current instruction execution, whether the interrupt is enabled, and the interrupt priority. In general, to finish execution of the current instruction, respond to the interrupt request, push the Program Counter (PC) and vector to the first instruction of the interrupt service routine requires from 38 to 86 oscillator periods (2.38 to 5.38 μs @ 16 MHz). If the interrupt is due to an Immediate Command or DSC, additional time is required to read the Immediate Command or DSC SFR and vector to the appropriate service routine. This means two service routines back to back. One service routine to read the Slave Status (SSTAT) SFR to determine the source of the Request for Service interrupt, and second the service routine pointed to by the Immediate Command or DSC byte read from the respective SFR.

DMA

DMA is the fastest and most efficient way for the Host or internal CPU to communicate.with the FIFO buffer. The UPI-452 provides support for both of these DMA paths. The two DMA paths and operations are fully independent of each other and can function simultaneously. While the Host DMA processor is performing a DMA transfer to or from the FIFO, the UPI-452 internal DMA processor can be doing the same.

Below are descriptions of both the Host and internal DMA operations. Both DMA paths can operate asynchronously and at different transfer rates. In order to make the most of this simultaneous asynchronous operation it is necessary to calculate the two transfer rates and accurately match their operations. Matching the different transfer rates is done by a combination of accurately programmed FIFO channel size and channel Thresholds. This provides the maximum Host and internal CPU to FIFO buffer interface decoupling. Below is a description of each of the two DMA operations and sample calculations for determining transfer rates. The next section of this application note, "Interface Latency", details the considerations involved in analyzing effective transfer rates when the overhead associated with each transfer is considered.

HOST FIFO DMA

DMA transfers between the Host and UPI-452 FIFO buffer are controlled by the Host CPU's DMA controller, and is independent of the UPI-452's internal two channel DMA processor. The UPI-452's internal DMA processor supports data transfers between the UPI-452 internal RAM, external RAM (via the Local Expansion Bus) and the various Special Function Registers including the FIFO Input and Output channel SFRs.

The maximum DMA transfer rate is achieved by the minimum DMA transfer cycle time to accomplish a source to destination move. The minimum Host UPI-452 FIFO DMA cycle time possible is determined by the $\overline{\text{READ}}$ and $\overline{\text{WRITE}}$ pulse widths, UPI-452 command recovery times in relation to the DMA transfer timing and DMA controller transfer mode used. Table 5 shows the relationship between the iAPX-286, iAPX-186 and UPI-452 for various DMA as well as non-DMA byte by byte transfer modes versus processor frequencies.

Host processor speed vs wait states required with UPI-452 running at 16 MHz:

Table 5. Host UPI-452 Data Transfer Performance

Processor & Speed	Wait States: Back to Back $\overline{\text{READ}}$/ $\overline{\text{WRITE}}$'s	DMA: Single Cycle	Two Cycle
iAPX-186* 8 MHz	0	N/A*	0
10 MHz	0	N/A*	0
12.5 MHz	1	N/A*	0
iAPX-286** 6 MHz	0	0	0
8 MHz	1	1	0
10 MHz	2	2	0

NOTES:
* iAPX 186 On-chip DMA processor is two cycle operation only.
** iAPX 286 assumes 82258 ADMA (or other DMA) running 286 bus cycles at 286 clock rate.

In this application note system example, shown in Figure 1, DMA operation is assumed to be two bus cycle I/O to memory or memory to I/O. Two cycle DMA consists of a fetch bus cycle from the source and a store bus cycle to the destination. The data is stored in the DMA controller's registers before being sent to the destination. Single cycle DMA transfers involve a simultaneous fetch from the source and store to the destination. As the most common method of I/O-memory DMA operation, two cycle DMA transfers are the focus of this application note analysis. Equation 2 illustrates a calculation of the overall transfer rate between the FIFO buffer and external Host for a maximum FIFO size transfer. The equation does not account for the latency of initiating the DMA transfer.

Equation 2. Host FIFO DMA Transfer Rate—Input or Output Channel

2 Cycle DMA Transfer-I/O (UPI-452) to System Memory
= FIFO channel size* (DMA $\overline{\text{READ}}/\overline{\text{WRITE}}$ FIFO time + DMA $\overline{\text{WRITE}}/\overline{\text{READ}}$ Memory Time)
= 128 bytes* (200 ns + 200 ns)
= 51.2 μs
= 256 bus cycles*

NOTES:
*10 MHz iAPX 286, 200 ns bus cycles.

The UPI-452 design is optimized for high speed DMA transfers between the Host and the FIFO buffer. The

UPI-452 internal FIFO buffer control logic provides the necessary synchronization of the external Host event and the internal CPU machine cycle during UPI-452 RD/WR accesses. This internal synchronization is addressed by the TCC AC specification of the UPI-452 shown in Figure 6. TCC is the time from the leading or trailing edge of a UPI-452 RD/WR to the same edge of the next UPI-452 RD/WR. The TCC time is effectively another way of measuring the system bus cycle time with reference to UPI-452 accesses.

In the iAPX-286 10 MHz system depicted in this application note the bus cycle time is 200 ns. Alternate cycle accesses of the UPI-452 during two cycle DMA operation yields a TCC time of 400 ns which is more than the TCC minimum time of 375 ns. Back to back Host UPI-452 READ/WRITE accesses may require wait states as shown in Table 5. The difference between 10 MHz iAPX-186 and 10 MHz iAPX 286 required wait states is due to the number of clock cycles in the respective bus cycle timings. The four clocks in a 10 MHz iAPX 186 bus cycle means a minimum TCC time of 400 ns versus 200 ns for a 10 MHz iAPX 286 with two clock cycle zero wait state bus cycle.

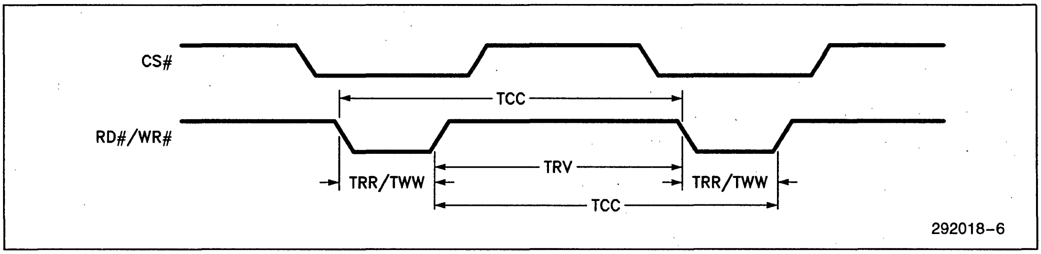
DMA handshaking between the Host DMA controller and the UPI-452 is supported by three pins on the UPI-452; DRQIN/INTRQIN, DRQOUT/INTRQOUT and DACK. The DRQIN/INTRQIN and DRQOUT/INTRQOUT outputs are two multiplexed DMA or interrupt request pins. The function of these pins is controlled by MODE SFR bit 6 (MD6). DRQIN and DRQOUT provide a direct interface to the Host system DMA controller (see Figure 1). In response to a DRQIN or DRQOUT request, the Host DMA controller initiates control of the system bus using HLD/HLDA. The FIFO Input or Output channel transfer is accomplished with a minimum of Host overhead and system bus bandwidth.

The third handshake signal pin is DACK which is used as a chip select during DMA data transfers. The UPI-452 Host READ and WRITE input signals select the respective Input and Output FIFO channel during DMA transfers. The CS and address lines provide DMA acknowledge for processors with onboard DMA controllers which do not generate a DACK signal.

The iAPX 286 Block I/O Instructions provide an alternative to two cycle DMA data transfers with approximately the same data rate. The String Input and Output instructions (INS & OUTS) when combined with the Repeat (REP) prefix, modifies INS and OUTS to provide a means of transferring blocks of data between I/O and Memory. The data transfer rate using REP INS/OUTS instructions is calculated in the same way as two cycle DMA transfer times. Each READ or WRITE would be 200 ns in a 10 MHz iAPX 286 system. The maximum transfer rate possible is 2.5 MBytes/second. The Block I/O FIFO data transfer calculation is the same as that shown in Equation 2 for two cycle DMA data transfers including TCC timing effects.

FIFO Data Structure and Host DMA

During a Host DMA write to the FIFO, if a DSC is to be written, the DMA transfer is stopped, the DSC is written and the DMA restarted. During a Host DMA read from the FIFO, if a DSC is loaded into the I/O Buffer Latch the DMA request, DRQOUT, will be deactivated (see Figure 2 above). The Host Status (HSTAT) SFR Data Stream Command bit is set and the INTRQ interrupt output goes active, if enabled. The Host responds to the interrupt as described above.



Symbol	Description	Var. Osc.	@16 MHz
TCC	Command Cycle Time	6 * Tclcl	375 ns min
TRV	Command Recovery Time	75	75 ns min

Figure 6. UPI-452 Command Cycle Timing

Once INTRQ is deactivated and the DSC has been read by the Host, the DMA request, $\overline{\text{DRQOUT}}$, is reasserted by the UPI-452. The DMA request then remains active until the transfer is complete or another DSC is loaded into the I/O Buffer Latch.

An Immediate Command written by the internal CPU during a Host DMA FIFO transfer also causes the Host Status flag and INTRQ to go active if enabled. In this case the Immediate Command would not terminate the DMA transfer unless terminated by the Host. The INTRQ line remains active until the Host reads the Host Status (HSTAT) SFR to determine the source of the interrupt.

The net effect of a Data Stream Command (DSC) on DMA data transfer rates is to add an additional factor to the data transfer rate equation. This added factor is shown in Equation 3. An Immediate Command has the same effect on the data transfer rate if the Immediate Command interrupt is recognized by the Host during a DMA transfer. If the DMA transfer is completed before the Immediate Command interrupt is recognized, the effect on the DMA transfer rate depends on whether the block being transmitted is larger than the FIFO channel size. If the block is larger than the programmed FIFO channel size the transfer rate depends on whether the Immediate Command flag or interrupt is recognized between partial block transfers.

The FIFO configuration shown in Equation 3 is arbitrary since there is no way of predicting the size relative to when a DSC would be loaded into the I/O Buffer Latch. The Host DMA rate shown is for a UPI-452 (Memory Mapped or I/O) to 286 System Memory transfer as described earlier. The equations do not account for the latency of intiating the DMA transfer.

Equation 3. Minimum host FIFO DMA Transfer Rate Including Data Stream Command(s)

Minimum Host/FIFO DMA Transfer Rate w/ DSC
= FIFO size* Host DMA 2 cycle time transfer rate
  + iAPX 286 interrupt response time (Eq. #1)
= (32 bytes* (200 ns + 200 ns)) + 2.3 μs
= 15.1 μs
= 75.5 bus cycles (@10 MHz iAPX286, 200 ns bus cycle)

UPI-452 INTERNAL DMA PROCESSOR

The two identical internal DMA channels allow high speed data transfers from one UPI-452 writable memory space to another. The following UPI-452 memory spaces can be used with internal DMA channels:

Internal Data Memory (RAM)
External Data Memory (RAM)
Special Function Registers (SFR)

The FIFO can be accessed during internal DMA operations by specifying the FIFO IN (FIN) SFR as the DMA Source Address (SAR) or the FIFO OUT (FOUT) SFR as the Destination Address (DAR). Table 6 lists the four types of internal DMA transfers and their respective timings.

Table 6. UPI-452 Internal DMA Controller Cycle Timings

Source	Destination	Machine Cycles**	@12 MHz	@16 MHz
Internal Data Mem. or SFR	Internal Data Mem. or SFR	1	1 μs	750 ns
Internal Data Mem. or SFR	External Data Mem.	1	1 μs	750 ns
External Data Mem.	Internal Data Mem. or SFR	1	1 μs	750 ns
*External Data Memory	External Data Memory	2	2 μs	1.5 μs

NOTES:
*External Data Memory DMA transfer applies to UPI-452 Local Bus only.
**MSC-51 Machine cycle = 12 clock cycles (TCLCL).

FIFO Data Structure and Internal DMA

The effect of Data Stream Commands and Immediate Commands on the internal DMA transfers is essentially the same as the effect on Host FIFO DMA transfers. Recognition also depends upon the programmed DMA Mode, the interrupts enabled, and their priorities. The net internal effect is the same for each possible internal case. The time required to respond to the Immediate or Data Stream Command is a function of the instruction time required. This must be calculated by the user based on the instruction cycle time given in the MSC-51 Instruction Set description in the Intel Microcontroller Handbook.

It is important to note that the internal DMA processor modes and the internal FIFO logic work together to automatically manage internal DMA transfers as data moves into and out of the FIFO. The two most appropriate internal DMA processor modes for the FIFO are FIFO Demand Mode and FIFO Alternate Cycle Mode. In FIFO Demand Mode, once the correct Slave Control and DMA Mode bits are set, the internal Input FIFO channel DMA transfer occurs whenever the Slave Control Input FIFO Request for Service flag is set. The DMA transfer continues until the flag is cleared or when the Input FIFO Read Pointer SFR (IRPR) equals zero. If data continues to be entered by the Host, the internal DMA continues until an internal interrupt of higher priority, if enabled, interrupts the DMA transfer, the internal DMA byte count reaches zero or until the Input FIFO Read Pointer equals zero. A complete description of interrupts and DMA Modes can be found in the UPI-452 Data Sheet.

DMA Modes

The internal DMA processor has four modes of operation. Each DMA channel is software programmable to operate in either Block Mode or Demand Mode. Demand Mode may be further programmed to operate in Burst or Alternate Cycle Mode. Burst Mode causes the internal processor to halt its execution and dedicate its resources exclusively to the DMA transfer. Alternate Cycle Mode causes DMA cycles and instruction cycles to occur alternately. A detailed description of each DMA Mode can be found in the UPI-452 Data Sheet.

INTERFACE LATENCY

The interface latency is the time required to accommodate all of the overhead associated with an individual data transfer. Data transfer rates between the Host system and UPI-452 FIFO, with a block size less than or equal to the programmed FIFO channel size, are calculated using the Host system DMA rate. (see Host DMA description above). In this case, the entire block could be transferred in one operation. The total latency is the time required to accomplish the block DMA transfer, the interrupt response or poll of the Host Status SFR response time, and the time required to initate the Host DMA processor.

A DMA transfer between the Host and UPI-452 FIFO with a block size greater than the programmed FIFO channel size introduces additional overhead. This additional overhead is from three sources; first, is the time to actually perform the DMA transfer. Second, the overhead of initializing the DMA processor, third, the handshaking between each FIFO block required to transfer the entire data block. This could be time to wait for the FIFO to be emptied and/or the interrupt response time to restart the DMA transfer of the next portion of the block. A fourth component may also be the time required to respond to Underrun and Overrun FIFO Errors.

Table 7 shows six typical FIFO Input/Output channel sizes and the Host DMA transfers times for each. The timings shown reflect a 10 MHz system bus two cycle I/O to Memory DMA transfer rate of 2.5 MBytes/second as shown in Equation 1. The times given would be the same for iAPX 286 I/O block move instructions REP INS and REP OUTS as described earlier.

Table 7. Host DMA FIFO Data Transfer Times

FIFO Size:	32	43	64	85	96	128	bytes
Full or Empty	1/4	1/3	1/2	2/3	3/4	Full or Empty	
Time	12.8	17.2	25.6	34.0	38.4	51.2	μs

Table 8 shows six typical FIFO Input/Output channel sizes and the internal DMA processor data transfers times for each. The timings shown are for a UPI-452 single cycle Burst Mode transfer at 16 MHz or 750 ns per machine cycle in or out of the FIFO channels. The machine cycle time is that of the MSC-51 CPU; 6 states, 2 XTAL2 clock cycles each or 12 clock cycles per machine cycle. Details on the MSC-51 machine cycle timings and operation may be found in the Intel Microcontroller Handbook.

Table 8. UPI-452 Internal DMA FIFO Data Transfer Times

FIFO Size:	32	43	64	85	96	128	bytes
Full or Empty	¼	⅓	½	⅔	¾	Full or Empty	
Time	24.0	32.3	48.0	64.6	72.0	96.0	μs

A larger than programmed FIFO channel size data block internal DMA transfer requires internal arbitration. The UPI-452 provides a variety of features which support arbitration between the two internal DMA channels and the FIFO. An example is the internal DMA processor FIFO Demand Mode described above. FIFO Demand Mode DMA transfers occur continuously until the Slave Status Request for Service Flag is deactivated. Demand Mode is especially useful for continuous data transfers requiring immediate attention. FIFO Alternate Cycle Mode provides for interleaving DMA transfers and instruction cycles to achieve a maximum of software flexibility. Both internal DMA channels can be used simultaneously to provide continuous transfer for both Input and Output FIFO channels.

Byte by byte transfers between the FIFO and internal CPU timing is a function of the specific instruction cycle time. Of the 111 MCS-51 instructions, 64 require 12 clock cycles, 45 require 24 clock cycles and 2 require 48 clock cycles. Most instructions involving SFRs are 24 clock cycles except accumulator (for example, MOV direct, A) or logical operations (ANL direct, A). Typical instruction and their timings are shown in Table 9.

Oscillator Period: @ 12 MHz = 83.3 ns
@ 16 MHz = 62.5 ns

Table 9. Typical Instruction Cycle Timings

Instruction	Oscillator Periods	@12 MHz	@16 MHz
MOV direct†, A	12	1 μs	750 ns
MOV direct, direct	24	2 μs	1.5 μs

NOTE:
† Direct = 8-bit internal data locations address. This could be an Internal Data RAM location (0–255) or a SFR [i.e., I/O port, control register, etc.]

Byte by byte FIFO data transfers introduce an additional overhead factor not found in internal DMA operations. This factor is the FIFO block size to be transferred; the number of empty locations in the Output channel, or the number of bytes in the Input FIFO channel. As described above in the FIFO Data Structure section, the block size would have to be determined by reading the channel read and write pointer and calculating the space available. Another alternative uses the FIFO Overrun and Underrun Error flags to manage the transfers by accepting error flags. In either case the instructions needed have a significant impact on the internal FIFO data transfer rate latency equation.

A typical effective internal FIFO channel transfer rate using internal DMA is shown in Equation 4. Equation 5 shows the latency using byte by byte transfers with an arbitrary factor added for internal CPU block size calculation. These two equations contrast the effective transfer rates when using internal DMA versus individual instructions to transfer 128 bytes. The effective transfer rate is approximately four times as fast using DMA versus using individual instructions (96 μs with DMA versus 492 μs non-DMA).

Equation 4. Effective Internal FIFO Transfer Time Using Internal DMA

Effective Internal FIFO Transfer Rate with DMA
= FIFO channel size * Internal DMA Burst Mode Single Cycle DMA Time
= 128 Bytes * 750 ns
= 96 μs

Equation 5. Effective FIFO Transfer Time Using Individual Instructions

Effective Internal FIFO Transfer Rate without DMA
= FIFO channel size * Instruction Cycle Time + Block size calculation Time
= 128 Bytes * (24 oscillator periods @ 16 MHz) + 20 instructions (24 oscillator period each @ 16 MHz)
= 128 * 1.5 μs + 300 μs
= 492 μs

FIFO DMA FREEZE MODE INTERFACE

FIFO DMA Freeze Mode provides a means of locking the Host out of the FIFO Input and Output channels. FIFO DMA Freeze Mode can be invoked for a variety of reasons, for example, to reconfigure the UPI-452 Local Expansion Bus, or change the baud rate on the serial channel. The primary reason the FIFO DMA Freeze Mode is provided is to ensure that the Host does not read from or write to the FIFO while the FIFO interface is being altered. ONLY the internal CPU has the capability of altering the FIFO Special Function Registers, and these SFRs can ONLY be altered during FIFO DMA Freeze Mode. FIFO DMA Freeze Mode inhibits Host access of the FIFO while the internal CPU reconfigures the FIFO.

FIFO DMA Freeze Mode should not be arbitrarily invoked while the UPI-452 is in normal operation. Because the external CPU runs asynchronously to the internal CPU, invoking freeze mode without first properly resolving the FIFO Host interface may have serious consequences. Freeze Mode may be invoked only by the internal CPU.

The internal CPU invokes Freeze Mode by setting bit 3 of the Slave Control SFR (SC3). This automatically forces the Slave and Host Status SFR FIFO DMA Freeze Mode to In Progress (SSTAT SST5 = 0, HSTAT SFR HST1 = 1). INTRQ goes active, if enabled by MODE SFR bit 4, whenever FIFO DMA Freeze Mode is invoked to notify the Host. The Host reads the Host Status SFR to determine the source of the interrupt. INTRQ and the Slave and Host Status FIFO DMA Freeze Mode bits are reset by the Host READ of the Host Status SFR.

During FIFO DMA Freeze Mode the Host has access to the Host Status and Control SFRs. All other Host FIFO interface access is inhibited. Table 10 lists the FIFO DMA Freeze Mode status of all slave bus interface Special Function Registers. The internal DMA processor is disabled during FIFO DMA Freeze Mode and the internal CPU has write access to all of the FIFO control SFRs (Table 11).

If FIFO DMA Freeze Mode is invoked without stopping the host, only the last two bytes of data written into or read from the FIFO will be valid. The timing diagram for disabling the FIFO module to the external Host interface is illustrated in Figure 7. Due to this synchronization sequence, the UPI-452 might not go into FIFO DMA Freeze Mode immediately after the Slave Control SFR FIFO 7 DMA Freeze Mode bit (SC3) is set = 0. A special bit in the Slave Status SFR (SST5) is provided to indicate the status of the FIFO DMA Freeze Mode. The FIFO DMA Freeze Mode operations described in this section are only valid after SST5 is cleared.

Either the Host or internal CPU can request FIFO DMA Freeze Mode. The first step is to issue an Immediate Command indicating that FIFO DMA Freeze Mode will be invoked. Upon receiving the Immediate Command, the external CPU should complete servicing all pending interrupts and DMA requests, then send an Immediate Command back to the internal CPU acknowledging the FIFO DMA Freeze Mode request. After issuing the first Immediate Command, the internal CPU should not perform any action on the FIFO until FIFO DMA Freeze Mode is invoked. The handshaking is the same in reverse if the HOST CPU initiates FIFO DMA Freeze Mode.

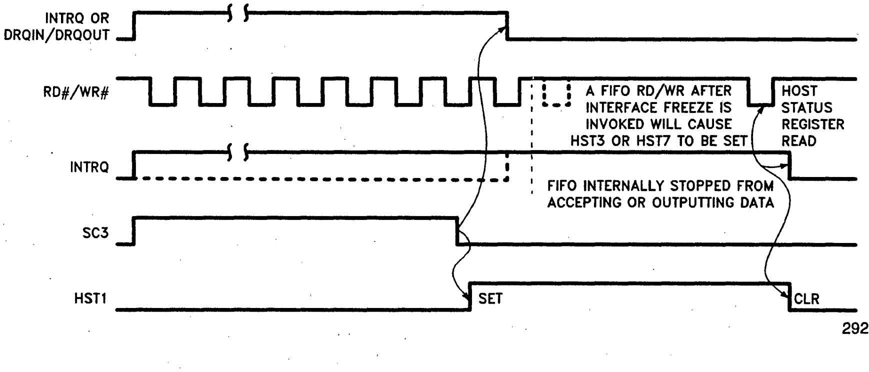
After the slave bus interface is frozen, the internal CPU can perform the operations listed below on the FIFO Special Function Registers. These operations are allowed only during FIFO DMA Freeze Mode. Table 11 summarizes the characteristics of all the FIFO Special Function Registers during Normal and FIFO DMA Freeze Modes.

For FIFO Reconfiguration

1. Changing the Channel Boundary Pointer SFR.
2. Changing the Input and Output Threshold SFR.

To Enhance the testability

3. Writing to the read and write pointers of the Input and Output FIFO's.
4. Writing to and reading the Host Control SFRs.
5. Controlling some bits of Host and Slave Status SFRs.
6. Reading the Immediate Command Out SFR and Writing to the Immediate Command in SFR.



NOTE:
Timing Diagram of disabling of FIFO Module—External Host Interface.

Figure 7. Disabling FIFO to Host Slave Interface Timing Diagram

The sequence of events for invoking FIFO DMA Freeze Mode are listed in Figure 8.

1. Immediate Command to request FIFO DMA Freeze Mode (interrupt)
2. Host/internal CPU interrupt response/service
3. Host/internal CPU clear/service all pending interrupts and FIFO data
4. Internal CPU sets Slave Control (SLCON) FIFO DMA

   Freeze Mode bit = 0, FIFO DMA Freeze Mode, Host Status SFR FIFO DMA Freeze Mode Status bit = 1, INTRQ active (high)
5. Host $\overline{\text{READ}}$ Host Status SFR
6. Internal CPU reconfigures FIFO SFRs
7. Internal CPU resets Slave Control (SLCON) FIFO DMA

   Freeze Mode bit = 1, Normal Mode, Host Status FIFO DMA Freeze Mode Status bit = 0.
8. Internal CPU issues Immediate Command to Host indicating that FIFO DMA Freeze Mode is complete

   or

   Host polls Host Status SFR FIFO DMA Freeze Mode bit to determine end of reconfiguration

Figure 8. Sequence of Events to Invoke FIFO DMA Freeze Mode

EXAMPLE CONFIGURATION

An example of the time required to reconfigure the FIFO 180 degrees, for example from 128 bytes Input to 128 bytes Output, is shown in Figure 9. The example approximates the time based on several assumptions;

1. The FIFO Input channel is full-128 bytes of data
2. Output FIFO channel is empty-1 byte
3. No Data Stream Commands in the FIFO.
4. The Immediate Command interrupt is responded to immediately—highest priority—by Host and internal CPU.
5. Respective interrupt response times
   a. Host (Equation 3 above) = approximately 1.6 μs
   b. Internal CPU is 86 oscillator periods or approximately 5.38 μs worst case.

Event	Time
Immediate Command from Host to UPI-452 to request FIFO DMA Freeze Mode (iAPX286 $\overline{\text{WRITE}}$)	0.30 μs
Internal CPU interrupt response/ service	5.38 μs
Internal CPU clears FIFO-128 bytes DMA	96.00 μs
Internal CPU sets Slave Control Freeze Mode bit	0.75 μs
Immediate Command to Host-Freeze Mode in progress Host Immediate Command interrupt response	2.3 μs
Internal CPU reconfigures FIFO SFRs	
Channel Boundary Pointer SFR	0.75 μs
Input Threshold SFR	0.75 μs
Output Threshold SFR	0.75 μs
Internal CPU resets Slave Control (SLCON) Freeze Mode bit = 1, Normal Mode, and automatically resets Host Status FIFO DMA Freeze Mode bit	2.3 μs
Internal CPU writes Immediate Command Out	0.75 μs
Host Immediate Command interrupt service	2.3 μs
Total Minimum Time to Reconfigure FIFO	112.33 μs

Figure 9. Sequence of Events to Invoke FIFO DMA Freeze Mode and Timings

Table 10. Slave Bus Interface Status During FIFO DMA Freezer Mode

Interface Pins;							Operation In	Status In
$\overline{\text{DACK}}$	$\overline{\text{CS}}$	A2	A1	A0	$\overline{\text{READ}}$	$\overline{\text{WRITE}}$	Normal Mode	Freeze Mode
1	0	0	1	0	0	1	Read Host Status SFR	Operational
1	0	0	1	1	0	1	Read Host Control SFR	Operational
1	0	0	1	1	1	0	Write Host Control SFR	Disabled
1	0	0	0	0	0	1	Data or DMA data from Output Channel	Disabled
1	0	0	0	0	1	0	Data or DMA data to Input Channel	Disabled
1	0	0	0	1	0	1	Data Stream Command from Output Channel	Disabled
1	0	0	0	1	1	0	Data Stream Command to Input Channel	Disabled
1	0	1	0	0	0	1	Read Immediate Command Out from Output Channel	Disabled
1	0	1	0	0	1	0	Write Immediate Command In to Input Channel	Disabled
0	X	X	X	X	0	1	DMA Data from Output Channel	Disabled
0	X	X	X	X	1	0	DMA Data to Input Channel	Disabled

NOTE:
X = don't care

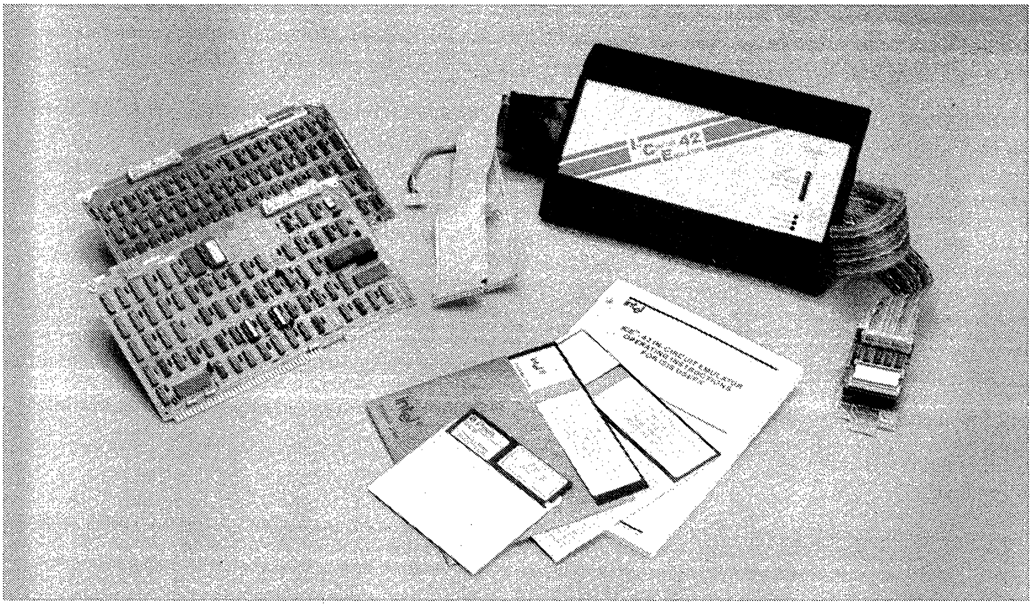
Table 11. FIFO SFR's Characteristics During FIFO DMA Freeze Mode

Label	Name	Normal Operation (SST5 = 1)	Freeze Mode Operation (SST5 = 0)
HCON	Host Control	Not Accessible	Read & Write
HSTAT	Host Status	Read Only	Read & Write
SLCON	Slave Control	Read & Write	Read & Write
SSTAT	Slave Status	Read Only	Read & Write
IEP	Interrupt Enable & Priority	Read & Write	Read & Write
MODE	Mode Register	Read & Write	Read & Write
IWPR	Input FIFO Write Pointer	Read Only	Read & Write
IRPR	Input FIFO Read Pointer	Read Only	Read & Write
OWPR	Output FIFO Write Pointer	Read Only	Read & Write
ORPR	Output FIFO Read Pointer	Read Only	Read & Write
CBP	Channel Boundary Pointer	Read Only	Read & Write
IMIN	Immediate Command In	Read Only	Read & Write
IMONT	Immediate Command Out	Read & Write	Read & Write
FIN	FIFO IN	Read Only	Read Only
CIN	COMMAND IN	Read Only	Read Only
FOUT	FIFO OUT	Read & Write	Read & Write
COUT	COMMAND OUT	Read & Write	Read & Write
ITHR	Input FIFO Threshold	Read Only	Read & Write
OTHR	Other FIFO Threshold	Read Only	Read & Write

ICE™-42
8042 IN-CIRCUIT EMULATOR

- Precise, Full-Speed, Real-Time Emulation
  - Load, Drive, Timing Characteristics
  - Full-Speed Program RAM
  - Parallel Ports
  - Data Bus
- User-Specified Breakpoints
- Execution Trace
  - User-Specified Qualifier Registers
  - Conditional Trigger
  - Symbolic Groupings and Display
  - Instruction and Frame Modes
- Emulation Timer
- Full Symbolic Debugging
- Single-Line Assembly and Disassembly for Program Instruction Changes
- Macro Commands and Conditional Block Constructs for Automated Debugging Sessions
- HELP Facility: ICE™-42 Command Syntax Reference at the Console
- User Confidence Test of ICE-42 Hardware

The ICE-42 module resides in the Intellec Microcomputer Development System and interfaces to any user-designed 8042 or 8041A system through a cable terminating in an 8042 emulator microprocessor and a pin-compatible plug. The emulator processor, together with 2K bytes of user program RAM located in the ICE-42 buffer box, replaces the 8042 device in the user system while maintaining the 8042 electrical and timing characteristics. Powerful Intellec debugging functions are thus extended into the user system. Using the ICE-42 module, the designer can emulate the system's 8042 chip in real-time or single-step mode. Breakpoints allow the user to stop emulation on user-specified conditions, and a trace qualifier feature allows the conditional collection of 1000 frames of trace data. Using the single-line 8042 assembler the user may alter program memory using the 8042 assembler mnemonics and symbolic references, without leaving the emulator environment. Frequently used command sequences can be combined into compound commands and identified as macros with user-defined names.



210818-1

FUNCTIONAL DESCRIPTION

Integrated Hardware and Software Development

The ICE-42 emulator allows hardware and software development to proceed interactively. This approach is more effective than the traditional method of independent hardware and software development followed by system integration. With the ICE-42 module, prototype hardware can be added to the system as it is designed. Software and hardware integration occurs while the product is being developed. Figure 1 shows the ICE-42 emulator connected to a user prototype.

The ICE-42 emulator assists four stages of development.

SOFTWARE DEBUGGING

This emulator operates without being connected to the user's system before any of the user's hardware is available. In this stage ICE-42 debugging capabilities can be used in conjunction with the Intellec text editor and 8042 macro-assembler to facilitate program development.

HARDWARE DEVELOPMENT

The ICE-42 module's precise emulation characteristics and full-speed program RAM make it a valuable tool for debugging hardware.

SYSTEM INTEGRATION

Integration of software and hardware begins when any functional element of the user system hardware is connected to the 8042 socket. As each section of the user's hardware is completed, it is added to the prototype. Thus, each section of the hardware and software is "system" tested in real-time operation as it becomes available.

SYSTEM TEST

When the user's prototype is complete, it is tested with the final version of the user system software. The ICE-42 module is then used for real-time emulation of the 8042 chip to debug the system as a completed unit.

The final product verification test may be performed using the 8742 EPROM version of the 8042 microcomputer. Thus, the ICE-42 module provides the ability to debug a prototype or production system at any stage in its development without introducing extraneous hardware or software test tools.

Symbolic Debugging

The ICE-42 emulator permits the user to define and to use symbolic, rather than absolute, references to program and data memory addresses. Thus, there is no need to recall or look up the addresses of key locations in the program, or to become involved with machine code.

When a symbol is used for memory reference in an ICE-42 emulator command, the emulator supplies the corresponding location as stored in the ICE-42 emulator symbol table. This table can be loaded with the symbol table produced by the assembler during application program assembly. The user obtains the symbol table during software preparation simply by using the "DEBUG" switch in the 8042 macroassembler. Furthermore, the user interactively modifies the emulator symbol table by adding new symbols or changing or deleting old ones. This feature provides great flexibility in debugging and minimizes the need to work with hexadecimal values.

Through symbolic references in combination with other features of the emulator, the user can easily:

- Interpret the results of emulation activity collected during trace.
- Disassemble program memory to mnemonics, or assemble mnemonic instructions to executable code.
- Reference labels or addresses defined in a user program.

Automated Debugging and Testing

MACRO COMMAND

A macro is a set of commands given a name. A group of commands executed frequently can be defined as a macro. The user executes the group of commands by typing a colon followed by the macro name. Up to ten parameters may be passed to the macro.

Macro commands can be defined at the beginning of a debug session and then used throughout the whole session. One or more macro definitions can be saved on diskette for later use. The Intellec text editor may be used to edit the macro file. The macro definitions are easy to include in any later emulation session.

The power of the development system can be applied to manufacturing testing as well as development by writing test sequences as macros. The macros are stored on diskettes for use during system test.

COMPOUND COMMAND

Compound commands provide conditional execution of commands (IF command) and execution of commands repeatedly until certain conditions are met (COUNT, REPEAT commands).

Compound commands may be nested any number of times, and may be used in macro commands.

Example:

```
*DEFINE .I=0        ;Define symbol .I to 0
*COUNT 100H         ;Repeat the following
                     commands 100H times.
.*IF .I AND 1 THEN  ;Check if .I is odd
..*CBYTE.I=.I       ;Fill the memory at
                     location .I to value .I
..*END
.*.I-.I+1           ;Increment .I by 1.
.*END               ;Command executes
                     upon carriage-return
                     after END
```

(The asterisks are system prompts; the dots indicate the nesting level of compound commands.)

Operating Modes

The ICE-42 software is an Intellec RAM-based program that provides easy-to-use commands for initiating emulation, defining breakpoints, controlling trace data collection, and displaying and controlling system parameters. ICE-42 commands are configured with a broad range of modifiers that provide maximum flexibility in describing the operation to be performed.

EMULATION

The ICE-42 module can emulate the operation of prototype 8042 system, at real-time speed (up to 12 MHz) or in single steps. Emulation commands to the ICE-42 module control the process of setting up, running, and halting an emulation of the user's 8042-based system. Breakpoints and tracepoints enable the ICE-42 emulator to halt emulation and provide a detailed trace of execution in any part of the user's program. A summary of the emulation commands is shown in Table 1.

Table 1. Major Emulation Commands

Command	Description
GO	Begins real-time emulation and optionally specifies break conditions.
BR0, BR1, BR	Sets or displays either or both Breakpoint Registers used for stopping real-time emulation.
STEP	Performs single-step emulation.
QR0, QR1	Specifies match conditions for qualified trace.
TR	Specifies or displays trace-data collection conditions and optionally sets Qualifier Register (QR0, QR1).
Synchronization Line Commands	Sets and displays status of synchronization line outputs or latched inputs. Used to allow real-time emulation or trace to start and stop synchronously with external events.

Breakpoints

The ICE-42 hardware includes two breakpoint registers that allow halting of emulation when specified conditions are met. The emulator continuously compares the values stored in the breakpoint registers with the status of specified address, opcode, operand, or port values, and halts emulation when this comparison is satisfied. When an instruction initiates a break, that instruction is executed completely before the break takes place. The ICE-42 emulator then regains control of the console and enters the interrogation mode. With the breakpoint feature, the user can request an emulation break when the program:

- Executes an instruction at a specific address or within a range of addresses.
- Executes a particular opcode.
- Receives a specific signal on a port pin.
- Fetches a particular operand from the user program memory
- Fetches an operand from a specific address in program memory.

Trace and Tracepoints

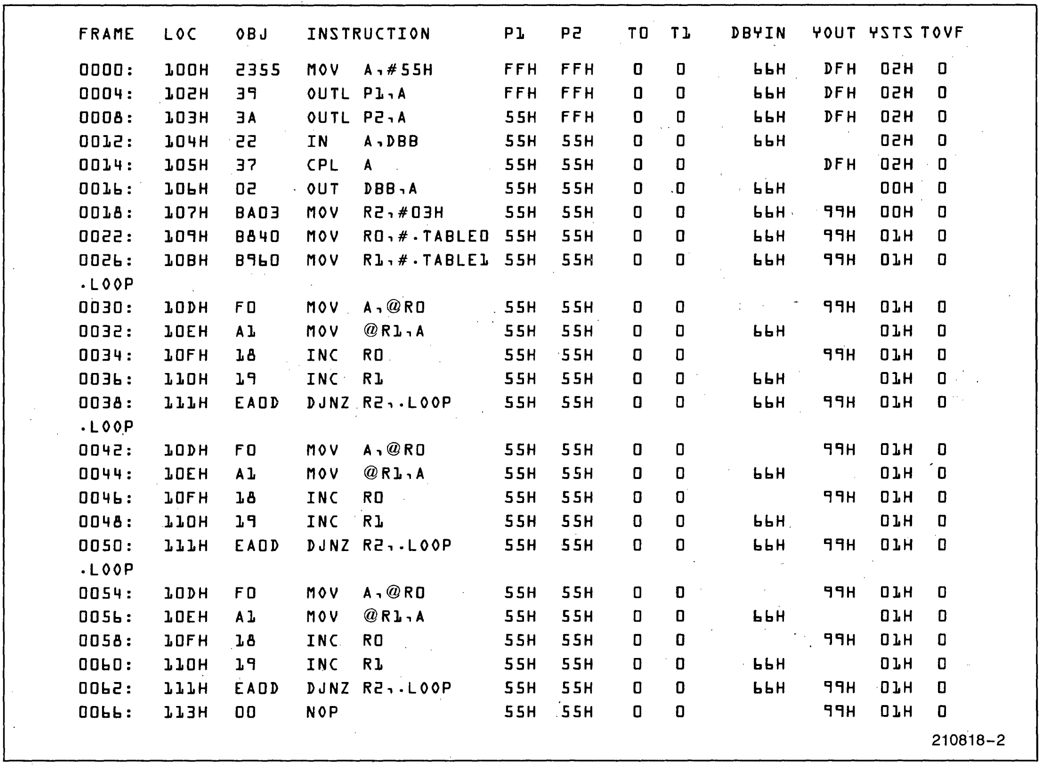
Tracing is used with real-time and single-step emulation to record diagnostic information in the trace buffer as a program is executed. The information collected includes opcodes executed, port values, and memory addresses. The ICE-42 emulator collects 1000 frames of trace data.

If desired this information can be displayed as assembler instruction mnemonics for analysis during interrogation or single-step mode. The trace-collection facility may be set to run conditionally or unconditionally. Two unique trace qualifier registers, specified in the same way as breakpoint registers, govern conditional trace activity. The qualifiers can be used to condition trace data collection to take place as follows:

- Under all conditions (forever).
- Only while the trace qualifier is satisfied.
- For the frames or instructions preceding the time when a trace qualifier is first satisfied (pre-trigger trace).
- For the frames or instructions after a trace qualifier is first satisfied (post-triggered trace).

Table 2 shows an example of trace display.

Table 2. Trace Display (Instruction Mode)



```
FRAME     LOC     OBJ     INSTRUCTION      P1    P2    T0  T1   DBYIN   YOUT  YSTS TOVF

0000:     100H    2355    MOV  A,#55H      FFH   FFH   0   0    66H     DFH   02H   0
0004:     102H    39      OUTL P1,A        FFH   FFH   0   0    66H     DFH   02H   0
0008:     103H    3A      OUTL P2,A        55H   FFH   0   0    66H     DFH   02H   0
0012:     104H    22      IN   A,DBB       55H   55H   0   0    66H           02H   0
0014:     105H    37      CPL  A           55H   55H   0   0            DFH   02H   0
0016:     106H    02      OUT  DBB,A       55H   55H   0   0    66H           00H   0
0018:     107H    BA03    MOV  R2,#03H     55H   55H   0   0    66H     99H   00H   0
0022:     109H    B840    MOV  R0,#.TABLE0 55H   55H   0   0    66H     99H   01H   0
0026:     10BH    B960    MOV  R1,#.TABLE1 55H   55H   0   0    66H     99H   01H   0
.LOOP
0030:     10DH    F0      MOV  A,@R0       55H   55H   0   0            99H   01H   0
0032:     10EH    A1      MOV  @R1,A       55H   55H   0   0    66H           01H   0
0034:     10FH    18      INC  R0          55H   55H   0   0            99H   01H   0
0036:     110H    19      INC  R1          55H   55H   0   0    66H           01H   0
0038:     111H    EA0D    DJNZ R2,.LOOP    55H   55H   0   0    66H     99H   01H   0
.LOOP
0042:     10DH    F0      MOV  A,@R0       55H   55H   0   0            99H   01H   0
0044:     10EH    A1      MOV  @R1,A       55H   55H   0   0    66H           01H   0
0046:     10FH    18      INC  R0          55H   55H   0   0            99H   01H   0
0048:     110H    19      INC  R1          55H   55H   0   0    66H           01H   0
0050:     111H    EA0D    DJNZ R2,.LOOP    55H   55H   0   0    66H     99H   01H   0
.LOOP
0054:     10DH    F0      MOV  A,@R0       55H   55H   0   0            99H   01H   0
0056:     10EH    A1      MOV  @R1,A       55H   55H   0   0    66H           01H   0
0058:     10FH    18      INC  R0          55H   55H   0   0            99H   01H   0
0060:     110H    19      INC  R1          55H   55H   0   0    66H           01H   0
0062:     111H    EA0D    DJNZ R2,.LOOP    55H   55H   0   0    66H     99H   01H   0
0066:     113H    00      NOP              55H   55H   0   0            99H   01H   0
```

210818-2

INTERROGATION AND UTILITY

Interrogation and utility commands give convenient access to detailed information about the user program and the state of the 8042 that is useful in debugging hardware and software. Changes can be made in memory and in the 8042 registers, flags, and port values. Commands are also provided for various utility operations such as loading and saving program files, defining symbols, displaying trace data, controlling system synchronization and returning control to ISIS-II. A summary of the basic interrogation and utility commands is shown in Table 3. Two additional time-saving emulator features are discussed below.

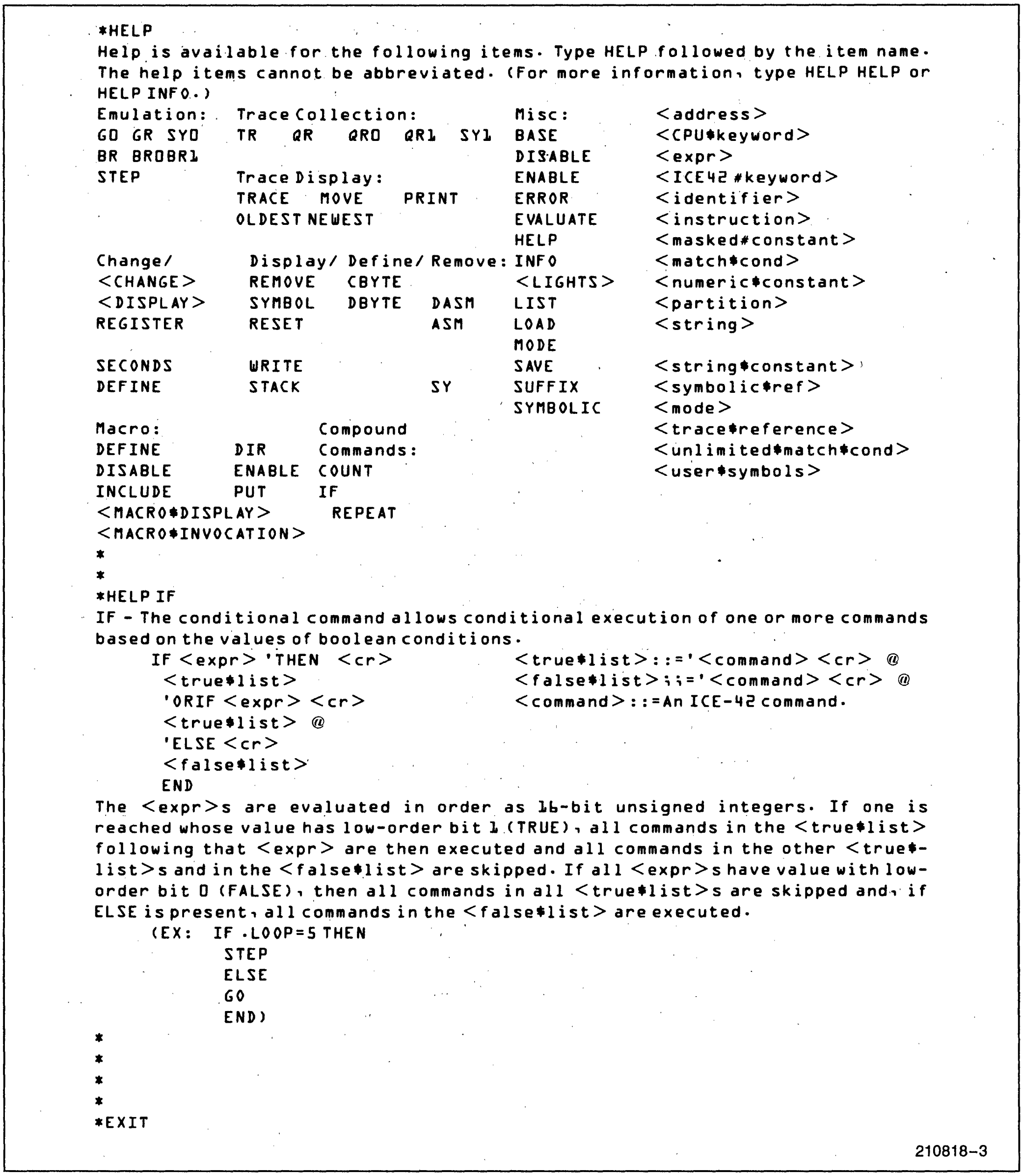
Single-Line Assembler/Disassembler

The single-line assembler/disassembler (ASM and DASM commands) permits the designer to examine and alter program memory using assembly language mnemonics, without leaving the emulator environment or requiring time-consuming program reassembly. When assembling new mnemonic instructions into program memory, previously defined symbolic references (from the original program assembly, or subsequently defined during the emulation session)

Table 3. Major Interrogation and Utility Commands

Command	Description
HELP	Displays help messages for ICE-42 emulator command-entry assistance.
LOAD	Loads user object program (8042 code) into user-program memory, and user symbols into ICE-42 emulator symbol table.
SAVE	Saves ICE-42 emulator symbol table and/or user object program in ISIS-II hexadecimal file.
LIST	Copies all emulator console input and output to ISIS-II file.
EXIT	Terminates ICE-42 emulator operation.
DEFINE	Defines ICE-42 emulator symbol or macro.
REMOVE	Removes ICE-42 emulator symbol or macro.
ASM	Assembles mnemonic instructions into user-program memory.
DASM	Disassembles and displays user-program memory contents.
Change/Display Commands	Change or display value of symbolic reference in ICE-42 emulator symbol table, contents of key-word references (including registers, I/O ports, and status flags), or memory references.
EVALUATE	Evaluates expression and displays resulting value.
MACRO	Displays ICE-42 macro or macros.
INTERRUPT	Displays contents for the Data Bus and timer interrupt registers.
SECONDS	Displays contents of emulation timer, in microseconds.
Trace Commands	Position trace buffer pointer and select format for trace display.
PRINT	Displays trace data pointed to by trace buffer pointer.
MODE	Sets or displays the emulation mode, 8041A or 8042.

Table 4. HELP Command



```
*HELP
Help is available for the following items. Type HELP followed by the item name.
The help items cannot be abbreviated. (For more information, type HELP HELP or
HELP INFO.)
Emulation:   Trace Collection:          Misc:         <address>
GO GR SY0    TR    QR   QR0  QR1   SY1  BASE          <CPU*keyword>
BR BR0BR1                               DISABLE       <expr>
STEP         Trace Display:             ENABLE        <ICE42 #keyword>
             TRACE   MOVE    PRINT      ERROR         <identifier>
             OLDEST NEWEST              EVALUATE      <instruction>
                                        HELP          <masked#constant>
Change/        Display/ Define/ Remove: INFO          <match*cond>
<CHANGE>       REMOVE   CBYTE           <LIGHTS>      <numeric*constant>
<DISPLAY>      SYMBOL   DBYTE   DASM    LIST          <partition>
REGISTER       RESET            ASM     LOAD          <string>
                                        MODE
SECONDS        WRITE                    SAVE          <string*constant>
DEFINE         STACK            SY      SUFFIX        <symbolic*ref>
                                        SYMBOLIC      <mode>
Macro:               Compound                         <trace*reference>
DEFINE       DIR     Commands:                        <unlimited*match*cond>
DISABLE      ENABLE  COUNT                            <user*symbols>
INCLUDE      PUT     IF
<MACRO*DISPLAY>       REPEAT
<MACRO*INVOCATION>
*
*
*HELP IF
IF - The conditional command allows conditional execution of one or more commands
based on the values of boolean conditions.
     IF <expr> 'THEN <cr>               <true*list>::='<command> <cr> @
      <true*list>                       <false*list>::='<command> <cr> @
      'ORIF <expr> <cr>                 <command>::=An ICE-42 command.
      <true*list> @
      'ELSE <cr>
      <false*list>
      END
The <expr>s are evaluated in order as 16-bit unsigned integers. If one is
reached whose value has low-order bit 1 (TRUE), all commands in the <true*list>
following that <expr> are then executed and all commands in the other <true*-
list>s and in the <false*list> are skipped. If all <expr>s have value with low-
order bit 0 (FALSE), then all commands in all <true*list>s are skipped and, if
ELSE is present, all commands in the <false*list> are executed.
     (EX:  IF .LOOP=5 THEN
            STEP
            ELSE
            GO
            END)
*
*
*
*
*EXIT
```

210818-3

may be used in the instruction operand field. The emulator supplies the absolute address or data values as stored in the emulator symbol table. These features eliminate user time spent translating to and from machine code and searching for absolute addresses, with a corresponding reduction in transcription errors.

HELP

The HELP file allows display of ICE-42 command syntax information at the Intellec console. By typing "HELP", a listing of all items for which help messages are available is displayed. Typing "HELP <Item>" then displays relevant information about the item requested, including typical usage examples. Table 4 shows some sample HELP messages.

EMULATION ACCURACY

The speed and interface demands of a high-performance single-chip microcomputer require extremely accurate emulation, including full-speed, real-time operation with the full function of the microcomputer. The ICE-42 module achieves accurate emulation with an 8042 emulator chip, a special configuration of the 8042 microcomputer family, as its emulation processor.

Each of the 40 pins on the user plug is connected directly to the corresponding 8042 pin on the emulator chip. Thus the user system sees the emulator as an 8042 microcomputer at the 8042 socket. The resulting characteristics provide extremely accurate emulation of the 8042 including speed, timing characteristics, load and drive values, and crystal operation. However, the emulator may draw more power from the user system than a standard 8042 family device.

Additional emulator processor pins provide signals such as internal address, data, clock, and control lines to the emulator buffer box. These signals let static RAM in the buffer box substitute for on-chip program ROM or EPROM. The emulator chip also gives the ICE module "back-door" access to internal chip operation, allowing the emulator to break and trace execution without interfering with the values on the user-system pins.

210818-4

Figure 1. A typical 8042 Development Configuration. The host system is an Intellec Series IV. The ICE-42 module is connected to a user prototype system.

SPECIFICATIONS

ICE™-42 Operating Requirements

Intellec Model 800, Series II, Series III, or Series IV Microcomputer Development System (64K RAM required)

System console (Model 800 only)

Intellec Diskette Operating System: ISIS (Version 3.4 or later).

Equipment Supplied
- Printed circuit boards (2)
- Emulation buffer box, Intellec interface cables, and user-interface cable with 8042 emulation processor
- Crystal power accessory
- Operating instructions manuals
- Diskette-based ICE-42 software (single and double density)

Emulation Clock

User's system clock (up to 12 MHz) or ICE-42 crystal power accessory (12 MHz)

Environmental Characteristics

Operating Temperature: 0°C to 40°C
Operating Humidity: Up to 95% relative humidity without condensation.

Physical Characteristics

Printed Circuit Boards

Width: 12.00 in. (30.48 cm)
Height: 6.75 in. (17.15 cm)
Depth: 0.50 in. (1.27 cm)

Buffer Box

Width: 8.00 in. (20.32 cm)
Length: 12.00 in. (30.48 cm)
Depth: 1.75 in. (4.44 cm)
Weight: 4.0 lb (1.81 kg)

Electrical Characteristics

DC Power Requirements (from Intellec® system)

V_{CC} = +5V, ±5%
I_{CC} = 13.2A max; 11.0A typical
V_{DD} = +12V, ±5%
I_{DD} = 0.1A max; 0.05A typical
V_{BB} = −10V, ±5%
I_{BB} = 0.05A max; 0.01A typical

User plug characteristics at 8042 socket—Same as 8042 or 8742 except that the user system sees an added load of 25 pF capacitance and 50 μA leakage from the ICE-42 emulator user plug at ports 1, 2, T0, and T1.

ORDERING INFORMATION

Part Number	Description
ICE-42	8042 Microcontroller In-Circuit Emulator, cable assembly and interactive diskette software

iUP-200A/iUP-201A UNIVERSAL PROM PROGRAMMERS

MAJOR iUP-200A/iUP-201A FEATURES:

- **Personality Module Plug-Ins Provide Programming Support for Intel and Intel-Compatible EPROMs, EPLDs, Microcontrollers, Flash Memories, and other Programmable Devices**
- **PROM Programming Software (iPPS) Makes Programming Easy with IBM PC, XT, AT, and PC Compatibles**
- **Supports Personality Modules and GUPI Base W/Adaptors**
- **iUP-200A Provides On-Line Operation with a Built-In Serial RS232 Interface and Software for a PC Environment**
- **iUP-201A Provides Same On-Line Performance and Adds Keyboard and Display for Stand-Alone Use**
- **iUP-201A Stand-Alone Capability Includes Device Previewing, Editing, Duplication, and Download from any Source Over RS232C Port**
- **Updates and Add-Ons Have Maintained Even the Earliest iUP-200 and iUP-201 Users at the State-of-Art**

The iUP-200A and iUP-201A universal programmers program and verify data in Intel and Intel compatible, programmable devices. The iUP-200A and iUP-201A universal programmers provide on-line programming and verification in a growing variety of development environments using the Intel PROM programming software (iPPS). In addition, the iUP-201A universal programmer supports off-line, stand-alone program editing, duplication, and memory locking. The iUP-200A universal programmer is expandable to an iUP-201A model.

210319–1

FUNCTIONAL DESCRIPTION

The iUP-200A universal programmer operates in on-line mode. The iUP-201A universal programmer operates in both on-line and off-line mode.

On-Line System Hardware

The iUP-200A and iUP-201A universal programmers are free-standing units that, when connected to a host computer with at least 64K bytes of memory, provide on-line programming and verification of Intel programmable devices. In addition, the universal programmer can read the contents of the ROM versions of supported devices.

The universal programmer communicates with the host through a standard RS232C serial data link. Different versions of the iUP-200A and iUP-201A are equipped with different cables, including the cable most commonly used for interfacing to that host. Care should be taken that the version with the correct cable for your particular system is selected, as cable requirements can vary with your host configuration. A serial converter is needed when using the MDS 800 as a host system. (Serial converters are available from other manufacturers.)

Each universal programmer contains the CPU, selectable power supply, static RAM, programmable timer, interface for personality modules, RS232C interface for the host system, and control firmware in EPROM. The iUP-201A also has a keyboard and display.

A personality module or GUPI Adaptor adapts the universal programmer to a family of devices; it contains all the hardware and software necessary to program either a family of Intel devices or a single Intel device. The user inserts the personality module into the universal programmer front panel.

On-Line System Software

The iUP-200A and iUP201A includes your choice of one copy of Intel's PROM Programming software iPPS, selected from a list of versions for different operating systems and hosts. Each version includes the software implementation designed for that host and O.S. and the RS232C cable most commonly used. Additional versions may be purchased separately if you decide to change hosts at a later date. The iPPS software provides user control through an easy-to-use interactive interface. The iPPS software performs the following functions to make EPROM programming quick and easy:

- Reads devices
- Programs devices directly or from a file
- Verifies device data with buffer data
- Locks device memory from unauthorized access (on devices which support this feature)
- Prints device contents on the network or development system printer
- Performs interactive formatting operations such as interleaving, nibble swapping, bit reversal, and block moves
- Programs multiple devices from the source file, prompting the user to insert new devices
- Uses a buffer to change device contents

All iPPS commands, as well as program address and data information, are entered through the host system ASCII keyboard and displayed on the system CRT.

The iPPS software supports data manipulation in the following Intel formats: 8080 hexadecimal ASCII, 8080 absolute object, 8086 hexadecimal ASCII, 8086 absolute object, and 80286 absolute object. Addresses and data can be displayed in binary, octal, decimal, or hexadecimal. The user can easily change default data formats as well as number bases. iPPS can also access disk files.

For programming Intel EPLDs, the iUP-200A/201A can be controlled by Intel's Logic Programming Software (LPS). LPS programs EPLDs from JEDEC files produced by Intel's logic compiler. (iPPS can also program EPLDs, but only from pre-programmed device masters.)

System Expansion

The iUP-200A universal programmer can be easily upgraded (by the user) to an iUP-201A universal programmer for off-line operation. The upgrade kit (iUP-PAK-A) is available from Intel or your local Intel distributor.

Off-Line System

The iUP-201A universal programmer has all the on-line features of the iUP-200A universal programmer plus off-line editing, device duplication, program verification, and locking of device memory independent of the host system. The iUP-201A universal programmer also accepts Intel hexadecimal programs developed on non-Intel development systems. Just a few keystrokes download the program into the iUP RAM for editing and loading into a device.

Off-line commands are entered via a 16-character keypad. A 24-character display shows programmer status.

SYSTEM DIAGNOSTICS

Both the iUP-200A and iUP-201A universal programmers include self-contained system diagnostics that verify system operation and aid the user in fault isolation.

PERSONALITY MODULES

For some devices, a personality module is the interface between the iUP-200A/iUP-201A universal programmer (or an iPDS system) and a selected device. Personality modules contain all the hardware and firmware for reading and programming a family of Intel devices. Table 1 lists the devices supported by the different modules.

For most devices, the GUPI module and interchangeable GUPI Adaptors provide the interface between the programmer and the device being programmed (see Figure 1). the GUPI (Generic Universal Programmer Interface) module is a base module that intefaces to the iUP-200A/201A and GUPI Adaptors. GUPI Adaptors tailor the GUPI module base signals to a family of devices or an individual device. The GUPI module and GUPI Adaptors provide a lower-cost method of device support than if unique Personality Modules were offered for each device/family. Tables 2 and 3 show which Adaptors support which devices.

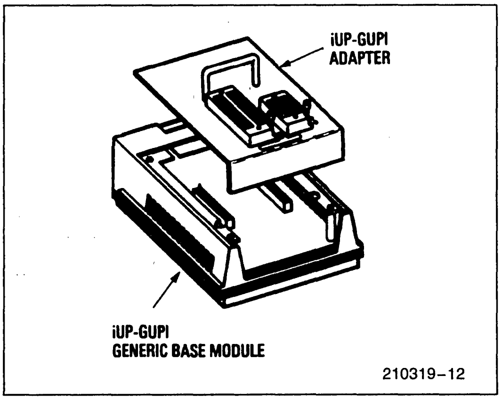


Figure 1. GUPI Adaptor

Table 1. iUP Personality Programming Modules

Device Type	Fast 27/K Module	Fast 27/K U2 Kit	Fast 27/K-CON* Kit	F27/128 Module	F87/44A Module	F87/51A Module
EPROM				2716		
				2732		
				2732A		
	2764	2764	2764	2764		
	2764A	2764A	2764A			
		27C64	27C64			
		87C64	87C64			
	27128	27128	27128	27128		
		27128A	27128A			
	27256	27256	27256			
		27C256	27C256			
		27512	27512			
		27513	27513			
E^{2}PROM				2815		
				2816		
		2817A	2817A			
Microcontroller					8041A	8748
					8042	8748H
					8044AH	
					8741H	8749H
					8742	8751
					8744H	8751H
						8048
						8048H
						8049
						8049H
						8050H
						8051
					8755A	

*Quick-Pulse Programming™ algorithm

Table 2. iUP-GUPI Adaptors for Programming Memories

Device Type	GUPI 27010	GUPI 27011	GUPI 27210	GUPI Flash	GUPI 8742	GUPI MCS-51	GUPI 8796	GUPI 8796LCC	GUPI 87C51GB	GUPI MCS-96LCC
EPROM	27010	27011	27210							
Flash				27F64 27F256 28F256						
Peripheral Microcontroller					8741AH 8742AH	8751H 87C51 8752BH 87C51FA 87C51FB	8794BH 8795BH 8796BH 8797BH	8796BH 8797BH	87C51GB	8797BH 87C196KB
Package Types	DIP	DIP	DIP	DIP	DIP	PLCC DIP	PGA DIP	LCC	PLCC	PLCC

Table 3. Programming Adaptors for EPLDs

Device Type	GUPI Logic-IID	GUPI Logic-12	GUPI 40D44J	GUPI Logic-18	GUPI Logic-18PGA	GUPI 85EPLD28	GUPI Logic-BIC
EPLD	5C060 5C090 5AC312	5C031 5C032 5C121	5AC324	5C180	5C180G	85C508	5CBIC
Package Types	DIP*	DIP	DIP PLCC	PLCC CJ	PGA	DIP PLCC	PLCC

*ADAPT units available to adapt DIP socket for PLCC package.

iUP-200A/iUP201A SPECIFICATIONS

Control Processor

Intel 8085A microprocessor
6.144 MHz clock rate

Memory

RAM—4.3 bytes static
ROM—12K bytes EPROM

Interfaces

Keyboard: 16-character hexadecimal and 12-function keypad (iUP-201A model only)

Display: 24-character alphanumeric (iUP-201A model only)

Software

Monitor— system controller in pre-programmed EPROM

iPPS — Intel PROM programming software on supplied diskette

Physical Characteristics

Depth: 15 inches (38.1 cm)
Width: 15 inches (38.1 cm)
Height: 6 inches (15.2 cm)
Weight: 15 pounds (6.9 kg)

Electrical Characteristics

Selectable 100, 120, 200, or 240 Vac ± 10%; 50-60 Hz

Maximum power consumption—80 watts

Environmental Characteristics

Reading Temperature:	10°C to 40°C
Programming Temperature:	25°C ±5°
Operating Humidity:	10% to 85% relative humidity

Reference Material

166041-001— *iUP-200A/201A Universal Programmer User's Guide.*

166042-001— *Getting Started with the iUP-200A/201A (For ISIS/iNDX Users).*

166043-001— *Getting Started with the iUP-200A/201A (For DOS Users).*

164853 — *iUP-200A/201A Universal Programmer Pocket Reference.*

ORDERING INFORMATION

Product Order Code	Description
iUP-200A 211A	On-Line PROM programmer with iPPS rel 1.4 on Single density ISIS II floppy
iUP-200A 212B	On-Line PROM programmer with iPPS rel 1.4 on Double density ISIS II floppy
iUP-200A 213C	On-Line PROM programmer with iPPS rel 2.0 for Series IV, on mini-floppy
iUP-200A 216D	On-Line PROM programmer with iPPS rel 2.0 for PC/DOS, and cable for PC or XT
iUP-200A 217D	On-Line PROM programmer with iPPS rel 2.0 for PC/DOS, and cable for AT
iUP-201A 211A	Off-Line and on-line PROM programmer with iPPS rel 1.4 on Single density ISIS II floppy
iUP-201A 212B	Off-Line and on-line PROM programmer with iPPS rel 1.4 on Double density ISIS II floppy
iUP-201A 213C	Off-Line and on-line PROM programmer with iPPS rel 2.0 for Series IV on mini-floppy
iUP-201A 216D	Off-Line and on-line PROM programmer with iPPS rel 2.0 for PC/DOS, and cable for PC or XT
iUP-201A 217D	Off-Line and on-line PROM programmer with iPPS rel 2.0 for PC/DOS, and cable for AT
iUP-200/201 U1* Upgrade Kit	Upgrades an iUP-200/201 universal programmer to an iUP-200A/201A universal programmer
iUP-DL	Download Support Kit for iUP-200A/201A upgrades programmer to support adaptors that use software programming (.DSS) files.
iUP-PAK-A Upgrade Kit	Upgrades an iUP-200/A universal programmer to an iUP-201A universal programmer

*Most personality modules can be used only with an iUP-200A/201A universal programmer or an iUP-200/iUP201 universal programmer upgraded to an A with the iUP-200/iUP-201 U1 upgrade kit.

Product Order Code	Description
piUP-GUPI	Generic Universal Programmer Interface (Base)

Software Sold Separately

Product Order Code	Description
211A	PROM programming software rel 1.4 on Single density ISIS II floppy
212B	PROM programming software rel 1.4 on Double density ISIS II floppy
213C	PROM programming software rel 2.0 for Series IV on mini-floppy
216D	PROM programming software rel 2.0 for PC/DOS with cable for PC or PC XT
217D	PROM programming software rel 2.0 for PC/DOS with cable for PC AT

Graphics Coprocessor Family

10

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.

For complete data sheet, see Volume I, Chapter 4

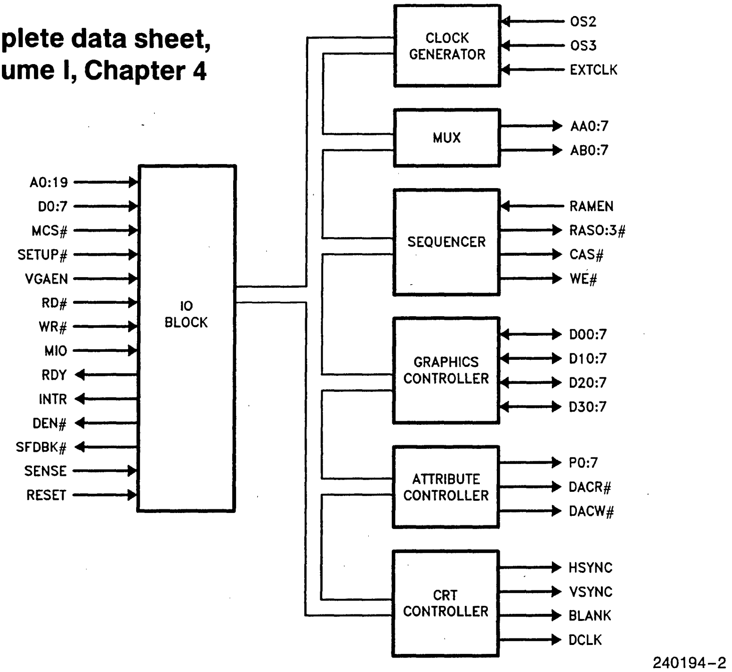


Figure 1. Block Diagram

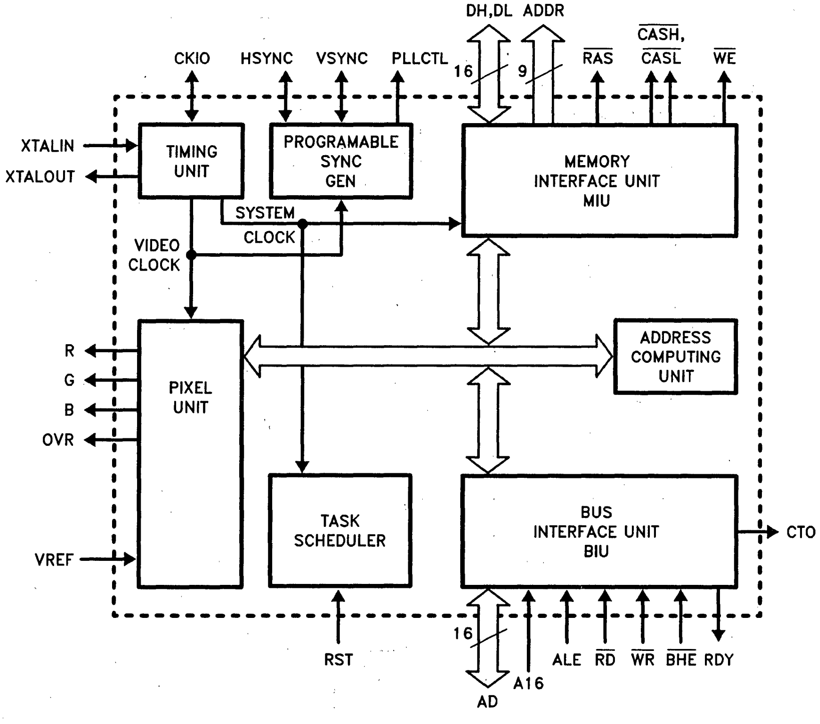
82716/VSDD
VIDEO STORAGE AND DISPLAY DEVICE

- Low Cost Graphics and Text Capability
- Minimum Chip Count Display Controller
- Displays Up to 16 Bit Map and Character Objects of Any Size
- On-Chip 16/4096 Color Palette
- On-Chip DRAM Controller
- On-Chip D/A Converters
- Arbitration of Processor RAM Requests
- NAPLPS and CEPT Compatible
- Objects Allow Windowing or Animation
- Horizontal Resolution up to 640 4-Bit Pixels
- Up to 512K Bytes of Display Memory
- Compatible with 8 and 16 Bit Processors/Micro Controllers
- Twin Mode Operation for Higher Throughput
- Powerful External Sync and Overlay Capabilities
- Video Clock Rate up to 20 MHz
- System Clock Rate up to 14.3 MHz

(Contact Factory for up-to-date specifications)

82716/VSDD is a low cost, highly integrated video controller. It displays graphics and textual information using a minimum of chips. It allows the management of up to 16 display objects on the screen at any one time. These objects may be formatted as bit map or character arrays and can be used for windowing or animation.

An on-chip color palette allows the selection of up to 16 colors, from a range of 4096. The palette can be programmed to drive a set of on-chip D/A converters. The VSDD also provides DRAM controller functions.



231680-1

Figure 1. VSDD Block Diagram

82716 VIDEO STORAGE AND DISPLAY DEVICE ADDITIONAL INFORMATION

You can obtain a free copy of the most recent 82716 data sheet specs by:

- **Calling your local Intel sales office and ask for Order No. 231680**
- **Calling toll-free (800) 548-4725 and ask for JB12**

OFFER EXPIRES 12/31/89

PRELIMINARY

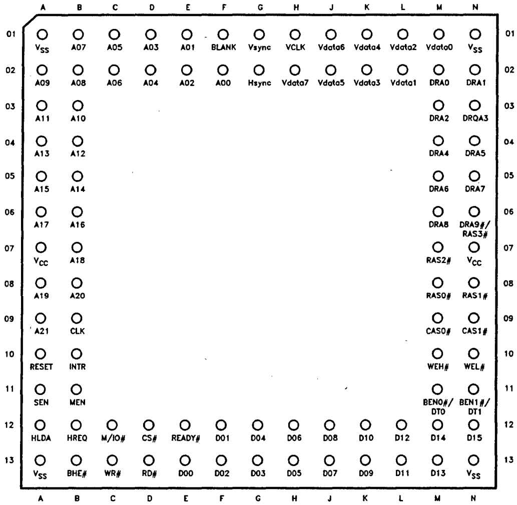
82786
CHMOS GRAPHICS COPROCESSOR

- High Performance Graphics
- Fast Polygon and Line Drawing
- High Speed Character Drawing
- Advanced DRAM/VRAM Controller for Graphics Memory up to 4 Mbytes
- Supports up to 200 MHz CRTs
  — up to 640 by 480 by 8 Bits (DRAMs)
  or 1400 by 1400 by 1 Bit (DRAMs)
  or 2048 by 2048 by 8 Bits (VRAMs)
- Up to 256 Simultaneous Colors
- Integral DRAM/VRAM Controller, Shift Registers and DMA Channel
- International Character Support
- Interface Designed for Device-Independent Standards
- Hardware Windows
- Fast Bit-Block Copies Between System and Bitmap Memories
- Third-Party Software Support
- Multi-tasking Support
- Provides Support for Rapid Filling with Patterns
- Programmable Video Timing
- Advanced CHMOS Technology
- Supports Dual Port Video DRAMs & Sequential Access DRAMs
- 88 Pin Grid Array and Leadless Chip Carrier

(See Intel Packaging; Order Number: 231369)

The 82786 is a powerful, yet simple component designed for microcomputer graphics applications including personal computers, engineering workstations, terminals, and laser printers. Its advanced software interface makes applications and systems level programming efficient and straight-forward. Its performance and high-integration make it a cost-effective component while improving the performance of nearly any design. Hardware windows provide instantaneous changes of display contents and support multiple graphics applications from multiple graphics bitmaps. Applications programs written for the IBM Personal Computer can be run within one or more windows of the display when used with Intel CPUs.

The 82786 works with all Intel microprocessors, and is a high-performance replacement for sub-systems and boards which have traditionally used discrete components and/or software for graphics functions. The 82786 requires minimal support circuitry for most system configurations, and thus reduces the cost and board space requirements of many applications. The 82786 is based on Intel's advanced CHMOS III process.


A B C D E F G H J K L M N
01 Vss A07 A05 A03 A01 BLANK Vsync VCLK Vdata6 Vdata4 Vdata2 Vdata0 Vss 01
02 A09 A08 A06 A04 A02 A00 Hsync Vdata7 Vdata5 Vdata3 Vdata1 DRA0 DRA1 02
03 A11 A10 DRA2 DRQA3 03
04 A13 A12 DRA4 DRA5 04
05 A15 A14 DRA6 DRA7 05
06 A17 A16 DRA8 DRA9#/RAS3# 06
07 Vcc A18 RAS2# Vcc 07
08 A19 A20 RAS0# RAS1# 08
09 A21 CLK CAS0# CAS1# 09
10 RESET INTR WEH# WEL# 10
11 SEN MEN BEN0#/DT0 BEN1#/DT1 11
12 HLDA HREQ M/IO# CS# READY# D01 D04 D06 D08 D10 D12 D14 D15 12
13 Vss BHE# WR# RD# D00 D02 D03 D05 D07 D09 D11 D13 Vss 13
A B C D E F G H J K L M N

231676-27

Figure 1. 82786 Pinout—Bottom View

November 1988
Order Number: 231676-004

INTRODUCTION

The 82786 is an intelligent peripheral capable of both drawing and refreshing raster displays. It has an integrated drawing engine with a high level VDI like graphics commands. Multiple character sets (fonts) can be used simultaneously for text display applications. The 82786 provides hardware support for fast manipulation and display of multiple windows on the screen. It supports high resolution displays with a 25 MHz pixel clock and can display up to 256 colors simultaneously. Using multiple 82786s and/or in conjunction with dual port video DRAMs (VRAMs), the 82786 is virtually unlimited in terms of color support and resolution.

Table 1. 82786 Pin Description

Symbol	Pin Number	Type	Description
A21:0	A09,B08,A08,B07, A06,B06,A05,B05, A04,B04,A03,B03, A02,B02,B01,C02, C01,D02,D01,E02, E01, F02	I/O	Address lines for the External Bus. Inputs for Slave Mode accesses of the 82786 supported Graphics memory array or 82786 internal memory or I/O mapped registers. Driven by the 82786 when it is the External Bus Master.
D15:0	N12,M12,M13,L12, L13,K12,K13,J12, J13,H12,H13,G12, G13,F13,F12,E13	I/O	Data Bus for the 82786 Graphics memory array and the External Bus.
$\overline{\text{BHE}}$	B13	I/O	Byte High Enable. An input of the 82786 Slave Interface; driven LOW by the 82786 when it is Bus Master. Determines asynchronous vs. synchronous operation for $\overline{\text{RD}}$, $\overline{\text{WR}}$ and HLDA inputs at the falling (trailing) edge of RESET. A HIGH state selects synchronous operation.
$\overline{\text{RD}}$	D13	I/O	Read Strobe. An input of the 82786 Slave Interface; driven by the 82786 when it is Bus Master. Selects normal/test mode at falling RESET.
$\overline{\text{WR}}$	C13	I/O	Write Strobe. An input of the 82786 Slave Interface; driven by the 82786 when it is Bus Master. Selects normal/test mode at falling RESET.
M/$\overline{\text{IO}}$	C12	I/O	Memory or I/O indication. An input of the 82786 Slave Interface; driven HIGH by the 82786 when it is the Bus Master. Determines synchronous 80286 or 80186 interface at the falling edge of RESET. A LOW state selects a synchronous 80286 interface.
$\overline{\text{CS}}$	D12	I	Chip Select. Slave Interface input qualifying the access.
MEN	B11	O	Master Enable. Driven HIGH when the 82786 controls the External Bus. (i.e., HLDA received in response to a 82786 HREQ.) Used to steer the data path and select source of bus cycle status commands.
SEN	A11	O	Slave Enable. Driven HIGH when the 82786 is executing a Slave bus cycle for an External Master into the 82786 graphics memory or registers. Used to enable the data path and as a READY indication to the External Bus Master.
$\overline{\text{READY}}$	E12	I	Synchronous input to the 82786 when executing External Bus cycles. Identical to 80286 $\overline{\text{READY}}$.

Table 1. 82786 Pin Description (Continued)

Symbol	Pin Number	Type	Description
HREQ	B12	O	Hold Request. Driven HIGH by the 82786 when an access is being made to the External Bus by the Display or Graphics Processors. Remains HIGH until the 82786 no longer needs the External Bus.
HLDA	A12	I	Hold Acknowledge. Input in response to a HREQ output. Asynchronous vs. synchronous input determined by state of $\overline{\text{BHE}}$ pin at falling RESET.
INTR	B10	O	Interrupt. The logical OR of a Graphics Processor and Display Processor interrupt. Cleared with an access to the BIU Control Register.
RESET	A10	I	Reset input, internally synchronized, halts all activity on the 82786 and brings it to a defined state. The leading edge of RESET synchronizes the 82786 clock to phase 2. The trailing edge latches the state of $\overline{\text{BHE}}$ to establish the type of Slave Interface. It also latches $\overline{\text{RD}}$, $\overline{\text{WR}}$ and MIO) to set certain test modes.
CLK	BO9	I	Double frequency clock input. Clock input to which pin timings are referenced. 50% duty cycle.
$\overline{\text{CAS0}}$	M09	O	Column Address Strobe 0. Drives the CAS inputs of the even word Graphics memory bank if interleaved; identical to $\overline{\text{CAS1}}$ if non interleaved Graphics memory. Capable of driving 16 DRAM/VRAM CAS inputs.
$\overline{\text{CAS1}}$	N09	O	Column Address Strobe 1. Drives the CAS inputs of the odd word Graphics memory bank if interleaved; identical to $\overline{\text{CAS0}}$ if non-interleaved Graphics memory. Capable of driving 16 DRAM/VRAM CAS inputs.
$\overline{\text{RAS2:0}}$	M07,N08,M08	O	Row Address Strobe. Drives the RAS input pins of up to 16 DRAMs/VRAMs. Drives the first three rows of both banks of Graphics memory.
DRA9/$\overline{\text{RAS3}}$	N06	O	Multiplexed most significant Graphics memory address line and $\overline{\text{RAS3}}$. DRA9 when using 1 Mb DRAMS; $\overline{\text{RAS3}}$ otherwise.
$\overline{\text{WEL}}$	N10	O	Write Enable Low Byte. Active LOW strobe to the lower order byte of Graphics memory.
$\overline{\text{WEH}}$	M10	O	Write Enable High Byte. Active LOW strobe to the higher order byte of Graphics memory.
DRA8:0	M06,N05,M05, N04,M04,N03, M03,N02,M02	O	Multiplexed Graphics memory Address. Graphics memory row and column address are multiplexed on these lines. Capable of driving 32 DRAMs/VRAMs.
$\overline{\text{BEN1:0}}$ DT1:0	N11,M11	O	Multiplexed Bank Enable and Data Transfer Line. In normal memory cycle enables the output of the Graphics memory array on to the 82786 data bus, D15:0. In data transfer cycle, loads the serial register in dual port video DRAMs (VRAMs). $\overline{\text{BEN1}}$/DT1 and $\overline{\text{BEN0}}$/DT0 control Bank1 and Bank0 respectively.
BLANK	F01	I/O	Output used to blank the display at particular positions on the screen. May also be configured as input to allow the 82786 to be synchronized with external sources.

Table 1. 82786 Pin Description (Continued)

Symbol	Pin Number	Type	Description
$V_{DATA}7:0$	H02,J01,J02, K01,K02,L01, L02,M01	O	Video data output.
V_{CLK}	H01	I	Video Clock input used to drive the display section of the 82786. Maximum frequency of 25 MHz.
H_{SYNC}/WS0	G02	I/O	Horizontal Sync. Window status is multiplexed on this pin. May also be configured as input to allow the 82786 to be synchronized with external sources. May also be configured to output Window status.
V_{SYNC}/WS1	G01	I/O	Vertical Sync. Window status is multiplexed on this pin. May also be configured as input to allow the 82786 to be synchronized with external sources. May also be configured to output Window status.
V_{SS}	A01,N01,A13, N13		4 V_{SS} pins.
V_{CC}	N07,A07		2 V_{CC} pins.

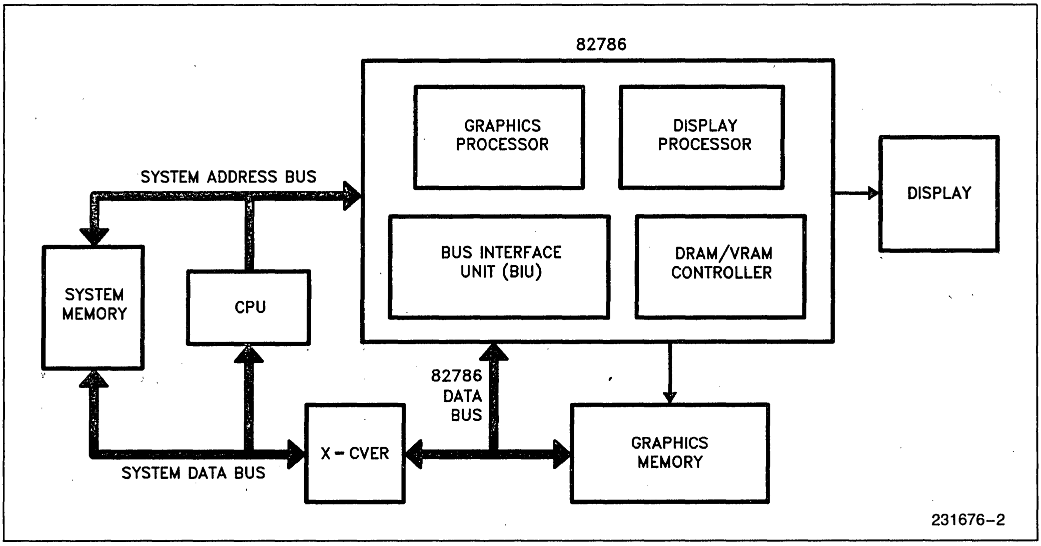


Figure 2

ARCHITECTURE

The 82786 is a high integration device which contains three basic modules (figure 2):

1. Display Processor (DP)
2. Graphics Processor (GP)
3. Bus Interface Unit (BIU) with DRAM/VRAM Controller.

Display Processor

The 82786 Display Processor controls the CRT timings and generates the serial video data stream for the display. It can assemble several windows on the screen from different bitmaps scattered across the memory accessible to the 82786.

Graphics Processor

The 82786 Graphics Processor executes commands from a Graphics Command Block (GCMB) (placed in memory by the host CPU) and updates the bitmap memory for the Display Processor. The Graphics Processor has high level VDI like commands and can draw graphical objects and text at high speeds.

Bus Interface Unit (BIU)

The BIU controls all communication between the 82786, external CPU and memory. The BIU includes an integrated DRAM/VRAM controller that can take advantage of the high speed burst access modes of page mode and fast page mode DRAMs to perform block transfers. The Display Processor and Graphics Processor use the BIU to access the bitmaps in memory.

Memory Structure and Internal Registers

The 82786 address range is 4 Mbytes. This is divided between the graphics memory directly supported by the 82786 and external system memory. The 82786 distinguishes between graphics memory and external system memory by assuming graphics memory space starts at address 0H and goes up to whatever amount of graphics memory is configured. External system memory occupies the rest of the address space. The amount of graphics memory configured, and therefore the graphics memory/external system memory boundary, is controlled by the "DRAM/VRAM Control Register" in the BIU. The upper limit of configured graphics memory is 4 Mbytes.

A 128 byte block (contiguous) of internal control registers is distributed throughout the three modules on the 82786. This block can be either memory or I/O mapped in the CPU address space. The base address and memory or I/O mapped for this register block is programmable through the "Internal Relocation Register" in the BIU.

External Memory Access (Master Mode)

The 82786 initiates "Master Mode" whenever it needs to access a memory address that is beyond the upper limit of configured graphics memory. This memory is typically external memory shared between the 82786 and the external CPU. The bus timings in this mode are similar to the 80286 style bus timings.

An 82786 request for the bus is indicated by a high level on the HREQ line. The 82786 drives the external bus (A21:0, D15:0, $\overline{RD}$, $\overline{WR}$, MIO and $\overline{BHE}$) only after receiving a HLDA (acknowledge) from the external master. The HLDA line could be externally synchronized (82786 synchronous mode) or internally synchronized (82786 asynchronous mode). The 82786 will deactivate the HREQ line when it no longer needs to access external memory or when it senses an inactive HLDA. The 82786 indicates that it is in control of the external bus by a high level on the MEN output. Screen corruption can occur if the master mode bus cycle, including HREQ/HLDA latency, is too long.

Slave Mode

The 82786 Slave Interface allows an external CPU access into the graphics memory array or the 82786 Internal Registers. The external CPU directs a (read/write) slave access to the 82786 by asserting the 82786 $\overline{CS}$ input. When the 82786 is not driving the external bus, the A21:0, $\overline{RD}$, $\overline{WR}$, MIO and $\overline{BHE}$ lines are inputs. The $\overline{RD}$, $\overline{WR}$, MIO and $\overline{CS}$ lines are constantly monitored by the 82786 to detect a CPU cycle directed at the 82786. After beginning a slave access to the 82786, the external CPU must go into a wait state. The 82786 will not process new slave commands from the CPU before the previous command has been serviced. The 82786 initiates a slave access by a high level on the SEN output and terminates the slave access when SEN is low. The data bus transceivers can be enabled by SEN.

SEN as Slave Ready Indication

Inverted SEN should be connected to the 82284 ARDY input when the Slave Interface is set in synchronous 80286 mode. The number of wait states for a read cycle is a function of the DRAM/VRAM speed. Write cycles execute with 2 wait states because the 82786 issues SEN with different timing during write cycles.

The 82786 supports byte accesses to graphics memory. The combination of $\overline{BHE}$ and A0 generate the proper $\overline{WEL}$ and $\overline{WEH}$ signals. $\overline{BHE}$ and A0 are ignored for read cycles. Since the Display and Graphics Processors always generate word addresses, the slave cycles directed to graphics memory are the only time $\overline{WEL}$ may not exactly follow $\overline{WEH}$.

The 82786 will acknowledge a slave access from an external CPU while waiting for an acknowledge (HLDA) to its own request for the external bus. This prevents a potential deadlock situation.

Synchronous/Asynchronous Operation

The synchronous/asynchronous mode is selected by the state of the $\overline{BHE}$ input at the falling edge of reset. A high state selects synchronous operation. The synchronous interface requires that the 82786 and the 80286/80186 clock inputs are shared. For the 80286 case, a common RESET ensures that the 82786 and the 80286 initialize to the same state. With the 80186, external hardware must ensure that the 82786 phase1 is coincident with the 80186 CLKOUT LOW. In the Master Mode, the HLDA line is sampled synchronously or asynchronously. The 82786 slave interface provides for synchronous or asynchronous sampling of the command lines ($\overline{RD}$ and $\overline{WR}$).

EIGHT AND SIXTEEN BIT HOST

On reset, the 82786 always assumes an 8 bit host interface. The first few accesses to the 82786 must be 8 bit accesses. The 82786 can be switched to a 16 bit interface by setting the "BCP" bit in the "BIU Control Register".

In 16 bit mode, the Internal Register Block is only word addressable. Odd word or odd byte accesses to the internal locations will not produce the desired result. Even byte access, however will work as desired. The least significant address bit, A0, is ignored in 16 bit mode.

In 8 bit mode, the internal registers must be accessed by two successive bus cycles. This is not necessary for reads, but is necessary for writes to the internal registers. The low byte (A0 = 0) must be written first, followed by the high byte (A0 = 1) of the register. A21:1 must be the same for both bus cycles. The register is not changed until the second byte (the high byte) is written to the 82786. There is no restriction on the time between the two bus cycles, but if successive low bytes are written before a high byte is written, the last low byte is the one written to the register. The BIU latches even bytes (A0 = 0) of write data in a temporary register. When an odd byte is subsequently written to location address + 1, this byte and the even byte in the temporary register are written to the desired location. A lock out mechanism prevents a high byte write to modify an internal register if there is no valid word in the temporary register.

There is no crossing done by the 82786 in 8 bit mode: low bytes are transferred on the low data lines D7:0 and the high bytes on D15:8. An external crossover creates the 8 bit bus for the host. This is not additional hardware since a crossover is needed for an 8 bit host accessing of the memory array anyway.

MEMORY ACCESS ARBITRATION

The BIU receives requests to access the graphics memory from the Display Processor, the Graphics Processor and the External CPU. Additionally the internal DRAM/VRAM Controller also generates refresh requests. The DRAM/VRAM refresh requests are always highest priority. The other requests are arbitrated with programmable priorities. A higher priority request can interrupt lower priority memory cycles. Block transfers however can only be interrupted on doubleword boundaries.

There are two priority levels for requests from the Display and Graphics Processors: a First Priority (FPL) and a Subsequent Priority (SPL). The First Priority is the priority at which the first request of a bus cycle is arbitrated with. The Subsequent Priority is the priority associated with subsequent requests of a block transfer bus cycle. This allows for block transfers to execute with a different priority level. If a higher priority request occurs while a block transfer is executing, the BIU suspends the current block transfer and acknowledges the higher priority request. After completion of that higher priority memory access, the requests are arbitrated again. The suspended block transfer is arbitrated with its SPL priority since it is still executing a block transfer. The External Request has no Subsequent Priority level since it cannot execute block transfers. It does have an Altered Priority, though, which is the priority it assumes once every 42 CLK's (maximum bus latency for IBM PC's). The default priorities from highest to lowest following RESET are:

External	APL	7
External	FPL	7
Display	FPL	6
Graphics	FPL	5
Displays	SPL	3
Graphics	SPL	2

Three bits describe the priorities; 7 is the highest and 0 is the lowest. If two priority registers are programmed with the same value, a default priority chain is used. The default order is, from highest to lowest priority:

1. Display Processor
2. Graphics Processor
3. External

Graphics Memory Interface

The 82786 directly supports up to 32 DRAMs without additional external logic. This capability allows the use of cost effective memory devices and can result in significant performance improvement through the use of either standard Page Mode or the newer Fast Page Mode/Static Column Decode sequential access RAMs. The Fast Page Mode/Static

Column Decode parts enable the 82786 to cycle the DRAMs in 100 ns instead of the 200 ns used for Page Mode parts. The 82786 also allows the memory to consist of either standard single port memory devices or dual port Video RAM devices (VRAMs).

The 82786 supports a wide range of DRAM/VRAM configurations. The choices include interleaving or non-interleaving (1 or 2 banks - one CAS line/bank), number of rows per bank (1, 2, 3 or 4 - one RAS line/row), width (x1, x4 or x8), height (16k, 64k, 256k or 1M) and performance (Page Mode or Fast Page Mode/Static Column Mode). The only limitation is the address space limit of 4Mbytes. The 82786 DRAM/VRAM address lines (DRAx) can directly drive 32 memory devices while the RAS, CAS, WE and BEN lines can directly drive 16 devices. When the memory array consists of more than 32(16) devices then external drivers must be used to drive the memory array.

DRAMs with a HEIGHT of 64k are not allowed in the graphics memory of a system in which Master Mode will be used. There is no limitation on the total DRAM density (64k x 4 DRAMs cannot be used; 256k x 1 DRAMs are okay).

There are some special DRAM configurations:

i) When 1 Mb x 1 DRAMs are used, $\overline{RAS3}$ is used as DRA9.

ii) When only one interleaved row is configured (32 devices), $\overline{RAS1}$ is identical to $\overline{RAS0}$. Additional buffering on $\overline{RAS0}$ is therefore not required.

iii) When two non interleaved rows are configured (32 devices), $\overline{CAS1}$ is identical to $\overline{CAS0}$. Additional buffering on $\overline{CAS0}$ is therefore not required.

DRAM Cycle Types

The 82786 supports two fundamental memory cycle types: single and block. A single cycle involves a single 16 bit word, while a block transfer is a minimum of 2 16 bit words with no maximum length. The single cycle types supported and their cycle times are given below. The cycle times are counted in system clocks, ½ the CLK input frequency.

1. Single Reads	3 cycles	300 ns @ 10 MHz
2. Single Writes	3 cycles	300 ns @ 10 MHz
3. Read-Modify-Writes	4 cycles	400 ns @ 10 MHz

The block cycles use the high speed sequential access modes of page mode, fast page mode (ripple mode) and static column DRAMs. Typical performance numbers for this case are:

1. Page Mode, Non-Interleaved	2 cycles 10 Mb/s @ 10 MHz
2. Page Mode, Interleaved	1 cycle 20 Mb/s @ 10 MHz
3. Fast Page Mode, Non-Interleaved	1 cycle 20 Mb/s @ 10 MHz
4. Fast Page Mode, Interleaved	.5 cycles 40 Mb/s @ 10 MHz

All accesses into the graphics memory by the Display Processor use the high speed sequential access mode whenever possible. The Graphics Processor uses a single Read-Modify-Write cycles for all pixel drawing operations. Block copy operations by the Graphics Processor use the high speed sequential access modes. External CPU access into graphics memory is always a single read or write cycle. When configured to interface with dual port VRAMs, the 82786 generates Page Mode and Fast Page Mode style control signals for memory access through the normal random access port. It also executes a data transfer cycle when the video shift register in the VRAMs have to be loaded.

Graphics Memory Refresh

The BIU has an internal DRAM/VRAM refresh controller. The refresh period is programmable through the "DRAM/VRAM Refresh Control" Register in the BIU. All configured rows are refreshed simultaneously by activating the corresponding $\overline{RAS}$ lines periodically ($\overline{RAS}$ only refresh). The refresh row address (10 bits) is generated internally. On power up, the refresh row address is undefined. On normal reset, the refresh row address is not affected. It is initialized only if the 82786 is reset into the "BIU Test Mode". Not modifying the refresh address during RESET allows for a "warm RESET" implementation: contents of DRAM/VRAM can be insured to remain valid if RESET is short enough (less than three DRAM/VRAM refresh cycles). DRAM/VRAM refresh will continue at the proper row after RESET goes inactive again.

The graphics memory refresh cycles are always the highest priority cycles. There is some latency possible between the internal refresh request and the actual refresh cycle. This latency is critical only in one case: The 82786 is in a wait state while executing a bus cycle on the External Bus.

The worst case is a refresh request occurring just after the 82786 receives a HLDA from the host CPU to execute a block transfer on the external bus. Refresh requests can interrupt block transfers, but only on doubleword boundaries — the 82786 must execute 2 full bus cycles on the external bus before the refresh cycle is run. The possibility of many wait states when executing these two bus cycles creates a need for a large refresh latency tolerance. The 82786 can queue up to 3 refresh requests internally.

In the default mode (a refresh request occurs every 15.2 micro seconds and at 10 MHz operation), this implies that each bus cycle to external memory should not have more than 225 wait states.

There is no warm up logic on the 82786. The system must either wait for sufficient number of refresh cycles to execute or the boot software on the host can quickly access the memory array for the required number of cycles.

The default value of the DRAM/VRAM Control Register configures the array as 4 rows of Non-Interleaved Page Mode 256k × 1 with refresh requests generated every 15.2 micro seconds.

Internal Register and Graphics Memory Slave Access

The external master can access either 82786 internal memory I/O mapped registers or the Graphics memory. The 82786 internal address space consists of a contiguous 128-byte block that starts on an even byte address. It is mapped to memory or I/O space depending on the state of the M/$\overline{IO}$ bit in the "Internal Relocation Register". If the M/$\overline{IO}$ bit is set to one, the Internal Register Block is memory mapped. If the M/$\overline{IO}$ bit is zero, the Internal Register Block is I/O mapped. An address comparison is done between the Internal Relocation Register and the incoming address to determine if the CPU access is directed to internal memory/IO mapped registers.

Intel reserves the right to add functions to future versions of the 82786. Users should not use reserved locations in order to ensure future compatibility.

PERFORMANCE

Slave performance is measured here by assuming a request is made to an idle 82786. A synchronous interface is assumed.

Minimum 80286 Wait States = 3
(10 MHz 80286 and 82786)

Minimum 80386 Wait States = 8
(16 MHz 80386, 8 MHz 82786)

Minimum 80186 Wait States = 3
(10 MHz 80186 and 82786, WT = 1)

Minimum 80186 Wait States = 2
(10 MHz 80186 and 82786, WT = 0)

The values mentioned above are for read cycles. In some cases, write cycles can operate with fewer wait states. For instance, the 80286 can execute 2 wait state synchronous write cycles. The 80186 can execute write cycles with one less wait state than mentioned above.

For asynchronous interfaces, if the CPU is operating at the same frequency as the 82786, the number of wait states are typically 1 more than those indicated above. For CPUs operating at a slower frequency than the 82786, the number of wait states are, on the average, less than 1 greater than those given above. In some cases, (eg. a 6 MHz 80286) an asynchronous interface acutally has less wait states than those quoted above for the synchronous interface.

INTERNAL REGISTERS

The 82786 Internal Register block is relocatable by programming an even byte address in the "Internal Relocation Register" in the BIU. The register block can be memory or I/O mapped based on the state of the M/$\overline{IO}$ bit. The Register Block is physically distributed between the three 82786 modules, BIU, Graphics Processor and Display Processor.

Accesses to reserved locations have no effect; they execute normally but may produce indeterminate read data. No register is altered when a write is executed to a reserved location.

Location of Internal Registers within 128 byte block:

Byte Address			
'00-0F H		BIU Registers	8 words
'10-1F H		reserved	
'20-2B H		GP Registers	6 words
'2C-3F H		reserved	
'40-4A H		DP Registers	5 words
'4B-7F H		reserved	

The BIU register map is as follows:

Byte Address	
BASE + 0H	Internal Relocation
BASE + 2H	Reserved
BASE + 4H	BIU Control
BASE + 6H	Refresh Control
BASE + 8H	DRAM Control
BASE + AH	Display Priority
BASE + CH	Graphics Priority
BASE + EH	External Priority

The field definitions for the BIU Registers are as follows:

Internal Relocation

	15	14	13	12	···	4	3	2	1	0
Addr = BASE + 0H	Base				Address					MIO

Reset values: xxx0

The Base Address determines the location of the 128 byte Internal Register Block. The MIO bit selects between memory or I/O mapping. If it is set (1), the Register Block is memory mapped. At RESET, the Base Address is set so that the Internal Relocation Register is located at every 128-byte address in the entire I/O space whenever $\overline{CS}$ is asserted. The Base Address must be written into this register before any other registers can be accessed.

BIU Control

	6	5	4	3	2	1	0
Addr = BASE + 4H	VR	WT	BCP	GI	DI	WPI	WP2
RESET value:	0	1	0	0	0	0	0

VR: If set (1) then the 82786 generates dual port video DRAM (VRAM) type memory cycles for display data fetch. If reset (0) then conventional page mode type memory cycles are performed to fetch display data.

WT: Determines the minimum number of wait states possible in a synchronous 80186 interface. If set (1), there is a minimum of 2 (3) wait states during memory write (read) cycles.

BCP: Determines whether the Internal Register block is accessed as bytes or words by the external CPU. If set (1), a 16 bit interface is selected.

GI: Graphics Processor Interrupt. Set when the Graphics Processor issues an Interrupt. Cleared with RESET or a read of this register.

DI: Display Processor Interrupt. Set when the Display Processor issues an Interrupt. Cleared with RESET or a read of this register.

WP1: Write Protection One. When set (1), all BIU Register contents except for the WP1 and WP2 bits of this register are write protected.

WP2: Write Protection Two. When set (1), all BIU Register contents are write protected, including WP1 and this bit, WP2. The only way to regain write access to the BIU registers after this bit is set, is to RESET the 82786.

Refresh Control

	6	5	4	3	2	1	0
Addr = BASE + 6H		Refresh Scalar					
RESET value:		0	1	0	0	1	0

The Refresh Scalar is a 6 bit quantity that determines the frequency of refresh cycles to the Graphics memory.

Refresh interval = (Scalar + 1) * 16 * Input clock period

DRAM/VRAM Control

	6	5	4	3	2	1	0
Addr = BASE + 8H	RW1	RW0	DC1	DC0	HT2	HT1	HT0
RESET value:	1	1	0	0	1	0	1

RW1:0: Number of Graphics memory Rows. One of the variables in defining the Graphics memory/External system boundary. Also disables RAS signals not driving any DRAMs/VRAMs.

RW1	RW0		
0	0	:	1 Rows
0	1	:	2 Rows
1	0	:	3 Rows
1	1	:	4 Rows

DC1:0: DRAM/VRAM Configuration. Controls the rate of block transfers and orientation of CAS1 and CAS0.

DC1	DC0		
0	0	:	Page Mode, Non-Interleaved
0	1	:	Page Mode, Interleaved
1	0	:	Fast Page Mode, Non-Interleaved
1	1	:	Fast Page Mode, Interleaved

HT2:0: DRAM/VRAM Height. Defines the HEIGHT (not size) of each DRAM/VRAM chip in the system. All DRAMs/VRAMs must be the same size.

HT2	HT1	HT0		
0	0	0	:	8K Devices
0	0	1	:	16K Devices
0	1	0	:	32K Devices
0	1	1	:	64K Devices
1	0	0	:	128K Devices
1	0	1	:	256K Devices
1	1	0	:	512K Devices
1	1	1	:	1M Devices

Display Processor Priority

	6	5	4	3	2	1	0
Addr = BASE + AH		FPL			SPL		
RESET value:		1	1	0	0	1	1

Graphics Processor Priority

Addr = BASE + CH	FPL			SPL		
RESET value:	1	0	1	0	1	0

External CPU Priority

Addr = BASE + EH	FPL			APL		
RESET value:	1	1	1	1	1	1

Specifies the priorities of the Display Processor, Graphics Processor and External CPU requests for the first request (FPL) and subsequent requests for block transfers (SPL). Code 111 is highest priority. Code 000 is lowest priority.

RESET AND INITIALIZATION

The state of $\overline{\text{BHE}}$ at trailing RESET determines synchronous vs. asynchronous operation. In Master mode, synchronous/asynchronous operation affects the sampling of the HLDA signal only. In Slave mode, synchronous/asynchronous operation affects the sampling of $\overline{\text{RD}}$/$\overline{\text{WR}}$ signals. Synchronous operation is set if $\overline{\text{BHE}}$ is sensed HIGH at trailing RESET. This enables direct connection of the 80286 $\overline{\text{BHE}}$ pin in synchronous systems since it is driven HIGH during RESET. The 80186 "tristates" its $\overline{\text{BHE}}$ during RESET so a small static load on this line can select asynchronous operation.

All internal registers are set to their default values on reset. The first slave I/O write access to the 82786 will always be directed at the Internal Registers (ignoring the upper fifteen address bits). The Internal Relocation Register must be programmed before any other Internal registers can be accessed. The DRAM/VRAM configuration registers must also be programmed to conform to any specific environment.

The 82786 assumes an 8 bit external CPU interface on reset. The graphics memory interface is always 16 bits wide. The BCP bit in the "BIU Control Register" must be set to 1 to enable a 16 bit external interface. Interrupts are cleared on reset.

GRAPHICS PROCESSOR

Introduction

The Graphics Processor (Graphics Processor) is an independent processor within the 82786. Its primary task is to draw bitmap graphics. It executes commands residing in the memory, accessing the memory through the Bus Interface Unit (BIU). The Graphics Processor addresses 4 MB of linear memory (22 bit addresses).

The Graphics Processor draws into a predefined area in the memory which is referred to as a "bitmap". A bitmap can be thought of as a rectangular drawing area composed of pixels. A coordinate system is defined for this bitmap with the origin at the upper left corner, the x-coordinate increasing from left to right and the y-coordinate increasing from top to bottom. A bitmap can be up to 32K pixels wide and 32K pixels high.

The 82786 can draw several graphics primitives such as points, lines, arcs, circles, rectangles, polygons and characters. During the figure drawing process, the 82786 follows several programmable attributes.

The graphics attributes supported by the 82786 are:

color
depth (bits/pixel)
texture
logical operation
color bit mask
clipping rectangle

Each graphics primitive can be drawn in any one of 2, 4, 16 or 256 "Colors". The color details (bits/pixel and exact color) are programmable. The "Texture" controls the appearance of any line (or figure). The texture pattern can be up to 16 bits long thus enabling drawing of solid, dashed, dotted, dot-dash etc. types of lines. Each bit in the Texture corresponds to one pixel. The 82786 supports all sixteen binary "Logical Operation" between a figure being drawn in bitmap memory and the existing contents of memory. It is thus possible to overlay a figure on a background. The "Color Bit Mask" restricts the drawing operation to only some "color planes". The clipping rectangle restricts the drawing operation to a specific area in the bitmap.

The pixel information is stored in the bitmap memory in a packed pixel format. Different color bits for the same pixel are stored in adjacent bit positions within the same byte. Each byte represents 1, 2, 4 or 8 pixels (in one of 256, 16, 4 or 2 colors).

The Graphics Processor fetches its commands directly from a linked list Memory-resident Graphics Processor Command Block (GCMB). The GCMB is created and maintained by the CPU. The initial address for the GCMB is contained in a Graphics Processor Opcode Register in the 82786. Addresses for subsequent (next) GCMBs are contained in the previous GCMBs. The Graphics Processor can be forced to stop by appropriate commands.

When the Graphics Processor is idle, it is said to be in the "Poll State". This is the default mode after reset. While in the Poll State, the Graphics Processor continuously monitors its internal "Opcode Register". A valid command in this register starts the

Graphics Processor. The first command placed in the internal Opcode Register must always be a "LINK" command directing the Graphics Processor to the main GCMB in memory.

Address	Register	Function	
BASE + 20h	GR0	OPCODE	GECL
BASE + 22h	GR1	Parameter 1 (Link Address Lower)	
BASE + 24h	GR2	Parameter 2 (Link Address Upper)	

Graphics Processor Internal Registers used in Poll State

Graphic Processor Command Format

The commands are placed (along with their parameters) sequentially in memory. Several GCMBs may be linked together through a LINK command. All commands have a standard format as described below:

15 8	7 1	0
OPCODE	0 0 0 0 0 0 0	GECL
Parameter 1		
● ● ●		
Parameter n		
etc.		

Each command to the Graphics Processor consists of an opcode, a Graphics End of Command List (GECL) bit and a list of parameters as required by the command. The opcode is 8-bits wide. The remaining 7-bits in the first word of the command must be all zeroes to ensure future compatibility. Also, whenever a parameter for the command is an address, 32-bits have been set aside but the 82786 uses only 22-bit addresses. The user must ensure that the higher 10-bits in the address parameter are always all zeroes All commands must lie at even byte addresses.

After fetching each command, the Graphics Processor checks the GECL bit. If the GECL bit is zero, the command executes and the next command is then fetched from the GCMB. If the GECL bit is set to one, the Graphics Processor does not execute the command and enters a POLL state.

Graphics Processor Status Register

One of the 82786 Internal Registers contains the Graphics Processor Status Byte. The bits in the Status Byte are represented as:

Address BASE + 26H	GPOLL	GRCD	GINT	GPSC	GBCOV	GBMOV	GCTP	GIBMD

1. GPOLL - Poll State

   Indicates if the Graphics Processor is in a POLL state.

2. GRCD - Reserved Command

   This bit is set if the Graphics Processor encounters an illegal opcode.

3. GINT - This bit is set as a result of the INTR__GEN command.

4. GPSC - Pick Successful

   This bit is set or cleared while the Graphics Processor is in the PICK mode. The bit is set if the pick operation resulted in success on any command.

5. GBCOV - bitmap Overflow for BitBlt or CharBlt

   An attempt to execute a CHAR or a BitBlt command with any portion of the destination rectangle lying outside the clip rectangle causes this bit to be set.

6. GBMOV - bitmap Overflow for Geometric Commands

   An attempt to draw a pixel lying outside clip rectangle as a result of any geometric drawing commands (LINE, CIRCLE etc.) causes this bit to be set. The reason for separating these two bits is the difference between the clipping operations for the two types of commands.

7. GCTP - Character Trap

   This bit indicates that a character specified in the character string as a parameter for the CHAR command had its TRAP bit set.

8. GIBMD - Illegal Bit Map Definition

   This bit is set if the DEF__BIT__MAP command is executed with illegal parameters. The illegal parameters are bits per pixel defined to be other than 1, 2, 4 or 8, Xmax defined to be greater than 32k-1, or the following equation not being met: ((Xmax + 1) * Bpp) mod 16 = 0.

All the status bits except GPOLL are cleared upon reset. The GPOLL bit is set on reset.

Graphics Instruction Pointer

The Graphics Processor Instruction Pointer is a 22 bit quantity stored in two registers in the Graphics processor. It points to the current command in the GCMB.

Address	Register	Function
BASE + 28h	GCIPL	Instruction Pointer Lower
BASE + 2Ah	GCIPH	Instruction Pointer Upper

Clipping Rectangle

The 82786 can be instructed to restrict drawing to certain portion of the bitmap only. This portion is called the "Clipping Rectangle". The default clipping rectangle is the entire bitmap. The clipping rectangle must be redefined after a DEF__BIT__MAP command. For figures that are partially inside and partially outside the clipping rectangle, only the part inside the clipping rectangle is updated in the bitmap. Character clipping is supported for word mode.

In order for the clipping to have predictable results, there are some restrictions on the x,y coordinates of each pixel. The rules to be observed are:

1. For lines, circles, polygons, polylines, BitBlts and CharBlts, each pixel lying on the figure (both the visible and the invisible parts) must not have its x or y coordinate outside ± 32K range.
2. For circular arcs, the above restriction applies to the circle of which the arc is a part.
3. If the top of characters are to be clipped, the top row of pixels of the character must begin on an even-numbered scan line (scan lines are numbered from top to bottom beginning with number 0).

Pick Mode

The Graphics Processor can be put in "PICK Mode" by executing the ENTER__PICK command. In the PICK Mode, the Graphics Processor performs all pixel computations for all drawing, BitBlt and Character commands. However, the bitmap memory is not updated. Instead every computed pixel is compared against the clipping rectangle. If any computed pixel is found to lie within the clipping rectangle, the GPSC bit in the Graphics Processor Status Register is set. Pick mode is not supported for circles and arcs.

Character Font Storage

The character fonts are stored in memory. Starting from an even address, the character information is stored in consecutive words of memory forming a character block. Each block can be of different lengths for different characters. A character font is selected by programming its base address into the 82786 through the DEF__CHAR__SET command. The font could be established for 8 or 16 bit character codes. Each character block within a font has the following format:

15	8	7	0
S	Width W	T	Height H
Dot Pattern for 1st row			
Dot Pattern for 2nd row			
—			
Dot Pattern for Hth row			

S - Character Space bit T - Trap Bit

Each character block must start at a word address and the dot patterns for each line of the font must reside in separate words. The height and width of each character cannot be more than 16 pixels. In case the width of a character is less than 16 pixels, the dot pattern for each line must be stored as right justified within the word.

Note that width and height of the character refer to the difference between their limiting x and y coordinates respectively. Thus width = 0 specifies a character one pixel wide and a height = 0 specifies a character one pixel high.

Graphics Processor Control and Context Registers

All Control and Context Registers in the Graphics Processor can be read from or written into, through the Graphics Processor commands DUMP__REG and LOAD__REG. Each register is identified by a 9-bit Register Id.

These registers are not directly addressable like the registers that are mapped into the 82786's On-Chip-Memory (I/O) space, i.e., they are accessible only through the DUMP__REG and the LOAD__REG commands. The four user accessible graphics control registers are listed below.

REGISTER NAME	REGISTER—ID (# of bits)	REGISTER FUNCTION
GPOEM	0003 (6)	Poll Mask
GIMR	0004 (8)	Interrupt Mask
GSP	010C (21)	Stack Pointer
GCNT	0015 (16)	Character Count while drawing characters in bitmap

The Graphics Processor also has Context Registers, which are normally of no use to a user except in the event of saving and restoring them during a CPU context switch. Any other direct access to these registers must be avoided.

Name	ID	Bits	Function
GCOMM	0002	(16)	Command
GCHOR	0007	(2,2)	Character Orientation and Path*
GCHA	010B	(21)	Character Font Base Address~
GCA	010D	(21)	Memory Address of Current Position (X, Y)~
GBORG	010F	(21)	Bitmap Origin Address~
GCX	0010	(16)	Current X Position
GCY	0011	(16)	Current Y Position
GPAT	0012	(16)	Line Pattern
GSPAC	0013	(16)	Spacing between Characters and Bitblts
GN	0016	(16)	Number of 16-Bit Words Spanning Width of Bitmap
GVERS	0017	(16)	Version Number*** (D Step Value = 5)
TEMP	0019	(16)	Temporary Storage
GXMAX	0090	(16)	Maximum X for Clipping Rectangle
GYMAX	0091	(16)	Maximum Y for Clipping Rectangle
GXMIN	0094	(16)	Minimum X for Clipping Rectangle^
GYMIN	0095	(16)	Minimum Y for Clipping Rectangle^
GMASK	0099	(16)	Pixel Mask
GBGC	009B	(16)	Background Color
GFGC	009C	(16)	Foreground Color
GFCODE	009E	(4)	Function Code^^
GCIP	01AC	(21)	Current Instruction Pointer~
GBPP(RO)	009F	(4)	Used with Dump Register Command to Get Current Bits per Pixel Value^^^
GBPP(WO)	0008	(4)	Used with Load Register Command to Write Current Bits per Pixel Value^^^

~21-bit registers use 2 consecutive words.

*These bits are right justified in each byte of the word in which they are stored. Two bits are stored in bits 1 and 0 and two bits are stored in bits 8 and 9; the remaining upper bits in each byte are zeroed.

***In D-Step, valid after RESET and prior to drawing or drawing control commands.

^Correction to previous GXMIN ID 0096 and GYMIN 0097 assignments.

^^GFCODE ID reassigned from 001C to 009E in D-Step. This code is read out inverted. The value read out must be inverted in order to restore it properly.

^^^New D-Step Bpp Registers.

NOTE:
The following information is not saved by saving the state of these registers:

1) Type of character font, word, or byte.
2) Whether or not you are in pick mode.
3) Transparent or opaque drawing.

Graphics Processor Exception Handling

The status bits GPOLL, GRCD, GINT, GPSC, GBCOV, GBMOV, GCTP, and GIBMD are capable of generating an interrupt to the CPU depending upon the Interrupt Mask Register (GIMR). If the corresponding bit in the GIMR is a "0" an interrupt is generated. If another bit in the Graphics Processor Status Register is set before an acknowledgement for a previously generated interrupt, then another interrupt is not generated. Reading the Status Register and the BIU Control Register serves the purpose of an Interrupt Acknowledge to the Graphics Processor. Reading the Graphics Processor Status Register clears the offending status bit(s) - bits not masked out in the Interrupt Mask. If the interrupt is generated due to the GPOLL bit, then this bit is not cleared on an interrupt acknowledge. However this does not generate repeated interrupts.

The status bits GINT, GPSC, GBCOV, GBMOV, GTRP and GIBMD can also cause the Graphics Processor to stop its normal instruction fetch/execution and enter the POLL state. This is determined by the contents of the POLL On Exception Mask register (GPOEM). The GPOEM is 6 bits wide. If the corresponding bit in the GPOEM is a "0", POLL state is entered. On entering POLL state, the GECL bit in the Opcode (GR0) register is automatically set. When the Graphics processor is in POLL state, it can be restarted by writing the appropriate opcode into the Opcode register (GR0) and writing a zero into the GECL bit. The act of clearing the GECL bit also causes the status bits that caused the POLL state to be cleared. Interrupt generation due to the GPOLL bit is enabled on exit from the POLL state.

The status bit GRCD when set, always causes the Graphics Processor to enter the Poll State. The Interrupt and the POLL mechanisms are two independent mechanisms. It is possible for the Graphics Processor to issue an interrupt and not POLL, or to issue an interrupt and POLL, or not to issue an interrupt and POLL or do none of them - all depending upon the GIMR and GPOEM Registers.

Initialization And Software Abort

The ABORT signal causes the Graphics Processor to enter POLL state after the execution of the currently executing command.

The two ways to initiate a software ABORT and force the Graphics Processor to enter POLL state are:

i) An attempt to write into the Graphics Processor Status Register

ii) An attempt to write into the Graphics Current Instruction Pointer.

Upon RESET, the Graphics Processor immediately enters a well defined state. The following events take place:

1. Command execution is halted and the Graphics Processor enters POLL state.
2. The GECL bit of the Opcode register (GR0) is set to one to indicate an End of Command List.
3. All status bits except GPOLL are cleared. GPOLL is set.
4. Interrupt Mask Register (GIMR) is set to all ones (disabled).
5. Poll on Exception Mask register (GPOEM) is set to all ones (disabled).
6. Graphics Processor exits pick mode.

The Graphics Processor command set is divided into the following classes:

1. Non-Drawing Commands
2. Drawing Control Commands
3. Geometric Commands
4. Bit Block Transfer (BitBlt) Commands
5. Character Block Transfer (CharBlt) Commands

List of Graphics Processor Commands
(Higher Byte - Hex)

Command	Opcode	Command	Opcode
LINK	02	POINT	53
NOP	03	LINE	54
DEF_TEXTURE_OPAQUE	06	LINE_OE	55
DEF_TEXTURE_TRANSPARENT	07	RECT	58
DEF_CHAR_SET_WORD	0A	BIT_BLT	64
DEF_CHAR_SET_BYTE	0B	ARC_EXCLUSION	68
INTR_GEN	0E	ARC_INCLUSION	69
CALL	0F	POLYGON	73
RETURN	17	POLYLINE	74
DEF_BIT_MAP	1A	CIRCLE	8E
DUMP_REG	29	CHAR_OPAQUE	A6
LOAD_REG	34	CHAR_TRANSPARENT	A7
DEF_COLOR	3D	CHAR_OPAQUE/REVERSE	A8
DEF_LOGICAL_OP	41	CHAR_TRANSPARENT/REVERSE	A9
ENTER_PICK	44	BIT_BLT_M	AE
EXIT_PICK	45	INCR_POINT	B4
DEF_CLIP_RECT	46	HORIZ_LINES	BA
DEF_SPACE	4D	BIT_BLT_EO	D4
DEF_CHAR_ORIENT	4E	BIT_BLT_ET	D5
ABS_MOVE	4F	BIT_BLT_ERO	D6
REL_MOVE	52	BIT_BLT_ERT	D7

NON-DRAWING COMMANDS

NOP = No Operation	0300h			
LINK = Link to Next Command	0200h	Link Address Low	Link Address High	
INTR__GEN = Generate Interrupt	0E00h			
DUMP__REG = Dump Register	2900h	Dump Address Low	Dump Address High	Register ID
LOAD__REG = Load Register	3400h	Load Address Low	Load Address High	Register ID
CALL = Call Subroutine	0F00h	Call Addr Low	Call Addr High	
RETURN = Return from Subroutine	1700h			
HALT = Enter Poll State	xx01h			

DRAWING CONTROL COMMANDS

DEF__BIT__MAP = Define bitmap	1A00h	Origin Addr Low	Origin Addr High	Xmax	Ymax	Bits/pixel
DEF__CLIP__RECT = Define Clip Rectangle	4600h	xmin	ymin	xmax	ymax	
DEF__COLORS = Define Colors	3D00h	Foreground Color	Background Color			
DEF__TEXTURE = Define Texture Opaque/Transparent	0600/0700h	Pattern				
DEF__LOGICAL__OP = Define Logic Operation	4100h	Color Bit Mask	Function Code (see table below)			

The functions performed and their codes are:

FCODE	FUNCTION	FCODE	FUNCTION
0000	0	1000	CMP (source) AND CMP (dest)
0001	source AND dest	1001	CMP (source) XOR dest
0010	CMP (source) AND dest	1010	CMP (source)
0011	dest	1011	CMP (source) OR dest
0100	source AND CMP(dest)	1100	CMP (dest)
0101	source	1101	source OR CMP (dest)
0110	source XOR dest	1110	CMP (source) OR CMP (dest)
0111	source OR dest	1111	1

DEF__CHAR__SET = Define Character Set (Word/Byte mode)	0A00/0B00h	Font Addr Low	Font Addr High
DEF__CHAR__ORIENT = Define Char Orientation	4000h	Path /Rotation	

There are four defined values for both the path and rotation. They are:

CODE	INCREMENT
00	0 degrees
01	90 degrees
10	180 degrees
11	270 degrees

DEF_SPACE = Define Inter Char and Bit Blt GCX Update Space	4D00h	Space	
ABS_MOV = Move	4F00h	x coordinate	y coordinate
REL_MOV = Relative Move	5200h	dx	dy
ENTER_PICK = Enter Pick Mode	4400h		
EXIT_PICK = Exit Pick Mode	4500h		

GEOMETRIC COMMANDS

POINT = Draw Point	5300h	dx	dy	
INCR_POINT = Draw Incremental Points	B400h	Array Addr Low	Array Addr High	N (# of pts)

INCREMENTAL POINTS ARRAY

INC4	INC3	INC2	INC1
—	—	—	—
—	INCN	INCN-1	INCN-2

The largest allowable single array of incremental points is 32K points. The upper two bits of the "inc" field specify the increment for the x coordinate while the lower two bits specify the increment for the y coordinate, The encoding for the two bits is as follows:

CODE	INCREMENT
00	0
01	+1
10	−1
11	Unused

NOTE:
Transparent mode is not supported with the INCR_POINT command if the texture is non-solid.

LINE = Draw Line (With End Point/ without End Point)	5400/5500h	dx	dy	
CIRCLE = Draw Circle	8E00h	radius		
RECT = Draw Rectangle	5800h	dx	dy	
POLYLINE = Draw Polyline	7400h	Array Addr Low	Array Addr High	N (# of lines)
POLYGON = Draw Polygon	7300h	Array Addr Low	Array Addr High	N (# of lines)

POLYLINE/POLYGON ARRAY

dx1
dy1
—
dxN
dyN

HORIZONTAL LINE ARRAY

dxy
dyl
deltaX1
—
dxN
dyn
deltaXN

ARC = Draw Arc (Exclusion/Inclusion)	6800/6900h	dxmin	dymin	dxmax	dymax	radius
SCAN_LINES = Draw Series of Horizontal Lines	BA00/BA01h	Array Addr Low	Array Addr High	N (# of lines)		

NOTE:
Transparent mode is not supported with the circle and arc commands if the texture is non-solid.

BITBLT COMMANDS

BIT_BLT = Bit Block Transfer within bitmap	6400h	Source x coord	Source y coord	dx	dy
BIT_BLT_M = Bit Block Transfer across bitmaps	AE00h	Source Addr Low	Source Addr High	Source Xmax	
	Source Ymax	Source x coord	Source y coord	dx	dy
BIT_BLT_E = Bit Block Transfer across bitmaps (opaque, transparent, opaque/reverse, transparent/reverse)	D400/D500/ D600/D700h	Source Addr Low	Source Addr High	Source Xmax	
	Source Ymax	Source x coord	Source y coord	dx	dy

CHARBLT COMMANDS

CHAR = Draw Character String (opaque, transparent, opaque/reverse, transparent/reverse)	A600/A700/ A800/A900h	String Ptr Low	String Ptr High	N(# of char)

CHARACTER STRING FORMAT

Word Mode
char1
char2
—
charN

Byte	Mode
char2	char1
char4	char3
—	—
charN	charN-1

NOTE:
In byte mode, the character code of the first character to be drawn must reside at an even address.

DISPLAY PROCESSOR

Introduction

The Display Processor (Display Processor) is an independent processor responsible for controlling the display of video data on a CRT, laser printer and other display devices. Its functions include the generation of horizontal and vertical timing signals, blanking signal and the control of 8 Video Data output pins.

The 82786 can function in two distinct types of graphics memory environments – i) using single port DRAMs (normal display mode) and ii) using dual port video DRAMs (VRAM mode). When the 82786 is configured to interface with single port DRAMs, the Display Processor uses the BIU to fetch the screen parameters and display data from memory. The Display Processor then internally shifts the video data into the video stream for screen refresh. When configured to run with VRAMs, the Display Processor uses the BIU to load the shift registers in the VRAMs at the beginning of every scan line. The screen refresh is then done by the second port of the VRAMs. The BIU and Graphics Processor have the rest of the scan line time to access the graphics memory.

Bitmap Organization

The Display Processor is optimized to display data in packed bitmap form. The Graphics Processor writes pixel data in the memory in this form. The Display Processor supports display of 1, 2, 4 or 8 bits/pixel data, stored in sequential bitmap form, with the first (left-hand) pixel to be displayed occupying the Most Significant Bit(s) of a word in memory, and subsequent pixels occupying sequentially lower bits in the word. Ascending word addresses represent subsequent pixels, moving left to right and top to bottom on the screen.

Windows and Normal Display Mode

In the normal display mode, Windows may be displayed on the screen in a flexible format. There can be up to 16 window segments or tiles appearing on any single display line. There is no limit on the number of windows vertically (limited by the number of scan lines in the active display area). At the basic video rate (25 MHz, 8 bpp), these windows may be placed at pixel resolution on the screen, and mapped at pixel resolution into the bitmap. Windows can be made to overlap, by breaking the windows into tiles and assembling the tiles on the screen.

Cursor (Normal Display Mode)

The Display Processor supports a single hardware cursor which may be 8 x 8 pixels or 16 x 16 pixels. This cursor may be positioned anywhere on the screen with a pixel resolution. The cursor may be defined to be transparent or opaque, and may be either a block cursor with its hot-spot at the top-left of the cursor pattern, or a cross-hair cursor one pixel across, stretching the width and height of the screen with its hot-spot at the center of the cross. The cursor color and pattern (shape) are programmable. The cursor may be programmed off if not required, or to implement a blinking cursor.

Video Rates (Normal Display Mode)

The Display Processor is clocked from an external Video Clock. In this mode, the 82786 fetches video data from memory into an internal FIFO. An internal shift register then generates the serial video data stream to the display. The 82786 will support CRT screens of up to about 640 x 480 pixels at 8 bits/pixel and 60 Hz non-interlaced, or about 1024 x 640 x 8 at 60 Hz interlaced. The Display Processor supports Interlaced, non-interlaced and interlace-sync displays.

The Display Processor also has higher speed modes which enable the user to trade off bits/pixel for dot-rate. Thus it is possible to run at a maximum of 8 bpp with a 25 MHz dot-rate, 4 bpp at a 50 MHz dot-rate, 2 bpp at a 100 MHz dot-rate or 1 bpp at 200 MHz dot-rate; with corresponding increase in size and resolution. Note that in the high speed modes, horizontal window and cursor placement resolution is reduced to 2, 4 or 8 pixel resolution at 50 MHz, 100 MHz, or 200 MHz rates respectively.

VRAM Mode

In the VRAM mode, the first tile for every scan line is used to load the shift register in the VRAMs by executing a data transfer cycle. Subsequent tiles (if any) for all strips will still be available through the VDATA pins of the 82786. The window status bits can be used to internally multiplex the VRAM video stream and the 82786 generated video stream. The address for this data transfer cycle is determined from the Tile Descriptor. The 82786 BEN# pin is used as a DT pin for this case. If the graphics memory banks are interleaved, then both banks are loaded in the transfer cycle. During the Blank period, Default VData appears on the VDATA pins.

CRT Controller

CRT timing signals HSYNC, VSYNC, and BLANK are each programmable at a pixel resolution, giving a maximum display size of 4096 x 4096 pixels. If High Speed, Very High Speed, or Super High Speed display modes are selected, the horizontal resolution of the CRT timing signals becomes 2 pixels, 4 pixels or 8 pixels at 50 MHz, 100 MHz and 200 MHz respectively.

Window Status

The HSync and VSync CRT timing pins may be configured to serve as Window Status output pins, which can be programmed to present a predefined code while the Display Processor is displaying a tile. This code is programmable as part of the Tile Descriptor, and may be used externally to multiplex in video data from another source, or select a pallette range for a particular window, etc. External logic must be used to enable VSync and HSync as CRT timing signals when Blank is high, and as encoded Window Status signals when Blank is low. This is valid in both DRAM and VRAM modes.

Zoom Support

The Display Processor allows windows to be zoomed in the normal display mode. The zoom factor is an integer between 1 and 64. There are independent zoom factors for the x and y direction. The zoom function results in pixel replication.

All zoomed windows on a display are zoomed by the same amount. A window is therefore either zoomed or not zoomed. Zoom offset is not supported—a pixel must either be fully displayed or not displayed at all. This places a restriction on window placement—a window may not be placed such that a zoomed pixel is partially obscured. VRAM displays can be zoomed vertically by using this feature. Horizontal zooming of VRAM windows requires external hardware support.

Only even zoom factors are supported in the Y direction with interlaced displays. In addition, when zooming, both descriptor lists (interlaced systems use two descriptor lists, one for each frame) must point to the same place in memory, i.e., they must be identical list.

Extended 82786 Systems

The CRT timing signal pins may be configured as output pins (for the normal stand-alone 82786 system), or as input pins for a system in which multiple 82786's are ganged in parallel to provide a greater number of bits/pixel, higher dot rates, larger display area, or more windows. In multiple 82786 systems, each of the Display Processors run in lock step, allowing the individual outputs to be combined on a single display. The HSync, VSync and Blank pins for the "Slave" 82786 are configured as inputs and are driven by the "Master" 82786.

When programmed as inputs, VSync and HSync still serve as outputs for Window Status while Blank is inactive.

External Video Source

The HSync and VSync pins on the 82786 can be configured as inputs to synchronize the 82786 to external video sources (VCR, broadcast TV etc.). In this case, the Blank pin is configured as output and the active 82786 display period is determined by the programmed 82786 parameters.

Memory Bandwidth Requirements

The memory bandwidth required by the Display Processor depends on the display size and mode of operation. The 82786 has a 40 Mbyte/sec maximum bandwidth during fast block accesses to graphics memory. In the normal display mode the Display Processor makes use of these fast block reads for screen refresh, thereby minimizing its use of the memory bus, which the other 82786 modules share. For worst-case displays, when the Display Processor is running at its maximum speed of 25 MHz and 8 bits/pixel, about 50% of the memory bandwidth is used for display refresh. Correspondingly, at only 1 bit/pixel the Display Processor's bus requirements are reduced to about one-eighth of its requirement at 8 bpp. In the VRAM mode, the Display Processor does not fetch any of the display data. The display data is passed directly from the graphics memory to the pixel logic. In this case about 1% of the graphics memory bandwidth is required by the Display Processor to fetch the Strip Descriptors.

Display Processor Registers

There are two different register sets for the Display Processor. Six of the 82786 Internal Registers are dedicated to the Display Processor. These registers are memory (or I/O) mapped in the external CPU address space. They can therefore be directly accessed by the external CPU. Another set of registers is totally local to the Display Processor. These are the display control registers and are used for display parameters.

82786 Registers For Display Processor

There are six of these Registers. They are listed below:

Address	Function
Base + 40	Display Processor Opcode
Base + 42	Param1
Base + 44	Param2
Base + 46	Param3
Base + 48	Display Processor Status
Base + 4A	Default Video

The Display Processor Opcode and the three parameter registers are used to send a command to the Display Processor. The Display Processor Status Register contains the status for the Display Processor. This is described in more detail later. The Default Video Register contains the data that appears on the Video Out pins during the blanking intervals. The CPU can use this register to address an external pallette RAM while loading the pallette, thereby eliminating a separate address path and external logic.

Display Control Registers

The display control registers can be loaded under control of the Display Processor during the Vertical Blanking interval. This synchronizes parameter updates with display refresh and ensures that the display remains clean, with no updates occurring during data display.

The Display Processor may also be programmed to provide a Frame Interrupt once per certain number of frames. This may be used to facilitate blinking, scrolling, panning, animation or other periodic functions.

Command Execution

At the beginning of each Vertical Blanking time, the Display Processor checks the ECL bit in the Display Processor Opcode Register. If the ECL bit is 1, the Display Processor status remains unchanged. If the ECL bit is 0, the Display Processor executes the command. Only one command is executed per frame.

On completion of the command, the Display Processor sets its ECL bit back to 1, indicating to the CPU that a new command may be written into the Command Register. This handshake prevents the CPU from writing a new command before the old one has finished executing. The commands for the Display Processor are:

1. Load Register
2. Load All Registers
3. Dump Register
4. Dump All Registers

The command formats are:

LOAD REGISTER (LD—REG):

0	0	0	0	0	1	0	0	0	0	0	0	0	WP	LP	ECL
Memory Address Lower															
Mem Addr Upper															
Register ID															

This command loads a pair of display control registers with values stored in memory starting at the location given by Memory Address. The Memory Address must be an even byte address. The Register ID for the register pair is given in the register block description below. This command may be used to update individual pairs of registers (such as the Cursor Position registers). The register ID must be an even number for this command.

LOAD ALL (LD—ALL):

0	0	0	0	0	1	0	1	0	0	0	0	0	WP	LP	ECL
Mem Address Lower															
Mem Addr Upper															

This command loads the entire block of display control registers in a block read starting from the Memory Address given in the command. The Memory Address must be an even byte address. This command must be the first command executed and has to be executed after reset to enable the display operation. The registers are listed below.

DUMP (DMP—REG):

0	0	0	0	0	1	1	0	0	0	0	0	0	WP	LP	ECL
Memory Address Lower															
Mem Addr Upper															
Register ID															

This command causes the Display Processor to write the contents of the display control register pair specified by Register ID to the location in memory specified by Memory Address. The Memory Address must be an even byte address. The register ID must be an even number for this command.

DUMP ALL (DMP—ALL):

0	0	0	0	0	1	1	1	0	0	0	0	0	WP	LP	ECL
Mem Address Lower															
Mem Addr Upper															

This command causes the Display Processor to write its entire display control register block out to a block in memory, starting at the Memory Address specified. The Memory Address must be an even byte address. The write occurs as a series of single write cycles.

For any of the Display Processor's four commands, setting the LP bit to 1 will cause that command to execute at the start of each VSYNC period. While in Loop Mode, the DP does not set the ECL bit back to 1 at the end of each execution. Exit Loop Mode by writing 0 to the LP bit.

The Write Protect bit (WP, bit 2 of the DP opcode register) allows the user to write protect the CRT Timing parameter registers (Display Control Register Block registers 06h–0Dh). Write protect is not enabled until after the first DP command has executed. This must be a LOAD ALL. Before changing the WP bit, the user should exit Loop Mode and wait for the ECL bit to return to 1.

Display Control Register Block

The display control register block is shown below. Each register is 16-bits wide. The numbers in parentheses are the number of bits per parameter.

		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
00h	VStat:															C	D	C- CsrOn(1); D- DspOn(1)
01h		—								IntMsk								
02h		—								TripPt								
03h		—								Frint−1								
04h		Reserved																
05h	CRTMode:	—									IL		W	S	B	AA		

IL - Interlace(2): 00 → Non-Interlace
01 → Reserved
10 → Interlace
11 → Interlace-Sync

W - Window Status Enable(1)
S - HSYNC, VSYNC Slave Mode(1)
B - Blank Slave Enable(1)
AA - Accelerated Video (High Speed Video, etc.)(2)
00 → Normal (25 MHz)
01 → High Speed (50 MHz)
10 → Very High Speed (100 MHz)
11 → Super High Speed (200 MHz)

		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
06h		—				HSynStp−3											
07h		—				HFldStrt−3											
08h		—				HFldStp−3											
09h		—				LineLen−3											
0Ah		—				VSynStp−1											
0Bh		—				VFldStrt−1											
0Ch		—				VFldStp−1											
0Dh		—				FramLen−1											
0Eh		Descriptor Addr. Pointer (L)															
0Fh		Descriptor Addr. Pointer (U)															
10h		Reserved															
11h		XZoom−1								YZoom−1							
12h		—								FldColor							
13h		—								BdrColor							
14h		—								1Bpp Pad							
15h		—								2Bpp Pad							
16h		—								4Bpp Pad							
17h	CsrMode:	S	X	T	CSt		CSC		—	CsrPad							

CsrStyle: S - CsrSize(1): 0 → 8 x 8 Csr
1 → 16 x 16 Csr

X - CsrX-Hair(1)
T - CsrTransparent(1)
CSt - CursorStatus (to Window Status output)(2)
CSC - CursorStatusControl(2): 00 → Current Window Status
01 → Foreground
10 → Background
11 → Block

	15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0
18	—	CsrPos X−2
19	—	CsrPos Y−1
1A	CsrPat0	
1B	CsrPat1	
1C	CsrPat2	
1D	CsrPat3	
1E	CsrPat4	
1F	CsrPat5	
20	CsrPat6	
21	CsrPat7	
22	CsrPat8	
23	CsrPat9	
24	CsrPatA	
25	CsrPatB	
26	CsrPatC	
27	CsrPatD	
28	CsrPatE	
29	CsrPatF	

The functions of the preceding registers are described in more detail below:

0. VStat:CsrOn(1) DspOn(1)

   If set, the internally generated display or cursor are turned on.

1. IntMsk

   Interrupt Mask Register. This register enables an 82786 interrupt whenever the corresponding bit in the Display Processor Status Register is set. A 0 for any bit enables the interrupt. This Interrupt Mask is different from the Interrupt Mask for the Graphics Processor. If using interrupts, mask bit 5 of this register.

2. TripPt

   The Trip Point register is a reserved field and must be programmed to 00h.

3. Frint

   Frame Interrupt Register. Enter the number of frames minus one elapsed between successive setting of the FRINT bit in the Display Processor Status Register.

4. Reserved field should always be set to zero.

5. CRTMode — IL(2) W(1) S(1) B(1) AA(2)

   These bits control the various modes of the CRT Controller.

   IL are the Interlace Control bits—if IL is 00, the display is Non-Interlaced. If IL is 10, the display is Interlaced (displaying the even lines (Field 1) of the frame and then the odd lines (Field 2)). If IL is 11, the display is interlace-sync (similar to interlace, except that the odd field display is identical to the even field display).

   W is the Window Status Enable bit. If W is 0, HSYNC and VSYNC will have normal operation. If W is 1, the Window Status Code programmed into the Tile Descriptors will be output on VSYNC and HSYNC pins while display data for that particular window is being displayed. VSYNC represents the MSB and HSYNC the LSB of the Window Status Code.

   S is the HSYNC/VSYNC Slave Mode bit. If S is 0, the VSYNC and HSYNC pins are outputs. If S is 1, they are inputs. In the Slave Mode, if Window Status is enabled, HSYNC and VSYNC will still be outputs while BLANK is low.

   B is the Blank Slave Mode bit. If B is 0, the BLANK pin is an output. If B is 1, it is an input.

NOTE:
Always program the slave 82786 first, then the master. The slave VSYNC, HSYNC, and BLANK pins must be held high until they are programmed.

AA are the Accelerated Video Mode bits. By using an external latch or shift register, 50, 100 or 200 MHz video data rates can be generated. In the Accelerated Video Modes, each memory byte represents 2, 4 or 8 physical pixels. The upper bit(s) of each byte represent the pixels that appear on the left on the display medium. Used in DRAM display mode. Must be programmed to zero for the first tile in the VRAM Mode.

6. HSynStp

   Enter the HSYNC width in number of VClks minus 3. (For a graphical representation of all the CRT timing signals, see Figure 3).

7. HFldStrt

   Enter the number of VClks minus 3 between the rising edge of HSYNC and the falling edge of BLANK (the start of Video Data).

8. HFldStp

   Enter the number of VClks minus 3 between the rising edge of HSYNC and the rising edge of the next BLANK (the end of Video Data).

9. LineLen

   Enter the number of VClks minus 3 between the rising edge of HSYNC and the rising edge of the next HSYNC.

10. VSynStp

    The number of Horizontal Synchronizations (HSYNCs) between the beginning of Vertical Synchronization (VSYNC) and the end of VSYNC.

    Enter VSYNC width as the number of HSYNC periods minus one. In the non-interlaced mode, VSYNC rises and falls on the rising edge of HSYNC. In interlaced and interlace-sync mode, VSYNC has the same timing as in non-interlace mode at the start of each Even Field (lines 0, 2, 4, etc), but is delayed by half LineLen at the start of each Odd Field (lines 1, 3, 5, etc). (See Figure 3.)

11. VFldStrt

    Enter the number of HSYNCs minus one between the beginning of VSYNC and the end of Vertical Blanking.

12. VFldStp

    Enter the number of HSYNCs minus one between the beginning of VSYNC and the beginning of the next Vertical Blanking.

13. FramLen

    Enter the number of HSYNCs minus one between the beginning of VSYNC and the beginning of the next VSYNC.

14. Descriptor Address Pointer (L)

    The address of the first Strip Descriptor for the display. After fetching the first descriptor the Display Processor uses the Link Address in the descriptor to fetch the next descriptor. The Descriptor address must be an even byte address.

15. Descriptor Address Pointer (U)

    The most significant bits of the Descriptor Address Pointer.

16. Reserved field should always be set to zero.

17. ZoomX, ZoomY

    Enter the x-zoom factor minus one and y-zoom factor minus one for the zoomed windows. The zoom factor can be any integer number between 1 and 64. In the VRAM mode, ZoomX is not used unless additional logic is added.

18. Field Color

    An 8-bit value indicating the color of the background field to be displayed in the absence of windows.

19. Border Color

    An 8-bit value indicating the color of the border to be displayed inside selected windows.

20. 1Bpp Pad

    An 8-bit value where the upper 7 bits represent the upper 7 bits of video data concatenated to the 1 bit video data from a 1 bit/pixel bitmap.

21. 2Bpp Pad

    An 8-bit value where the upper 6 bits represent the upper 6 bits of video data concatenated to the 2 bit video data from a 2 bit/pixel bitmap.

22. 4Bpp Pad

    An 8-bit value where the upper 4 bits represent the upper 4 bits of video data concatenated to the 4 bit video data from a 4 bit/pixel bitmap.

23. CsrStyle:S(1) X(1) T(1) CSt(2) CSC(2) CsrPad

    The Cursor Mode Register. The Cursor Pad is an 8-bit value where the upper 7 bits are the higher 7 bits for the cursor color.

    Cursor Style: S is the size bit. If S is 0 an 8 x 8 pixel cursor will be displayed. If S is 1, a 16 x 16 pixel cursor will be displayed.

    X is the CrossHair Mode bit. If X is 0, a block cursor will be displayed. The pattern for the cursor is specified in the Cursor Pattern registers. The cursor hot-spot is at the top-left of the cursor block. If X is 1, a crosshair cursor will be displayed. Its hot-spot is at the center of the cross, and it will stretch the full height and width of the display.

    T is the Transparent Mode bit. If T is 0, the cursor is opaque. Its forground color is determined by the concatenation of the cursor padding bits (7 MSB's) with 1. The background color is determined by the concatenation of the cursor padding bits with 0. If T is 1, the cursor background reverts to whatever bitmap data is being displayed "behind" the cursor.

    CSt is the Cursor Status. The code to be output onto the Window Status outputs while the Cursor is being displayed.

CSC is the Cursor Status Control (2 bits). The cursor status may be output whenever the cursor foreground color is being output, whenever the cursor background color is being output, or whenever the cursor block is active, whether it is displaying background color or foreground color or transparent pixels (useful for inverse video), or else the cursor status may default to the current Window Status. The code is shown in the Display Control Register Block.

CsrPad: Cursor padding bits.

24. CsrPos X

This is the Cursor X-Position Register—the position of the cursor hot-spot relative to the beginning of the line (the rising edge of the previous HSYNC). Enter the value minus 2.

25. CsrPos Y

This is the Cursor Y-Position Register—the position of the cursor hot-spot relative to the beginning of the frame (the beginning of the previous VSYNC). Enter the value minus one.

26. CsrPat0:F

These 16 registers contain the pattern to be displayed as a cursor. CsrPat0 is the top row of the cursor, and the MSB is the left bit of the cursor. For an 8 x 8 cursor, the cursor pattern used is the higher byte of the first eight cursor registers.

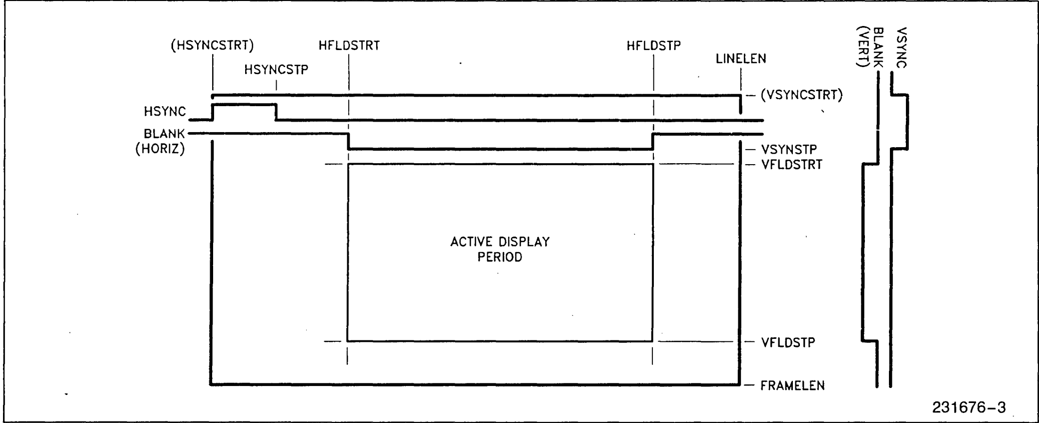


Figure 3. Timing Parameters

NOTE: In slave video mode, at least a 1-line vertical front porch and a 7-line vertical back porch are required.

Windows

The CPU creates Strip Descriptors in memory that describe windows for the Display Processor. The Strip Descriptors are organized as one Descriptor per strip of window segments (tiles) as shown in Figure 4. Each Descriptor contains information for the tiles within that strip in the order they are displayed on the screen (left to right). The Descriptor for a particular strip must be contiguous in memory. The Strip Descriptors for several strips are linked to each other in the order they are displayed (top to bottom). The linking is done through the Link to Next Strip Descriptor parameters in each Descriptor, which points to the following Descriptor. The Descriptor for the first strip is accessed during the VBlank interval, using an address specified by the Descriptor Address Pointer, one of the Display Control Register pairs.

The Strip Descriptor consists of a header followed by one or more Tile Descriptors. The header and Tile Descriptors must occupy one contiguous block in memory.

The format of the Window Strip Descriptors is:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Header	Number of Lines in Strip − 1															
	Link to Next Strip Descr. (L)															
	Link to Next Strip Descr. (U)															
	C	Number of Tiles in Strip − 1														
1st Tile Descr.	Bitmap Width															
	Mem Start Address (L)															
	Mem Start Address (U)															
					Bpp				StartBit				StopBit			
	Fetch Count (bytes − 2)															
	T	B	L	R	WSt		—						PC		Z	F
2nd Tile Descr.	Bitmap Width															
	Mem Start Address (L)															
	Mem Start Address (U)															
					Bpp				StartBit				StopBit			
	Fetch Count (bytes − 2)															
	T	B	L	R	WSt		—						PC		Z	F

etc...

NOTE:
The first tile of any scan line must be greater than 1 pixel.

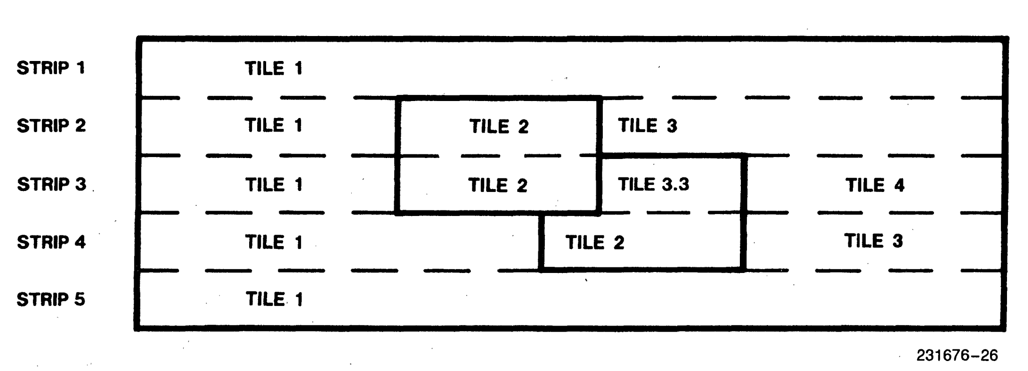


Figure 4. Display Shows Strips and Tiles with Two Overlapping Windows

The Strip Descriptor Header is programmed with values for the number of display lines minus one and the number of tiles in the strip minus one. There may be any number of lines in a strip, up to the number of lines on the display (within their restrictions imposed by zoom, if used). The first VRAM strip must be at least 2 lines. In DRAM mode there may be up to 16 tiles within a single strip. In the VRAM Mode the first tile is used to load the VRAM shift register, leaving up to 15 tiles to be used by the Display Processor. The header also contains Link to Next Strip Descriptor parameters.

NOTE:
You must only define in the strip descriptors the number of scan lines that will actually be displayed.

The C bit (the most significant bit) in the Number of Tiles in Strip parameter tells the DP to color the display area following the current strip with FldColor data or link to the next strip. If the C bit is set to one, the DP colors the remainder of the display with the background color defined in the FldColor Register of the Display Control Register Block. If the C bit is zero, the DP links to the next strip.

Each Tile Descriptor contains the following parameters:

1. Bitmap Width—the width of the bitmap in bytes. This must be an even byte address. Bitmap Width is added to the Memory Address for each scan line in the window (each HSync period within the strip) to get the start address of the next display line (if y-zoom inactive or counted out). In case of interlaced displays, the Memory Address is incremented by twice the bitmap width. In the VRAM Mode, the bitmap width of the first tile must be a power of 2 and must be less than the maximum width of the VRAM shift register.
2. Memory Start Address—the memory address for the window. This is an even byte address, corresponding to the address of the first word of bitmap data for the window tile (top left corner). In the VRAM mode the start address for the first tile must guarantee that the entire scan line is contained in a single row of the VRAM.
3. Bpp—The number of bits/pixel in the current window—must be programmed to 1, 2, 4, or 8 in the normal mode. In the VRAM mode this field should be zero.
4. StartBit—The bit position in the corresponding memory word for the first bit of the first pixel in the window. Gives bit resolution to the Memory Start Address (and pixel resolution to the start of the window). In the normal mode this must be programmed to be consistent with the Bpp defined for that window. In the VRAM mode, this must be programmed to zero for the first tile.
5. StopBit—The bit position in the corresponding memory word for the last bit of the last pixel in the widow. Gives bit resolution to the window width. In the normal mode this must be programmed to be consistent with the Bpp defined for that window. An illegal value will result in incorrect display. In VRAM mode, this must be programmed to zero for the first tile.
6. Fetch Count—In the DRAM mode, this specifies the number of bytes minus two from the bitmap to be fetched for each scan line in the current window tile. This must be an even quantity. The value programmed in this field is 2 less than the number of bytes to be fetched rounded off to the next higher even number. In the VRAM mode, this must be programmed to zero for the first tile.
7. TBLR—Border Control Bits—When a bit is set to one, it turns on the border on Top, Bottom, Left or Right of window tile. This is a four bit field with one bit controlling each border. The most significant bit controls the top border and the least significant bit controls the right border. All four bits must be programmed to zero for the first tile in VRAM Mode.
8. WST—Window Status (2 bits)—The code to be presented on the Window Status pins while the window is being displayed.
9. PC—IBM PC Mode—Indicates that this window is being displayed from a bitmap created in IBM PC format. The Display Processor supports the IBM Color Graphics Adapter bitmap format in which the least significant byte of a word appears on the left of the most significant byte on the screen as opposed to the 82786 format in which the least significant byte appears to the right of the most significant byte. Also, the 2-bank and 4-bank bank oriented bitmaps used in the PC and PCjr systems are supported. These modes enable bitmaps created by IBM PC or PCjr (or compatible) systems to be upward compatible with 82786 displays, with the PC format bitmaps being displayed either as the whole screen, or as windows on a screen together with 82786 created bitmaps. The PC mode bitmaps can be zoomed or used with interlaced or accelerated displays. In the VRAM mode, this field must be programmed to zero for the first tile.

   Note that although the Display Processor can display bitmaps created in these formats, the Graphics Processor always draws bitmaps in 82786 format. The vertical mapping of IBM format bitmaps is restricted in that the Memory Start Address of an IBM format window must be in the first of the 2 or 4 banks.

The coding for IBM PC mode is given below:

00 → 82786 Mode
01 → Swapped Byte Mode
10 → Swapped Byte, 2 banks*
11 → Swapped Byte, 4 banks*

*Not supported in Interlaced mode.

Bitmap formats in 82786 and PC Modes are shown below:

Pixel # (from left as displayed on screen):

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
82786 Mode Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC Mode Bit #	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8

10. Z—Zoom—This bit if set, indicates that in the normal display mode the window is to be zoomed using, the zoom parameters programmed into the ZoomX and ZoomY registers.
11. F—Field—This bit if set, indicates that the window tile is background field. In the normal mode the field color is displayed for the window. The number of pixels of Field to be displayed should be programmed into what would normally be the Bpp, StartBit, StopBit fields. This bit must be set to zero for a the first tile in the VRAM mode.

If the Strip Descriptor list causes a window to be displayed that extends beyond the active display area, then only the upper left hand portion of the window is displayed and the rest of it is truncated.

In interlace mode, in order to maintain a line resolution on vertical positioning of windows, a double-length Descriptor Table must be used. The first part contains window position information for the even lines, the second part for the odd lines. Also note that in interlace mode, one frame takes two fields to display. Command execution occurs at frame boundaries, not field boundaries, so the instruction execution frequency will typically be 25/30 Hz instead of the non-interlaced 50/60 Hz.

Initialization

The Display Processor is reset during the main 82786 reset process. Upon reset it enters a well defined reset state described below:

1. Any command execution is immediately halted.
2. Parameter, Descriptor, or Display Data fetches are terminated.
3. Display Outputs VDATA7:0 are all reset to default video.
4. HSync, VSync, Blank are tristated (Display Processor defaults to Slave Operation). These stay tristated until the first LOAD_ALL instruction.
5. Display Processor Status Register is cleared.
6. Display Processor Interrupt Mask to set to all 1's (all interrupts disabled).
7. ECL bit is set to 1.

Display Processor Interrupts, Status Register and Exception Handling

The Display Processor Status Register is an 8-bit memory (or I/O) mapped register which indicates the current status of the Display Processor, and allows the generation of interrupts depending on the state of individual bits. Interrupts may be masked off using the Display Processor Interrupt Mask Register. The format of the Display Processor Status Register is:

ADDRESS	7	6	5	4	3	2	1	0
BASE + 48 h	FRI	RCD		FMT	BLK	EVN	ODD	ECL

Display Processor Status Register

The functions of each bit, and the action taken in the case of exceptions is described below:

FRI—Frame Interrupt. This bit is set every n frames, where n is a value between 1 and 256 loaded into the Frint Register. This may be used, for example, for timing in animation applications, or to time blink rates.

RCD—Reserved Command. This bit is set if the Display Processor does not recognize the Opcode it has been instructed to execute. The Display Processor will not execute the command.

Reserved

FMT—FIFO Empty. This indicates that the Display FIFO has underrun. This forces an End of Line condition and the rest of the Display Line will display the FldColor defined in the Display Control Register Block. At the beginning of HBlank, the Display Processor uses the current Descriptor to start a new Display Data fetch. A FIFO underrun therefore does not necessarily mean that the whole field is lost—just the current display line is corrupted.

BLK—Blank. This indicates that the BLANK pin is currently active for Vertical Sync.

EVN—Even Field. In Interlace and Interlace-Sync modes, this bit is set during the even field (Field 1).

ODD—Odd Field. In Interlace and Interlace-Sync modes, this bit is set during the odd field (Field 2). The Even and Odd status bits assist in synchronizing the 82786 with other interlaced display systems.

ECL—End of Command List. This is set at the same time the ECL bit in the Opcode Register is set, and allows the Display Processor to inform the CPU as soon as it has completed execution of a command. In Loop Mode, the ECL bit is not set. It will be set upon exiting Loop Mode.

All active interrupts are OR'ed together to drive a single 82786 interrupt line. Once set, the interrupt line remains active until the Status Register is read. The active bits in the Status Register (bits with 0 in the corresponding bit in the Interrupt Mask) are reset to zeroes after the Status Register is read.

Test Modes

The 82786 implements several special modes of operation beyond normal use to aid in debug, characterization and production testing. When RESET goes inactive, the RD and WR pins are sampled. If either of these two pins is low, one of the special test modes is enabled according to the state of RD, WR and MIO pins.

A 16-bit Linear Feedback Shift Register signature analyzer is placed on the Video output bus to compress the video data stream into a single signature that is output onto the Video Data pins during Blank time. The signature is also readable by the CPU at the end of a Frame using the Dump__Reg command at Register ID 3D. This signature analyzer output onto the VDATA lines is activated in DP Test Mode. Once in DP Test Mode, the signature Analyzer is enabled by setting bit 14 of the DP Opcode register to 1.

The 82786 implements three global pin conditioning features. Specifically, the 82786 can drive all output and I/O pins high, or low, or can tristate all pins. The test modes are activated according to the following table:

RD#	WR#	MIO	Mode
0	0	0	Reserved
0	0	1	Reserved
0	1	0	DP Test Mode
0	1	1	Drive Output Pins High
1	0	0	Drive Output Pins Low
1	0	1	Tristate Pins
1	1	X	Normal Operation

NOTE:
All timing numbers in the parametric section are preliminary and are subject to change.

V_{OL}/V_{OH} Pin Conditioning

The 82786 has the capability to bring all its output pins to a constant logic high or low state. This feature can be used for testing the output buffers on the 82786.

Tristate Feature

The 82786 has the ability to tristate all of its I/O and output pins to effectively isolate the 82786 from any connected circuitry. This allows testing a completely assembled PC board by isolating the 82786. Leakage on all I/O pins can also be tested in this mode.

82786 PARAMETRICS

ABSOLUTE MAXIMUM RATINGS

Storage Temperature	−65°C to +150°C
Operating Temperature	0°C to 70°C
Voltage V_{CC}−V_{SS}	−0.5V to +6.5V
Voltage on Other Pins	−0.5V to V_{CC} + 0.5V

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

Symbol	Parameter	Min	Max	Units	Notes
V_{ILC}	Input Low Voltage	−0.5	+0.8	V	CLK Input
V_{IHC}	Input High Voltage	+2.0	V_{CC} + 0.5	V	CLK Input
V_{ILVC}	Input Low Voltage	−0.5	+0.8	V	V_{CLK} Input
V_{IHVC}	Input High Voltage	+3.9	V_{CC} + 0.5	V	V_{CLK} Input
V_{IL}	Input Low Voltage	−0.5	+0.8	V	All Other Pins
V_{IH}	Input High Voltage	+2.0	V_{CC} + 0.5	V	All Other Pins
V_{OL}	Output Low Voltage	—	+0.45	V	All Pins I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	+2.8	—	V	All Pins I_{OH} = −400 μA

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

Symbol	Parameter	Min	Max	Units	Notes
I_{LI}	Input Leakage Current	—	±1	µA	$0 < V_{IN} < V_{CC}$
I_{LO}	Output Leakage Current	—	±10	µA	$0.45 < V_{IN} < V_{CC}$
I_{CC}	Power Supply Current	—	200	mA	@ 0°C Temp CLK @ 20 MHz V_{CLK} @ 25 MHz

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

CLOCK and RESET Timings

AC timings are referenced to 1.5V on clock input and 0.8V/2.0V on other pins

Symbol	Parameter	Min	Max	Units	Notes
T_C	CLK Period	50	200	ns	@ 1.5V
T_{CL}	CLK Low Time	20	—	ns	@ 1.5V
T_{CH}	CLK High Time	20	—	ns	@ 1.5V
T_{CR}	CLK Rise Time	—	10	ns	@ 0.8V–2.0V
T_{CF}	CLK Fall Time	—	10	ns	@ 0.8V–2.0V
T_{R1}	Test Input Setup Time	10	—	ns	
T_{R2}	Test Input Hold Time	5	—	ns	
T_{R3}	Reset Active Hold Time	25	$2 T_C$	ns	
T_{R5}	Reset Inactive Hold Time	10	—	ns	
T_{R6}	Reset Active Setup Time	10	—	ns	
T_{R7}	Forced Output Delay	30	—	ns	
T_{R8}	Reset Width	$10 T_C$	—	ns	

DRAM Interface Timings

AC timings are referenced to 0.8V/2.4V on all pins and are valid for total DRAM capacitance on each pin between 30 pF and 200 pF

SINGLE READ, WRITE, READ MODIFY WRITE AND PAGE MODE CYCLES

Symbol	Parameter	Min	Max	Units
T_{RC}	Single Cycle Time	$6 T_C - 5$	—	ns
T_{RAC}(1)	Access Time from RAS#	—	$4 T_C - 30 - 0.050 C_R$	ns
T_{CAC}(1)	Access Time from CAS#	—	$2 T_C + T_{CH} - 20 - 0.050 C_C$	ns
T_{CAA}(1)	Acc Time from Col Addr	—	$3 T_C - 20 - 0.075 C_A$	ns
T_{OAC}(1)	Access Time from BEN#		$2 T_C - 25 - 0.050 C_B$	ns
T_{RP}	RAS# Precharge Time	$2 T_C - 10$	—	ns
T_{RAS}	RAS# Width	$4 T_C - 30 - 0.025 C_R$	—	ns
T_{RCD}	RAS# to CAS# Delay	$T_C + T_{CL} - 25 + 0.050 C_C - 0.050 C_R$	—	ns
T_{RSH}	RAS# Hold Time	$2 T_C + T_{CH} - 15 + 0.025 C_R - 0.050 C_C$	—	ns
T_{CSH}	CAS# Hold Time	$4 T_C - 15 + 0.025 C_C - 0.050 C_R$	—	ns

SINGLE READ, WRITE, READ MODIFY WRITE AND PAGE MODE CYCLES (Continued)

Symbol	Parameter	Min	Max	Units
T_{CAS}	CAS# Width	$2\,T_C + T_{CH} - 10 - 0.025\,C_C$	—	ns
T_{ASR}	Row Address Setup Time	$T_C - 10 + 0.075\,C_R - 0.075\,C_A$	—	ns
T_{RAH}	Row Address Hold Time	$T_C - 25 + 0.075\,C_A - 0.050\,C_R$	—	ns
T_{ASC}	Column Addr Setup Time	$T_{CL} - 17 + 0.075\,C_C - 0.075\,C_A$	—	ns
T_{CAR}	Col Addr Setup to RAS#	$3\,T_C - 10 + 0.025\,C_R - 0.075\,C_A$	—	ns
T_{OFF}	Data in Hold Time	10	—	ns
T_{BOV}	BEN0# to BEN1# Overlap	0	—	ns

SINGLE WRITE CYCLE

Symbol	Parameter	Min	Max	Units
T_{RWL}	WE# to RAS# Lead Time	$T_C - 14 + 0.025\,C_R - 0.050\,C_W$	—	ns
T_{WCH}	WE# Hold Time	$3\,T_C - 15 + T_{CH} + 0.025\,C_W - 0.050\,C_C$	—	ns
T_{WP}	WE# Width	$2\,T_C - 20 - 0.025\,C_W$	—	ns
T_{CWL}	WE# to CAS# Lead Time	$T_C - 13 + 0.025\,C_C - 0.050\,C_W$	—	ns
$T_{DS(W)}$	Data Out Setup Time	$T_C - 15 + 0.075\,C_W - 0.075\,C_D$	—	ns
T_{DH}	Data Out Hold Time	$T_C - 20 + 0.075\,C_D - 0.050\,C_W$	—	ns

READ MODIFY WRITE CYCLE

Symbol	Parameter	Min	Max	Units
T_{RWC}	RMW Cycle Time	$8\,T_C - 5$	—	ns
$T_{DS(RW)}$	Data Out (RMW) Setup Time	$T_{CH} - 20 + 0.075\,C_W - 0.075\,C_D$	—	ns
T_{DH}	Data Out (RMW) Hold Time	$T_C - 10 + 0.075\,C_D - 0.050\,C_W$	—	ns
$T_{OFF(RW)}$	Data In Hold/Data Out (RMW) Drive Time	10	$T_{CL} + 5 + 0.075\,C_D - 0.075\,C_B$	ns

PAGE MODE READ AND WRITE CYCLES

Symbol	Parameter	Min	Max	Units
T_{PC}	Page Mode Cycle Time	$4\,T_C - 5$	—	ns
T_{CP}	CAS# Precharge Time	$T_C + T_{CL} - 15$	—	ns
T_{CAS}	CAS# Width	$2\,T_C + T_{CH} - 10 - 0.025\,C_C$	—	ns
$T_{CAH(n)}$	Col Addr Hold (Non Interleaved)	$3\,T_C + T_{CH} - 20 + 0.075\,C_A - 0.050\,C_C$	—	ns
$T_{DS(n)}$	Data Out Setup (Non Interleaved)	$T_C + T_{CL} - 20 + 0.075\,C_C - 0.075\,C_D$	—	ns
$T_{DH(n)}$	Data Out Hold (Non Interleaved)	$2\,T_C + T_{CH} - 10 + 0.075\,C_D - 0.050\,C_C$	—	ns
$T_{CAH(i)}$	Col Addr Hold (Interleaved)	$T_C + T_{CH} - 20 + 0.075\,C_A - 0.050\,C_C$	—	ns
$T_{DS(i)}$	Data Out Setup (Interleaved)	$T_{CL} - 25 + 0.075\,C_C - 0.075\,C_D$	—	ns
$T_{DH(i)}$	Data Out Hold (Interleaved)	$T_C + T_{CH} - 10 + 0.075\,C_D - 0.050\,C_C$	—	ns

FAST PAGE MODE READ AND WRITE CYCLES

Symbol	Parameter	Min	Max	Units
T_{PC}	Fast Cycle Time	$2 T_C - 5$	—	ns
T_{CP}	CAS# Precharge Time	$T_{CL} - 10$	—	ns
T_{CAS}	CAS# Width	$T_C + T_{CH} - 10 - 0.025\ C_C$	—	ns
T_{CAA}(1)	Col Address Access Time		$2 T_C - 15 - 0.075\ C_A$	ns
T_{CAC}(1)	CAS# Access Time		$T_C + T_{CH} - 15 - 0.050\ C_C$	ns
T_{CAP}(1)	Access Time from Col Precharge		$2 T_C - 25 - 0.075\ C_C$	ns
$T_{OAC(i)}$	Access Time from BEN# (Interleaved)		$T_C - 21 - 0.050\ C_B$	ns
$T_{CAH(n)}$	Col Addr Hold (Non Interleaved)	$T_C + T_{CH} - 20 + 0.075\ C_A - 0.050\ C_C$	—	ns
$T_{DS(n)}$	Data Out Setup Non Interleaved)	$T_{CL} - 25 + 0.075\ C_C - 0.075\ C_D$	—	ns
$T_{DH(n)}$	Data Out Hold (Non Interleaved)	$T_C + T_{CH} - 10 + 0.075\ C_D - 0.050\ C_C$	—	ns
$T_{CAH(i)}$	Col Addr Hold (Interleaved)	$T_{CH} - 12 + 0.075\ C_A - 0.050\ C_C$	—	ns
$T_{DS(i)}$	Data Out Setup (Interleaved)	$T_{CL} - 25 + 0.075 C_C - 0.075\ C_D$	—	ns
$T_{DH(i)}$	Data Out Hold (Interleaved)	$T_{CH} - 6 + 0.075\ C_D - 0.050\ C_C$	—	ns

DUAL PORT DRAM DATA TRANSFER CYCLE

Symbol	Parameter	Min	Max	Units
T_{DTR}	DT High to RAS# High Setup	$T_C - 10 + 0.025\ C_R - 0.075\ C_B$	—	ns
T_{DTH}	DT High from RAS# High Hold	$T_C - 10 + 0.075\ C_B - 0.075\ C_R$	—	ns
T_{RDHN}	DT Low from RAS# Low Hold	$3 T_C - 10 + 0.025\ C_B - 0.050\ C_R$	—	ns
T_{RDHP}	DT Low from RAS# Low Hold	$7 T_C - 10 + 0.025\ C_B - 0.050\ C_R$	—	ns
T_{RDHF}	DT Low from RAS# Low Hold	$5 T_C - 10 + 0.025\ C_B - 0.050\ C_R$	—	ns
T_{DLS}	DT Low to RAS# Low Setup	$T_C - 10 + 0.075\ C_R - 0.050\ C_B$	—	ns
T_{CDH}	DT Low from CAS# Low Hold	$T_C + T_{CH} - 10 + 0.025\ C_B - 0.050\ C_C$	—	ns
T_{DTC}	DT High to CAS# High Setup	$T_C - 10 + 0.025\ C_C - 0.075\ C_B$	—	ns

MASTER MODE TIMINGS $C_L = 100$ pF on all output pins
AC timings are referenced to 1.5V on clock input and 0.8V/2.0V on other pins

Symbol	Parameter	Min	Max	Units
T_{M1A}(2)	CLK to MEN Delay	—	40	ns
T_{M1B}(3)	HLDA to MEN Delay	—	45	ns
T_{M2A}(2)	CLK to A21:0, MIO, RD#, WR#, BHE# Drive	—	40	ns
T_{M2B}(3)	HLDA to A21:0, MIO, RD#, WR#, BHE# Drive	—	45	ns
T_{M3}	HREQ, MEN Inactive Delay	—	45	ns
T_{M4}	A21:0, D15:0 Float Delay	—	40	ns
T_{M5}	Async HLDA Setup	5	—	ns
T_{M8}	Read Data Setup Time	10	—	ns

MASTER MODE TIMINGS C_L = 100 pF on all output pins (Continued)
AC timings are referenced to 1.5V on clock input and 0.8V/2.0V on other pins

Symbol	Parameter	Min	Max	Units
T_{M9}	Read Data Hold Time	10	—	ns
T_{M10}	READY Setup Time	20	—	ns
T_{M11}	READY Hold Time	5	—	ns
T_{M12}	Command Valid Delay	—	35	ns
T_{M13}	Address Valid Delay	—	40	ns
T_{M14}	Write Data Valid Delay	—	40	ns
T_{M15}	Write Data Hold Time	—	40	ns
T_{M16}	Sync HLDA Setup : 01	5	—	ns
T_{M17}	Sync HLDA Setup : 02	20	—	ns
T_{M18}	CLK to HREQ Delay		35	ns

SLAVE INTERFACE TIMINGS C_L = 100 pF on all output pins
AC timings are referenced to 1.5V on clock input and 0.8V/2.0V on other pins

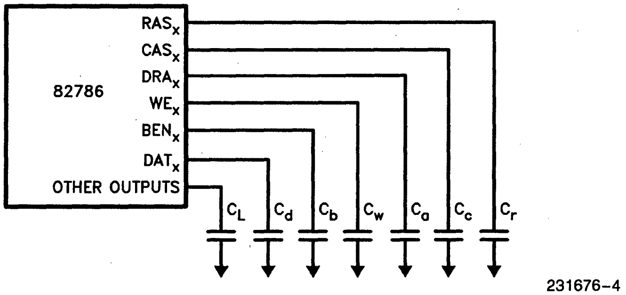
Symbol	Parameter	Min	Max	Units
T_{S1}	Active Input Setup	5	—	ns
T_{S2}	Active Input Hold Time	10	—	ns
T_{S3}	Inactive Input Setup	5	—	ns
T_{S4}	Inactive Hold Time	10	—	ns
T_{S14}	Active Command Width	$2\,T_C + 30$	—	ns
T_{S16}	A21:0, MIO, CS#, BHE# Hold Time	$2\,T_C + 30$	—	ns
T_{S17}	A21:0, MIO, CS#, BHE# Delay	—	$T_C - 20$	ns
T_{S18}	SEN Active Delay	0	35	ns
T_{S19}	Write Data Delay	0	$2\,T_C - 25$	ns
T_{S20}(4)	Write Data Hold (Memory Write)	$3\,T_C + T_{DH} + 30$	—	ns
T_{S20}	Write Data Hold (Int. Write)	$4\,T_C$	—	ns
T_{S21}	SEN Inactive Delay	0	45	ns
T_{S22}(5)	Read Data Delay (Memory Read)	0	(Note 5)	ns
T_{S22}	Read Data Delay (Int. Read)	0	$T_C + 40$	ns
T_{S23}	Read Data Hold	$5\,T_C - 15$	—	ns
T_{S24}(6)	RD/WR to SEN Delay (Mem Write)	$4\,T_C + 20$	—	ns
T_{S24}(6)	RD/WR to SEN Delay (Int. Write)	$4\,T_C + 20$	—	ns
T_{S24}(6)	RD/WR to SEN Delay (Mem Read)	$5\,T_C + 20$	—	ns
T_{S24}(6)	RD/WR to SEN Delay (Int. Read)	$7\,T_C + 35$	—	ns
T_{S25}	SEN Width (Write Cycle)	$4\,T_C - 25$	$4\,T_C + 35$	ns
T_{S25}	SEN Width (Read Cycle)	$5\,T_C - 25$	$5\,T_C + 35$	ns

VIDEO INTERFACE C_L = 50 pF on all output pins
AC timings are referenced to 1.5V on clock input and 0.8V/2.0V on other pins

Symbol	Parameter	Min	Max	Units	Notes
T_{VCYC}	VCLK Cycle Time	40	—	ns	@ 1.5V
T_{VCL}	VCLK Low Time	19	—	ns	@ 1.5V
T_{VCH}	VCLK High Time	19	—	ns	@ 1.5V
T_{VDL}	Delay VCLK to Output Valid	0	25	ns	
T_{VDH}	Output Valid Hold from VCLK	4	—	ns	@ 1.5V
T_{VS}	Input Setup Time	5	—	ns	
T_{VH}	Input Hold Time	8	—	ns	

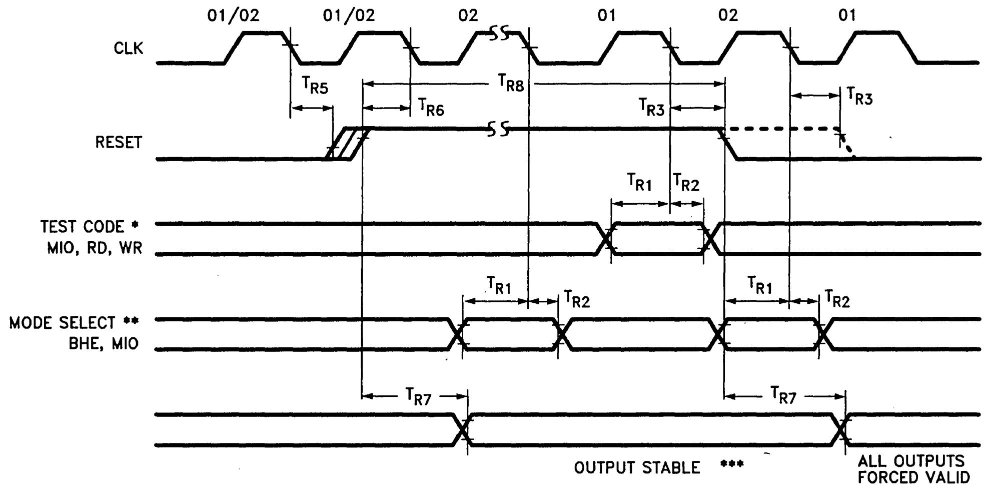
NOTES:
1. Subtract transceiver delay from these number for xl devices.
2. Valid for asynchronous interface or for synchronous interface when TM16 is satisfied. Synchronous interface requires same clock and reset input for 82786 and 80286.
3. Valid for synchronous interface when TM16 is not satisfied.
4. TS20 (memory write) is dependent on DRAM specs.
5. TS22 (memory read) is dependent on DRAM specs. This is the maximum of:
   i) $T_{RAC} - T_C + 10$
   ii) $T_C - T_{CH} + T_{CAC} + 10$
   iii) $T_C - T_{OAC} + 10$
6. TS24 numbers mentioned above are the absolute minimum values for a synchronous 80286/82786 type interface (or synchronous 80186/82786 interface with WT = 0). Add T_C to get corresponding minimum number for an asynchronous interface. Add T_C for 80186 interface with WT = 1. Typical delay from Command to SEN active will be greater than the minimum value, depending on level of activity of the 82786 and priority for external access.

AC TEST LOADS (Use capacitance values in pico farads in the timing equations)



C_d, C_b, C_w, C_a, C_c, C_r in the range 30 pF 200 pF

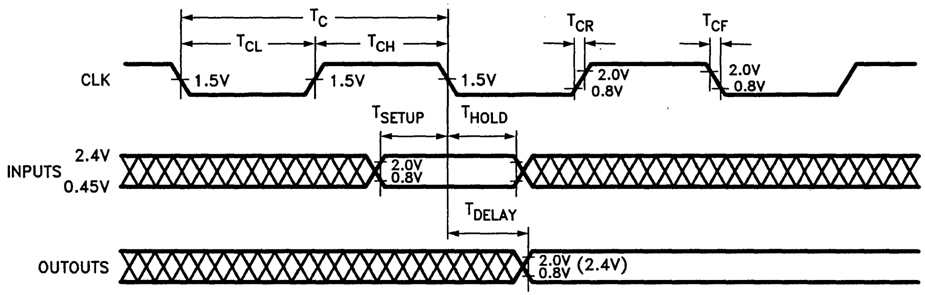
RESET TIMINGS



231676-5

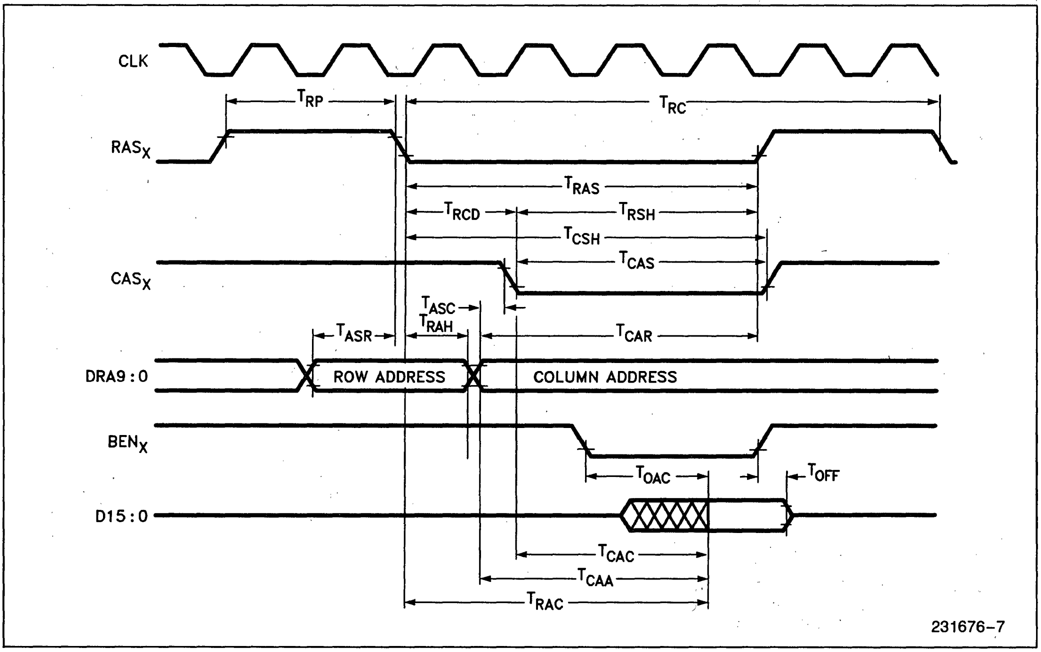
*Test code sampled at end of 01 preceding falling edge of RESET
**Mode select sampled at end of 02 preceding falling edge of RESET
***$\overline{RD}$, $\overline{WR}$, $\overline{BHE}$, MIO, A21:0, D15:0, BLANK, HSYNC, VSYNC—TRISTATED
MEN, SEN, HREQ, INTR—LOW
CAS$_x$, RAS$_x$, WEL, WEH, BEN$_x$—HIGH
DRA8:0, VDATA7:0—Indeterminate

CLOCK AND AC TEST CONDITIONS



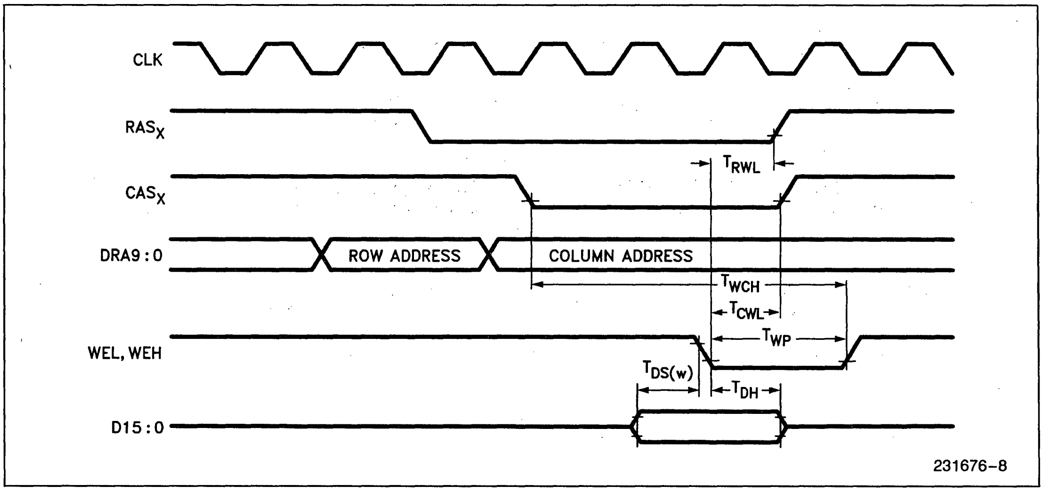
231676-6

DRAM SIGNALS—SINGLE READ CYCLE



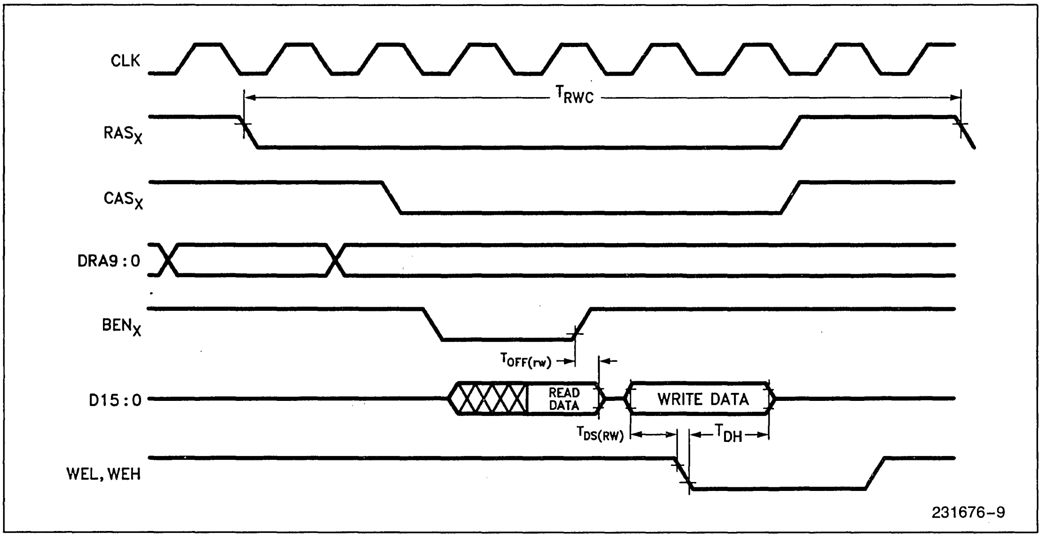
231676-7

DRAM SIGNALS—SINGLE WRITE CYCLE

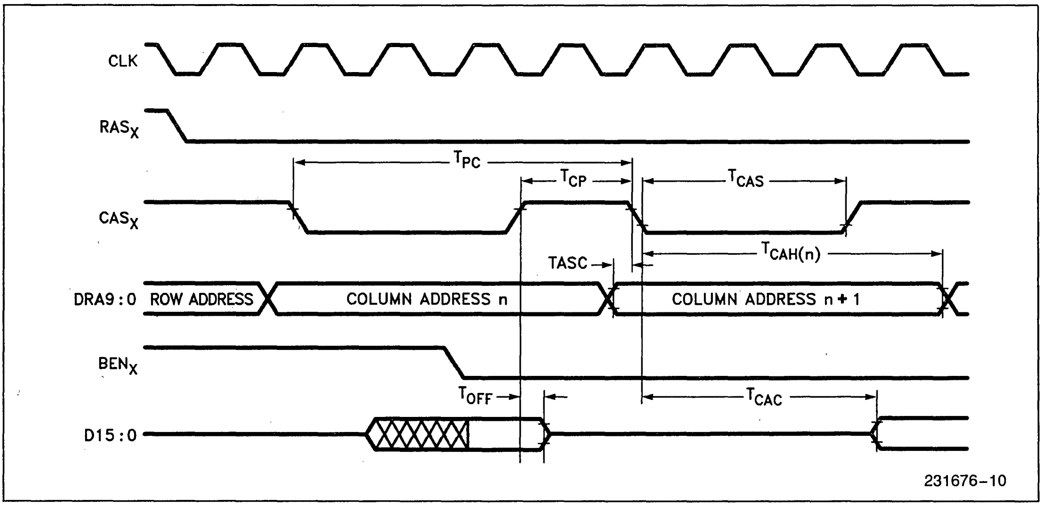


231676-8

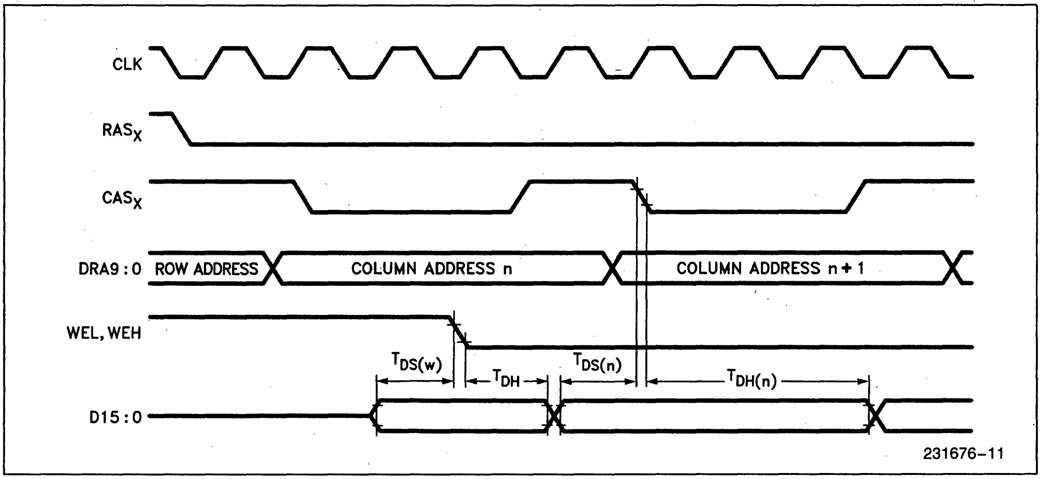
DRAM SIGNALS—READ MODIFY WRITE CYCLE



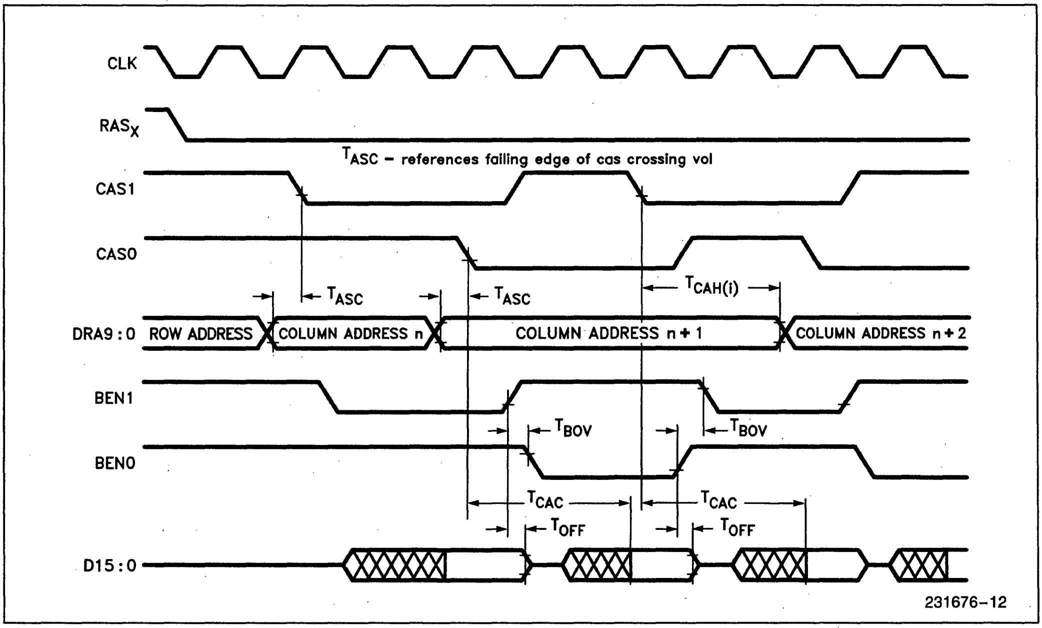
DRAM SIGNALS—NON-INTERLEAVED PAGE MODE READ



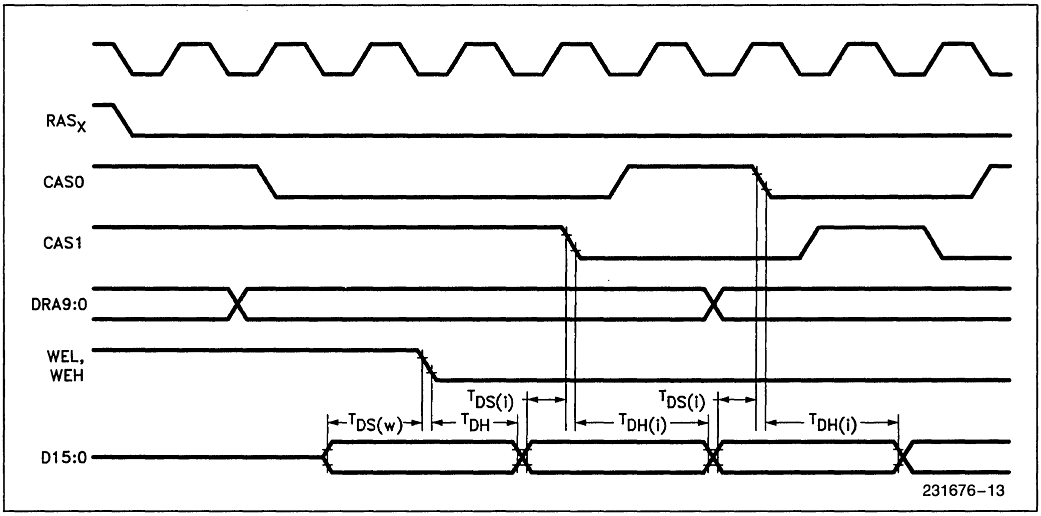
DRAM SIGNALS—NON INTERLEAVED PAGE MODE WRITE



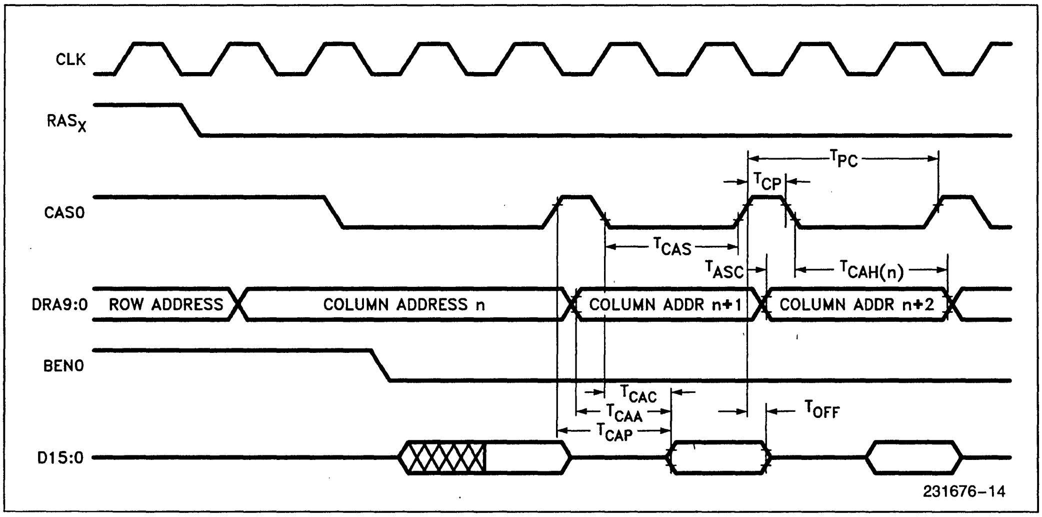
DRAM SIGNALS—INTERLEAVED PAGE MODE READ



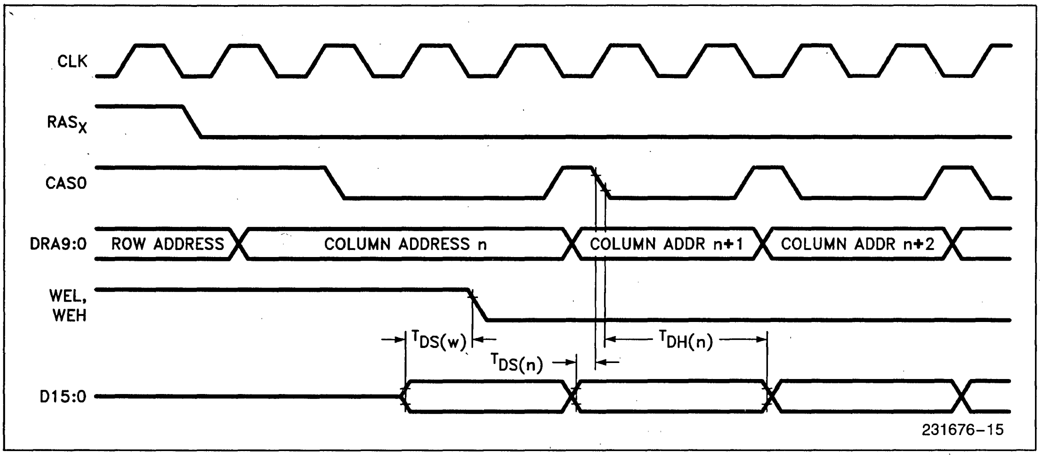
DRAM SIGNALS—INTERLEAVED PAGE MODE WRITE



DRAM SIGNALS—NON-INTERLEAVED FAST PAGE MODE READ

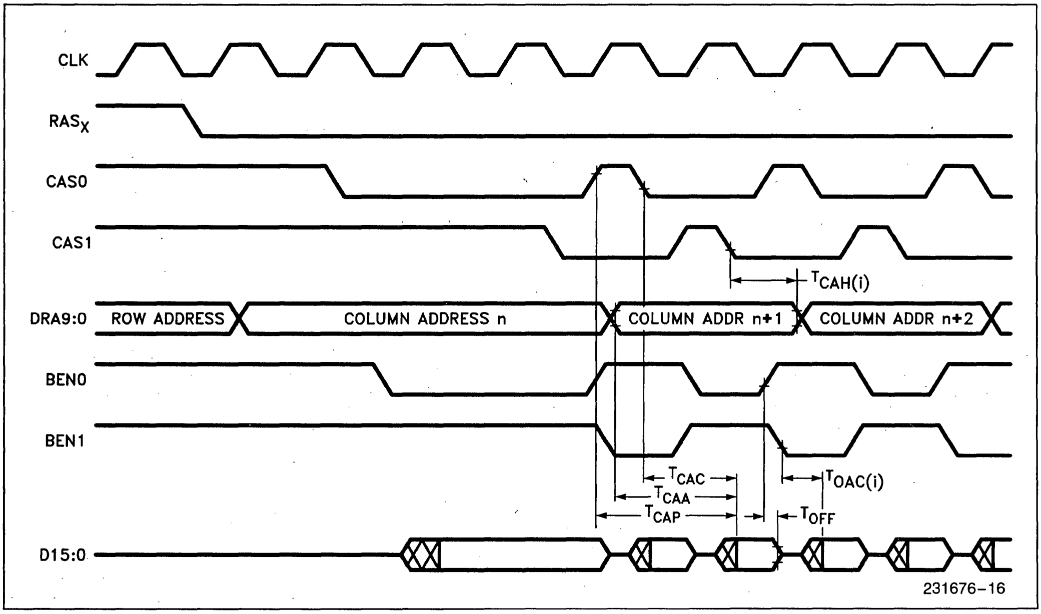


DRAM SIGNALS—NON-INTERLEAVED FAST PAGE MODE WRITE



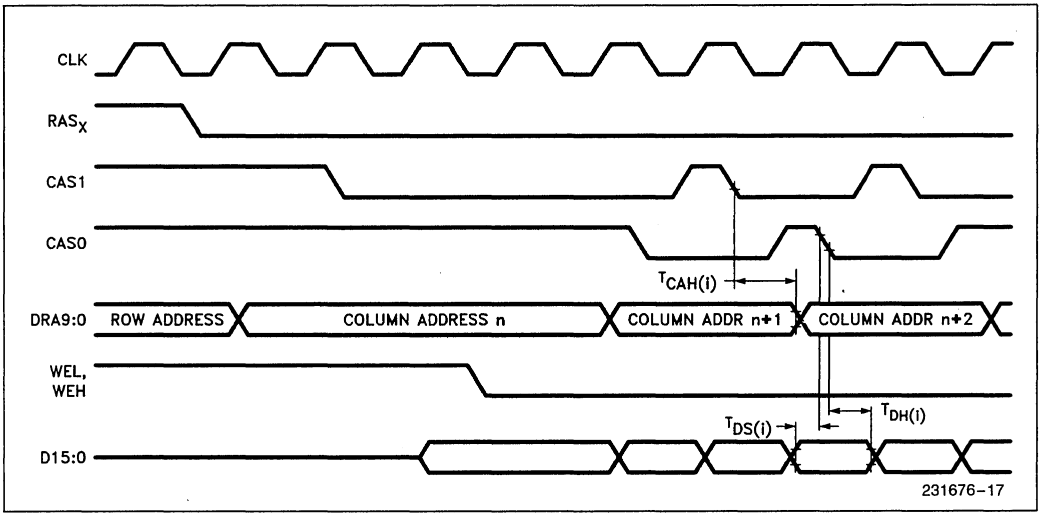
231676-15

DRAM SIGNALS—INTERLEAVED FAST PAGE MODE READ

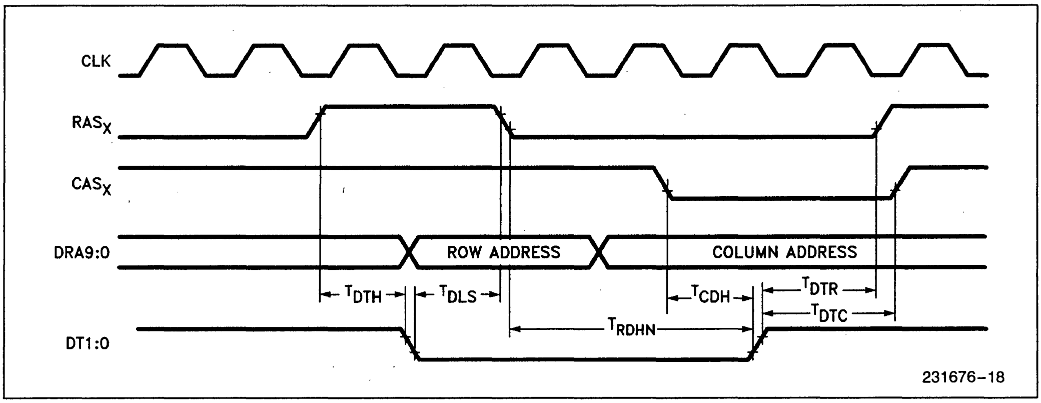


231676-16

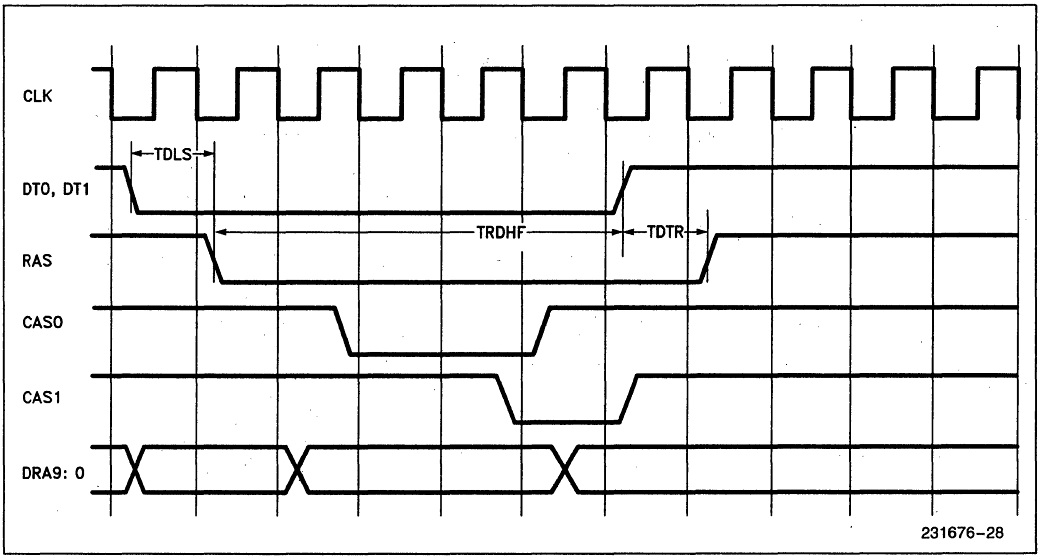
DRAM SIGNALS—INTERLEAVED FAST PAGE MODE WRITE



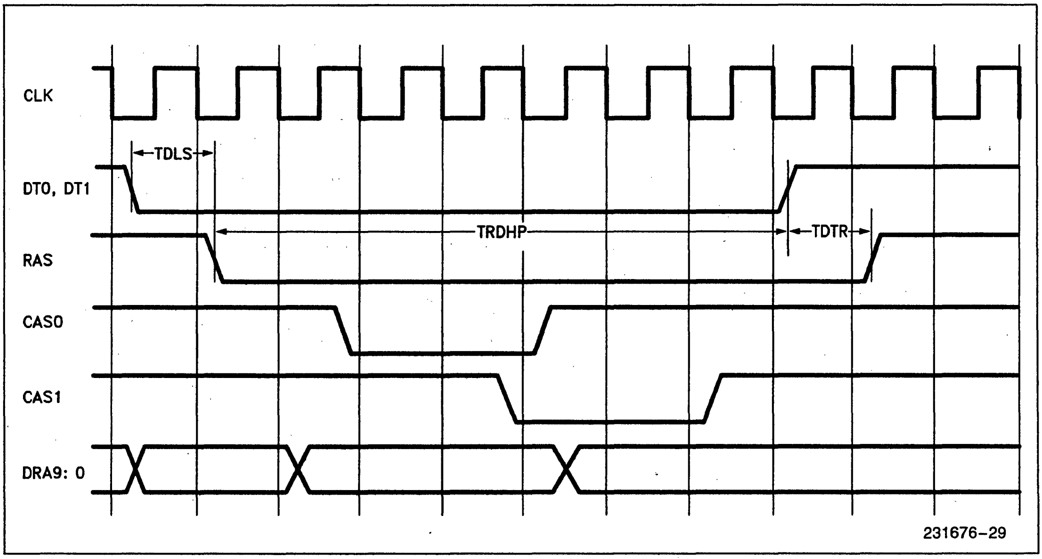
DRAM SIGNALS— NON-INTERLEAVED PAGE MODE AND FAST PAGE MODE VRAM DATA TRANSFER CYCLE



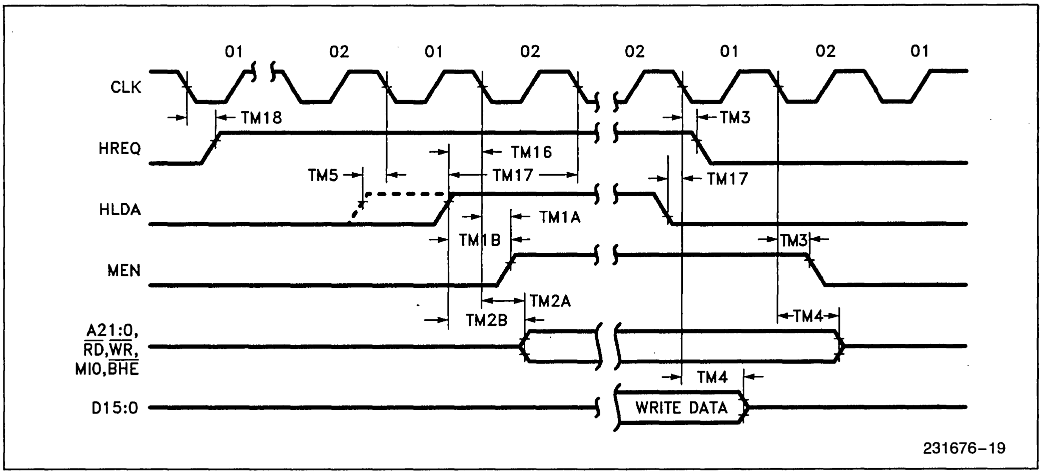
INTERLEAVED FAST PAGE MODE VRAM DATA TRANSFER CYCLE



INTERLEAVED PAGE MODE VRAM DATA TRANSFER CYCLE

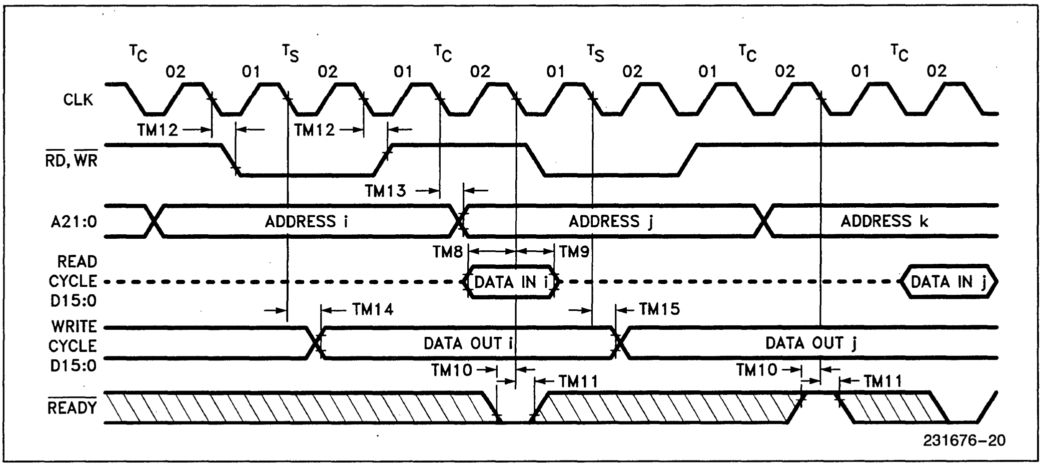


ENTERING AND LEAVING MASTER MODE



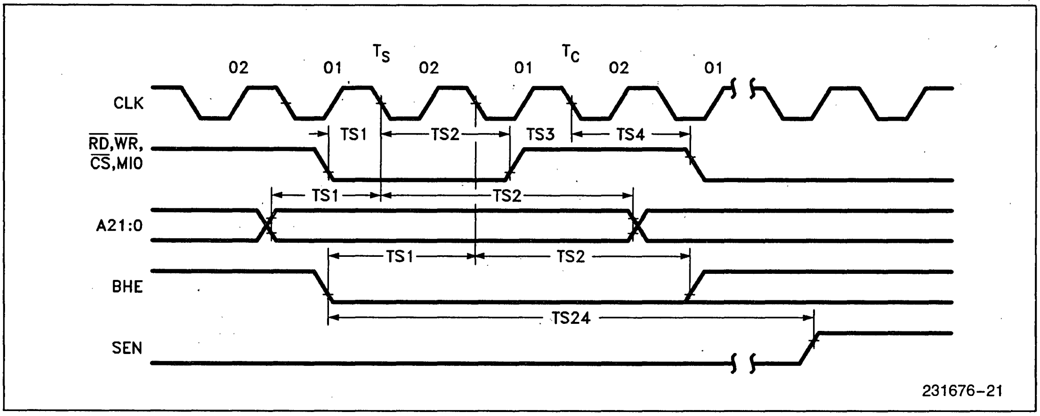
231676-19

MASTER MODE TIMINGS



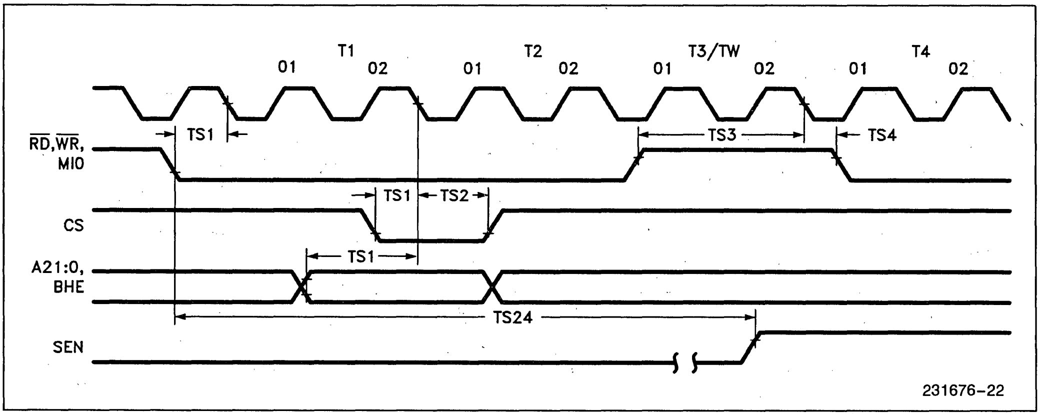
231676-20

SYNCHRONOUS 80286 (STATUS) SLAVE INTERFACE



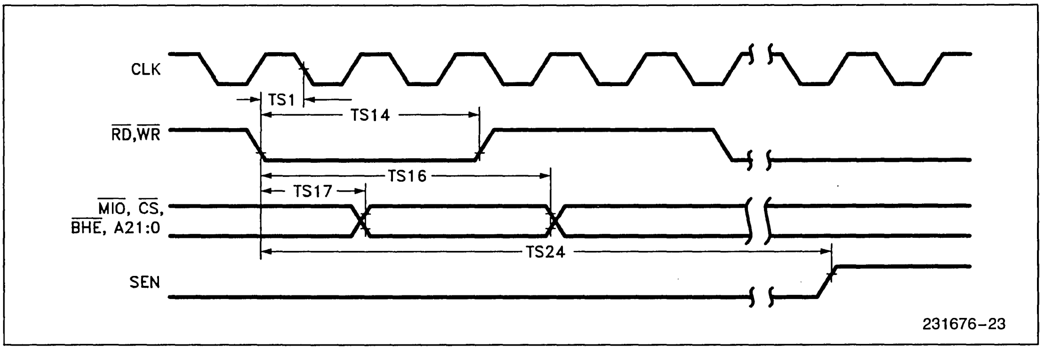
231676-21

SYNCHRONOUS 80186 (STATUS) SLAVE INTERFACE

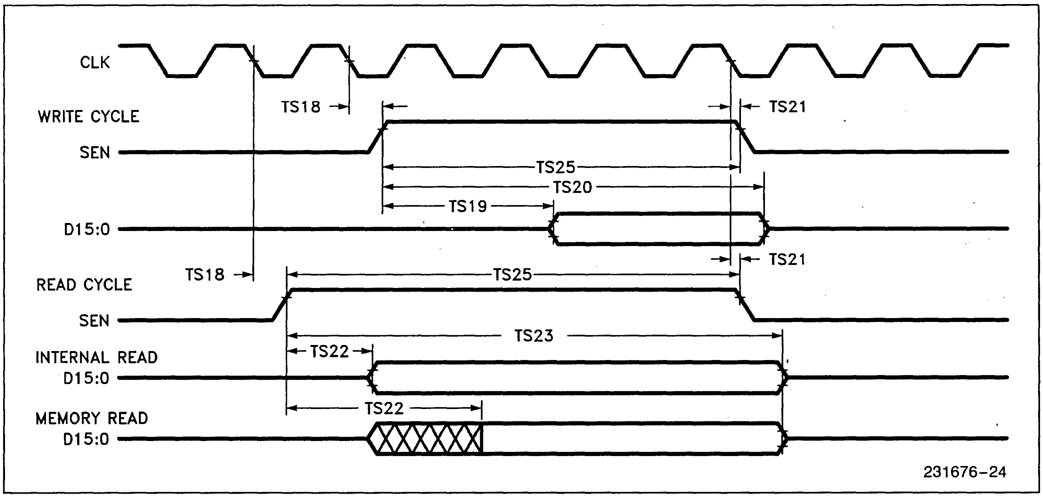


231676-22

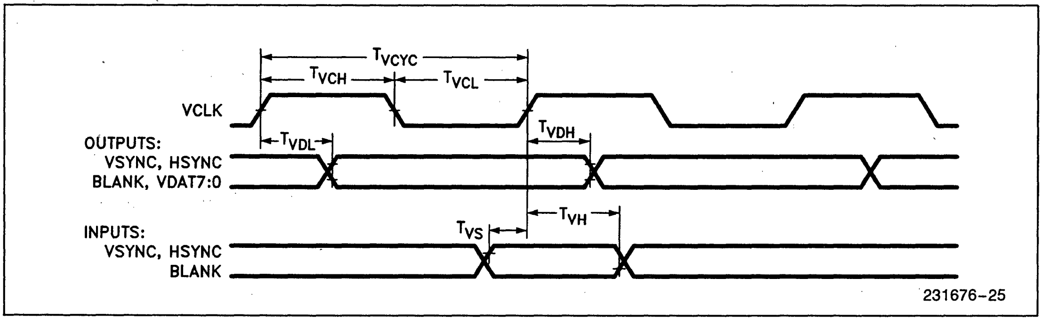
ASYNCHRONOUS SLAVE INTERFACE



SEN/DATA—SLAVE INTERFACE



VIDEO TIMINGS



Data Sheet Revision History

This 82786 datasheet, version -004, contains updates and improvements to the previous version. A revision summary is listed here for your convenience.

The sections that are significantly revised since version -003 are:

GRAPHICS PROCESSOR

Clipping Rectangle — A note was added on character clipping.

Pick Mode — Not supported for circles and arcs.

Context Registers — An updated table has been added.

CHAR Command — Opcodes corrected for all four versions of CHAR command.

INCR_POINT Command — Notes added on largest array size and on transparent mode.

Circle and Arc Commands — Note added on transparent mode.

DISPLAY PROCESSOR

Zoom Support — Information added on Interlaced zoom.

Command Execution — Information added on register ID numbers for Load and Dump Register commands.

Windows
- — Note added on VRAM strips
- — Note deleted on number of tiles per strip.
- — Note added on 2- and 4-bank Swapped Byte modes.

AC SPECS

Single Write Cycle — T_{RWL} and T_{CWL} specification values altered.

Fast Page Mode Read and Write Cycles — T_{CP}, $T_{OAC(i)}$, $T_{CAH(i)}$, and $T_{DH(i)}$ specification values altered.

Dual Port DRAM Data Transfer Cycles
- — T_{RDH} renamed to T_{RDHN}.
- — T_{RDHP} and T_{RDHF} added.

TIMING DIAGRAMS

"DRAM Signals — Data Transfer Cycle" — Renamed to "Non-Interleaved Page Mode and Fast Page Mode VRAM Data Transfer Cycle"

Two timing diagrams on Interleaved Page Mode and Fast Page Mode VRAM Data Transfer Cycles added.

intel®

APPLICATION NOTE

AP-268

November 1986

A Low Cost and High Integration Graphics System Using 82716

VIREN SHAH
APPLICATIONS ENGINEER

Order Number: 231679-001

1.0 INTRODUCTION

The role of graphics in personal computers and engineering workstations is becoming increasingly important. Lately, the graphics software which support separate windows for separate tasks have made multitasking an attractive feature on the PCs. To support this feature, the system must handle windows efficiently.

Windowing and graphics make very heavy demands on processing power. If a main processor in a PC or graphics workstation is used to move information around on the display, the response time becomes unacceptably slow. While keeping the system cost low, the VSDD solves this problem by supporting hardware windows and providing other additional features on chip.

The purpose of this application note is to familiarize the reader with the 82716 VSDD (video storage and display device) operation. This note will guide the reader in designing a text and graphics system. This document is intended as a supplement to the VSDD data sheet and user's manual.

The VSDD is aimed at applications needing a low cost and highly integrated color graphics controller for both bit-mapped and alphanumeric displays. The chip's high level of integration allows designers to build graphics systems with a very low chip count thus improving the reliability of their equipment. The system designed in this note needs only 7 components besides VSDD. VSDD's high integration and low cost makes it ideal for compact, low cost video displays found in home computers, home information systems and industrial and commercial monitoring equipments. The chip also supports videotex standards such as NAPLPS, TELETEL, PRESTEL, and CAPTAIN.

82716 Description

The block diagram of the VSDD is shown in Figure 1.

The VSDD has the following features:

- Manages up to 16 bit-map and character objects.
- On chip 16/4096 color palette.
- On chip DRAM controller.
- Up to 640 x 512 pixel resolution.
- Extremely simple interface to Intel 8 bit and 16 bit CPUs.
- On chip D/A converters.
- Low chip count display controller.
- Analog or digital video outputs.
- Up to 512K bytes of display memory.
- 2, 4, or 8 bits/pixel.

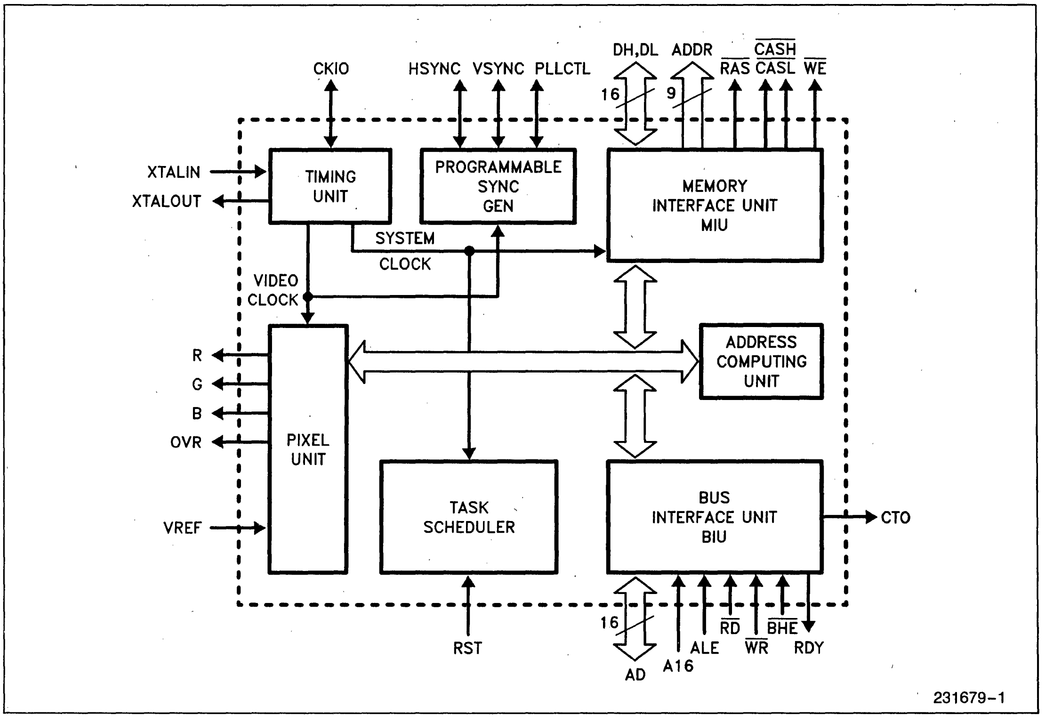


Figure 1. VSDD Block Diagram

The VSDD can control up to 16 simultaneous windows. It can change the position and contents of any window independently. This allows easy scrolling and animation. It interfaces easily to 8088 and 8086 family microprocessors without glue logic. The chip controls CRT monitors with up to 640 x 512 x 4 resolution. The on chip DRAM controller manages up to 512K bytes of display RAM. A pair of pixel buffers—each can hold 640 pixels at 4 bits/pixel—help speed up the operation by providing a continuous video data output stream.

The VSDD also integrates a color lookup table (storing 16 colors from a possible 4096), three 4 bit D/A converters and a programmable sync and timing generator. A microprocessor, its program ROM, DRAMs and a VSDD will complete a workstation—less than 10 chips in all. The VSDD also provides digital video outputs. 8 bits/pixel digital output combined with external color look up table and DACs can provide 256 colors. The VSDD supports overlapped objects. Transparent windows too are supported by the display controller.

The screen image is constructed from various user-specified objects residing in the VSDD's display memory (mapped into the processor's address space). Figure 2 shows how the CPU's address space is mapped onto the VSDD's space. The data window in the CPU space maps onto the data segment in the VSDD space and the register window maps into the register segment. The CPU uses these windows to access the display RAM. The register segment length is fixed at 32 bytes. But the data window length can vary from 4K bytes up to 64K bytes. The 512K bytes of display memory can be thought of as 8 banks of 64K bytes each. The CPU can access only one bank at a time. But all the eight banks are accessible via memory mapping, thus allowing it effectively to access all of 512K bytes.

Pixels are taken directly from the memory for display on the screen. Characters are constructed using user-defined RAM-based character generators. The VSDD takes the object data from its memory, buffers it and runs it through the color look-up table and D/A converters to produce video signals. These signals then drive the display.

There are two segments in the display memory—the data segment and the register segment. The data segment contains the actual object data, window attributes such as object's position on the screen, object's width, etc., access table, color look up table and two character

Memory Mapping

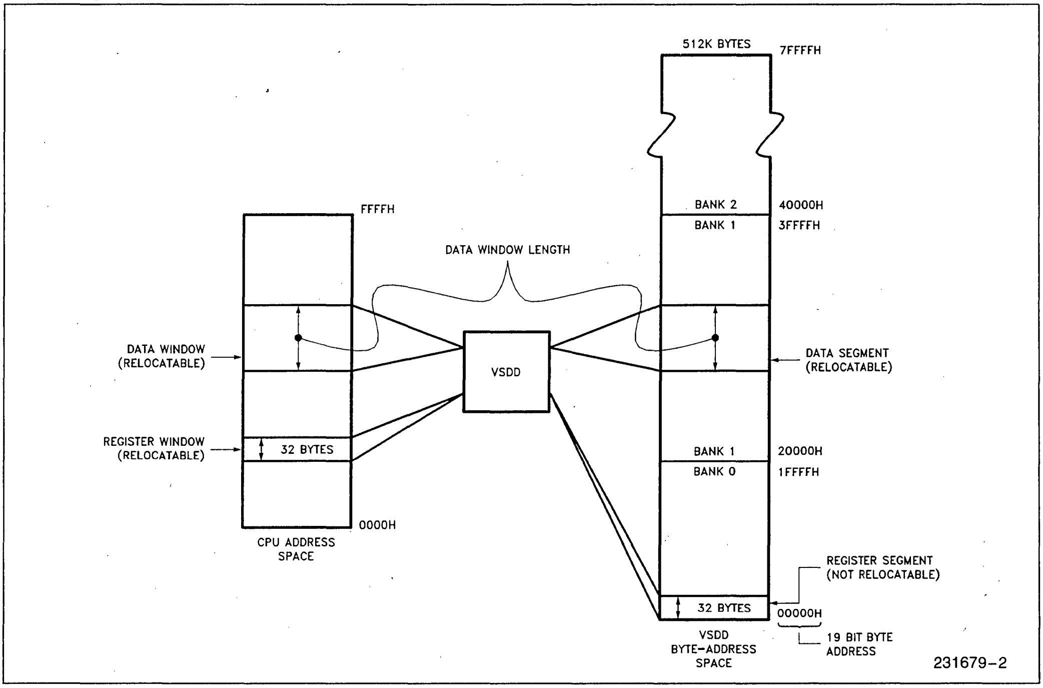


Figure 2

generators. The access table contains the vertical position and priority of each object. The data segment can be placed anywhere within the external 512K byte display RAM.

Information on the system's configuration is kept in a 32-byte register segment, which always begins at hexadecimal address 00000 in the display DRAM. The VSDD reads the register segment once per frame to update its on-chip registers. The register segment stores the size and speed of DRAM, the raster parameters and the base addresses of the other tables stored in the data segment. Figure 3 shows the register segment.

			DRAM Loc.
R15	Horiz. Constant 3	Vert. Constant 3	1EH
R14	Horiz. Constant 2	Vert. Constant 2	1CH
R13	Horiz. Constant 1	Vert. Constant 1	1AH
R12	Horiz. Constant 0	Vert. Constant 0	18H
R11	Access Table Base Address Counter		16H
R10	Character Base Address		14H
R9	Color Table Base Address		12H
R8	Access Table Base Address		10H
R7	Object Descriptor Table Base Address		0EH
R6	Priority Access Quantity		0CH
R5	Data Segment Base Address		0AH
R4	Data Length Mask		08H
R3	Data Window Base Address		06H
R2	Register Window Base Address		04H
R1	Video Configuration Register 1		02H
R0	Video Configuration Register 0		00H

Figure 3

The CPU programs the data segment and the register segment. After these segments are initialized the VSDD assumes control of the CRT and controls refresh of the DRAM. This relieves the CPU of maintaining the display thus considerably increasing the performance of the graphics system.

The chief purpose of this application note is to show the user how to program data and register segments by describing a specific design example.

2.0 DESIGN EXAMPLE

In this design, the VSDD is used to display 3 bit-mapped objects and one character object on the screen. Hardware is very simple and compact. Only seven chips besides the VSDD are needed to build the system. 80186 (8 MHz) is used as the CPU. 4 DRAMs (64K x 4) give a total display memory of 128K bytes. This much memory can support a resolution of 640 x 400 at 4 bits/pixel. No glue logic is needed between the 80186 and the VSDD for the bus interface. The display used is an IBM color monitor. The digital video outputs are used to drive the monitor through the line drivers (74LS244). The VSDD generates active low VSYNC and HSYNC. Since the IBM color monitor needs active high VSYNC and HSYNC, the VSDD generated sync signals are inverted (74LS368). By using PAL for the random logic in this design the total chip count for this graphics system can be reduced to seven including the VSDD. Appendix E shows the circuit schematics.

2.1 Initialization

The VSDD and the register segment must be initialized before a display can be obtained. The RESET pin on the 82716 is active high. It is a high impedance input to a Schmitt Trigger. On power up, RESET should be held active long enough to allow the VSDD system oscillator (on XTALIN) to start, and then be held for a minimum of 20 clock periods with the oscillator running.

RESET STATUS

RESET puts the 82716 into a special initialization mode. When RESET goes low, the device commences generating refresh cycles to the external DRAM. The CPU should not try to access the DRAM for at least 10 μs after RESET has been released. The status of the device at this time is as follows:

Control Bit	Reset Value	Effect
TMM TMS	0 0	Single Mode
DEN	0	No Display
DEI	0	Analog Mode
MAS	0	Sync pins are in high impedance
RE	0	CPU can access DRAM only to write to it
PCE	0	Priority Access Mode disabled
FAE	0	RDY pin emits Ready signal
EVC	0	Video clock is from XTALIN signal

DEN = 0 means no display is being generated. DEI = 0 means the device is in the analog mode, and, since DEN = 0, the R, G, B, and 0 pins are emitting retrace black.

MAS = 0 configures the sync pins in high impedance state. EVC = 0 would normally configure the CKIO pin as an output, but in the initialization mode CKIO is in a high impedance state.

The device comes out of reset with the following default values for the screen constants:

HCO = 1	HSYNC Width = 32 XTALIN periods	4.0 μs
HC1 = 2	AHZ Start Time = 48 XTALIN periods	6.0 μs
HC2 = 5	AHZ Stop Time = 96 XTALIN periods	12.0 μs
HC3 = 8	HSYNC Period = 144 XTALIN periods	18.0 μs
VCO = 1	VSYNC Width = 2 lines	36.0 μs
VC1 = 2	AVZ Start Time = 3 lines	54.0 μs
VC2 = 3	AVZ Stop Time = 4 lines	72.0 μs
VC3 = 7	VSYNC Period = 8 lines	144 μs

This is called the "fast frame" mode. Its main purpose is to give the PSG something to work with that is not contradictory (such as HCO > HC3) until the CPU has written the correct values into DRAM. The timings listed assume a 125 nsec clock (8.0 MHz) at XTALIN.

Most important to the initialization procedure are the reset values of the Register and Data Windows:

Register Window Base Address: 00400H
Data Window Base Address: (undefined)

INITIALIZATION PROCEDURE

The first access must be a write cycle to Register R0 at 00400H. In the first write cycle the CPU should program the DRAM configuration bits DS1, DSO and DOF for 64K x 4 DRAMS employed in this design.

This first write cycle should leave DEN and UCF at 0. UCF (Update Control Flag) should be left clear till the entire Register Segment has been initialized, lest the device "update" its on-chip control bits with random data.

After the first write, the CPU can continue to initialize the Register Segment by writing to the Register Window addresses, 00400H through 0041FH. All this information is mapped by the VSDD into its register segment from 00000H to 0001FH. Excpet for the control bits written into R0, the values as written do not take effect as long as UCF = 0.

After the Register Segment has been completely initialized, one more write cycle is directed to R0 to set UCF to 1, while holding DEN = 0. The CPU now waits 1 frame time (144 μs, in the "fast frame" mode). At the end of the frame time in progress, during the FRAMESTOP sequence, the VSDD will be flagged by UCF = 1 to update its internal registers from the Register Segment in DRAM. Further access to the Register Segment by the CPU must be through the newly defined Register Window.

Now, through the newly defined Data Window, the CPU can commence initializing the display data. The DEN flag in R0 should be set(1) after the display data is loaded in the DRAM. This bit will then enable the display.

At the beginning the CPU programs R0 at 00400H as follows:

```
                          DEN      UCF
R0  011  00000  011  1  0  0  1    0    6072H
DEN and UCF are set to zeroes.
```

After the Register Segment is completely initialized, CPU sets the UCF bit by writing to R0.

```
R0  011  00000  011  1  0  0  1  1  6073H
```

The VSDD would now update its on-chip control registers with the data programmed by CPU in the Register Segment.

The CPU then programs the display data through the data window,. After the data has been written into memory, the CPU enables the display by setting DEN bit.

```
R0  011  00000  011  1  1  0  1  1  607BH
```

The register segment is programmed as follows to obtain a display shown in Figure 4. The VSDD reads the register segment on every frame to update its on-chip registers. The reader should refer to the user's manual for description of the bits in the register segment. See Appendix A for horizontal and vertical sync constants. The VSDD DRAM is word addressable while the CPU address space is byte addressable.

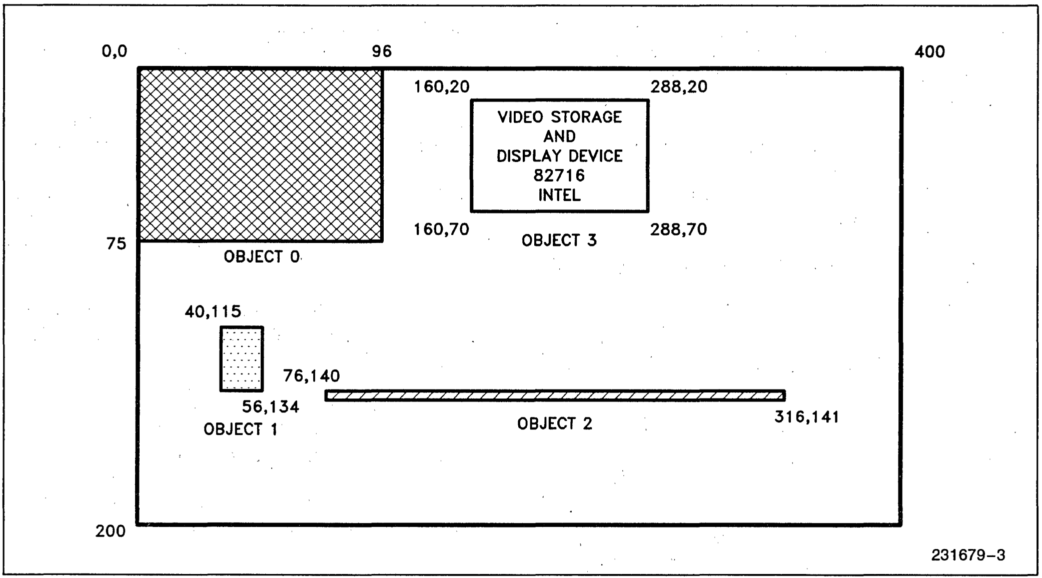


Figure 4. Display Screen

REGISTER SEGMENT

CPU ADDR	DRAM WORD ADDR	REGISTER	CONTENTS		COMMENTS
0000H	00000H	R0	011 00000 01111011		607BH
			Duty Cycle	011	50%
			Blink Rate	00000	7.5 Hz for 60 Hz frame rate
			DS1 DS0 DOF	011	64K x 4 DRAMs
			HRS	1	640 Pixels horizontally
			DEN	1	Display Enabled
			SAB	0	Fast DRAM
			DEI	1	Digital Outputs on RGB & OVR pins
			UCF	1	Update all the registers on every frame
0002H	00001H	R1	1010 010 00 00101 0 0		A414H
			Char height	1010	10 Scan lines
			INL	0	Non interlaced
			MAS	1	HSYNC, VSYNC are outputs
			SM	0	Non composite SYNC mode
			TMM, TMS	00	Twin mode is disabled
			EVC	0	CKIO is O/P. Video Clk derived from XTALIN.
			PCE	1	Priority counter enable
			FAE	0	Use RDY as ready signal
			RE	1	CPU can read the DRAM
			PSA	0	GCLK = 1/16 XTALIN
			PRE	0	Disable pipeline read
0004H	00002H	R2	0000 0000 00000 11 0		0006H
			RWBA	000H	RWBA = 0
			TF2 TF1	11	Digital pixel codes
			ME	0	No margin

REGISTER SEGMENT (Continued)

CPU ADDR	DRAM WORD ADDR	REGISTER	CONTENTS		COMMENTS
0006H	00003H	R3	0000 00 0101000 000		0140H
			DWBA	00000	A16 Should be low
			Screen	0101000 111	= Rt edge of the screen is at
			Boundary		X=327
0008H	00004H	R4	1 0000 00000000000		8000H
			Length Mask	10000	64K byte data window
000AH	00005H	R5	0000 0000 0000 0000		0000H
			Data		
			Segment		
			Base S16–S12	00000	Bank 0
			Bank Select Bits	0 0	
000CH	00006H	R6	0000 0000 0000 1010		000AH
			PAQ 1010		CPU is allowed 10 DRAM accesses during line building
000EH	00007H	R7	0000 0101 0000 0000		0500 H
			ODTBA	0500H	Object descriptor table starts at 0500H in bank 0
0010H	00008H	R8	0000 0000 0010 0000		0010 H
			ATBA	0010H	Access table starts at 0010H in bank 0
0012H	00009H	R9	0000 0001 1000 0000		0180H
			CTBA	0180H	The color table is not used in this design. The space is reserved for future use.
0014H	0000AH	R10	0000 0000 0010 0011		0023H
			CGBA0	0010	Char gen 0 starts at 2000H in bank 0
			CGBA2	0011	Char gen 1 starts at 3000H in bank 0
0016H	0000BH	R11	0000 0000 0010 0000		Access table address counter = access table base address (initially) = 0010H
0018H	0000CH	R12	000001 0000000010B		
			HC0		HSYNC Width = 4 μs
			VC0		VSYNC Width = 198 μs
001AH	0000DH	R13	000100 0000100100B		
			HC1		AHZ Start = 10 μs
			VC1		AVZ Start = 2.5 ms
001CH	0000EH	R14	011101 0011101100		
			HÇ2		AHZ Stop = 60 μs
			VC2		AVZ Stop = 16.17 ms
001EH	0000FH	R15	100000 0011110100		Hori. sweep rate = 66 μs
					Vert. sweep rate = 16.67 ms

NOTE:
See Appendix I on how to program registers R12–R15.

3.0 THE DATA SEGMENT

The actual object data and the different tables are stored in the data segment by the 80186. There are 5 tables in the data segment: The access table, the object descriptor table, the color lookup table and two character generators. Since digital outputs are used to drive the monitor in this design, the color lookup table is left blank.

3.1 Access Table

The access table contains the vertical start and end locations of each object. The table begins at the locations designated by access table base address register, R8, in the register bank. Each entry in the table contains 16 bits—each bit representing one object. Bit number 0 has the lowest priority and bit number 15 the highest. The first entry in the table corresponds to the topmost line on the screen and so on. Each entry indicates to the VSDD which objects are to be present on this line of the display. If a bit is set (1), then there is no change in the objects display status; that is, if the object did not appear on the previous line, it will also not appear on this line. If the object's access flag is set to zero, then the display status is reversed from what it was on the previous line. The VSDD assumes that at the beginning of a frame all objects are turned off (1's).

The access table for the screen in Figure 4 is shown in the following pages. The screen has 400 x 200 resolution. There are 200 entries in the table for 200 vertical lines on the screen. Object 0 starts on line 1 and ends at line 75. Bit 0 is set to zero at line 1 to turn the object 0 on and again set to 0 at line 76 to turn the object off. Note that the 80186's address space is byte addressable and the VSDD's space is word addressable.

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

Figure 5. Access Table Word

ACCESS TABLE

CPU ADDR	DRAM WORD ADDR	ACCESS FLAGS																
		b 15	b 14	b 13	b 12	b 11	b 10	b 9	b 8	b 7	b 6	b 5	b 4	b 3	b 2	b 1	b 0	
0020H	00010H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	Line 1
0022H	00011H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0024H	00012H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0026H	00013H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0028H	00014H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
002AH	00015H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
002CH	00016H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
002EH	00017H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0030H	00018H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0032H	00019H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 10
0034H	0001AH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0036H	0001BH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0038H	0001CH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
003AH	0001DH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
003CH	0001EH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
003EH	0001FH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0040H	00020H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0042H	00021H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0044H	00022H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0046H	00023H	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	Line 20
0048H	00024H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Turn on the
004AH	00025H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	character
004CH	00026H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	object
004EH	00027H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0050H	00028H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0052H	00029H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0054H	0002AH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0056H	0002BH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0058H	0002CH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

ACCESS TABLE (Continued)

CPU ADDR	DRAM WORD ADDR	ACCESS FLAGS																
		b 15	b 14	b 13	b 12	b 11	b 10	b 9	b 8	b 7	b 6	b 5	b 4	b 3	b 2	b 1	b 0	
005AH	0002DH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 30
005CH	0002EH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
005EH	0002FH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0060H	00030H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0062H	00031H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0064H	00032H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0066H	00033H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0068H	00034H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 37
0070H	00035H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 38
...	...	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
...	...	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
...	...	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
...	...	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	Line 71
...	...	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	(Turn off the
...	...	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	character object)
00B4H	0005AH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 74

ACCESS TABLE (Continued)

		ACCESS FLAGS																
CPU ADDR	**DRAM WORD ADDR**	**b 15**	**b 14**	**b 13**	**b 12**	**b 11**	**b 10**	**b 9**	**b 8**	**b 7**	**b 6**	**b 5**	**b 4**	**b 3**	**b 2**	**b 1**	**b 0**	
00B6H	0005BH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	Line 76 (Turn off Obj. O)
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
00FEH	0007FH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 111

ACCESS TABLE (Continued)

CPU ADDR	DRAM WORD ADDR	ACCESS FLAGS																
		b 15	b 14	b 13	b 12	b 11	b 10	b 9	b 8	b 7	b 6	b 5	b 4	b 3	b 2	b 1	b 0	
00100H	00080H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 112
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	Line 115 (Turn on Obj. 1)
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
Rectangle		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 120
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
Object #1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 125
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 130
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 134
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	Line 135 (Turn off Obj. 1)
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	Line 140 (Turn on Obj. 2)
Horizontal Line		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
Object #2		1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	Line 142 (Turn off Obj. 2)
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 151

Up to 200 Lines

ACCESS TABLE (Continued)

CPU ADDR	DRAM WORD ADDR	ACCESS FLAGS																
		b 15	b 14	b 13	b 12	b 11	b 10	b 9	b 8	b 7	b 6	b 5	b 4	b 3	b 2	b 1	b 0	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 200

3.2 Object Descriptor Table:

This table contains a 4-word object descriptor field for each object in the display. The table starts at the location specified by the OBTBA register, R7. This field specifies the base address of the object in the RAM, horizontal position, its width and other attributes. The descriptor fields for bit-map and character objects are shown in Figure 6.

Bitmapped

N: Current Object Entry Address N15–N0																HI
O: Object Base Address O15–O0																
W: Object Width						X: X0 Coordinate										
0	0	0	0	C/B	R1	R0	0	O17	O16	OBL	BLA	0	TDE	C1	C0	LO

Character

Z: Slice No.				N: Current Object Entry Address N11–N0												HI
O: Object Base Address O15–O0																
W: Object Width						X: X0 Coordinate										
Y: Slice No.				C/B	R1	R0	CRS	PSE	FAD	OBL	BLA	HCR	TDE	C1	C0	LO

Figure 6

Objects are rectangular windows on the screen. The object data begins in the display RAM at the object base address specified in the object descriptor field. The length of the data file depends on the objects height, width and resolution. The width is specified in 4-word units by the "W" field. In this design a 4 bits/pixel specification is chosen. Hence, each 4-word unit represents 16 pixels. Objects 0 in Figure 4 is 6 x four word wide, that is 96 pixels wide. The object descriptor field and object data for each of the objects in Figure 4 are as follows:

OBJECT 0 DESCRIPTOR FIELD

Fill 75 Lines on the Screen with One Color

CPU ADDR	DRAM WORD ADDR	CONTENTS	COMMENTS
0A00H	00500H	0000011000000000	0600H
		C/B = 0	Bit mapped object
		R1R0 = 11	4 bits/pixel
		017, 016 = 00	Object is in bank 0
		OBL = 0	No blinking
		BLA = 0	Object is not turned off
		TDE = 0	Non transparent pixels
		C1C0 = 00	Don't care for 4 bits/pixel
0A02H	00501H	000110 0000000000	
		WIDTH: 000110	6 four word wide = 96 pixels wide
		X 0000000000	Object starts at the left edge of the screen
0A04H	00502H	0001 0000 0000 0000	
		Object base address	01000H
0A06H	00503H	0001 0000 0000 0000	
		Current object entry	01000H

OBJECT DATA

OBJECT 0

OBJECT BASE ADDR = 01000H

CPU ADDR	DRAM WORD ADDR	PIXEL DATA	
2000H	01000	8888H	Line 1
2002H	01001	8888H	
2004H	01002	8888H	
2006H	01003	8888H	
2008H	01004	8888H	24 Words = 96 Pixels
. . .	01005	8888H	wide @ 4 bits/pixel
	01006	8888H	
	01007	8888H	
	01008	8888H	
	01009	8888H	
	0100A	8888H	
	0100B	8888H	
	0100C	8888H	
	0100D	8888H	
	0100E	8888H	
	0100F	8888H	
	01010	8888H	
	01011	8888H	
	01012	8888H	
	01013	8888H	
	01014	8888H	
	01015	8888H	
	01016	8888H	
202EH	01017	8888H	End of Line 1
	. . .	. . .	
	. . .	. . .	

OBJECT 0 DATA

CPU ADDR	DRAM WORD ADDR	PIXEL DATA	
2060H	01030H	8888H	Line 3
	. . .	. . .	
	. . .	. . .	
	016F0H	8888H	Line 75
	016F1H	8888H	
	016F2H	8888H	
	016F3H	8888H	
	016F4H	8888H	
	016F5H	8888H	
	016F6H	8888H	
	016F7H	8888H	
	016F8H	8888H	
	016F9H	8888H	
	016FA	8888H	
	016FB	8888H	
	016FC	8888H	
	016FD	8888H	
	016FE	8888H	
	016FG	8888H	
	01700	8888H	
	01701	8888H	
	01702	8888H	
	01703	8888H	
	01704	8888H	
	01705	8888H	
	01706	8888H	
2EOE	01707	8888H	End of line 75

OBJECT 1 DESCRIPTOR FIELD

RECTANGLE 20 SCAN LINES/
16 PIXELS WIDE

CPU ADDR	DRAM WORD ADDR	CONTENTS	COMMENTS
0A08H	00504H	0000 0 11 00 00 00 00	0600H
		C/B = 0	Bit Mapped Object
		R1R0 = 11	4 Bits/Pixel
		017, 016 = 00	Object in bank 0
		OBL = 0	No Blinking
		BLA = 0	Object is not turned off
		TDE = 0	Non-transparent pixels
		C1C0 = 00	Don't care
0A0AH	00505H	000001 0000010100	
		WIDTH = 000001	1 Four word wide = 16 Pixels wide
		*X = 0000010100	Objects starts at X = 20
0A0CH	00506H	0001 0111 0000 1010	Object base addr 0170AH
0A0EH	00507H	0001 0111 0000 1010	Current obj. entry 0170AH

***NOTE:**
When HRS = 1, unit displacement in X direction moves the object by 2 pixels. Thus the object 1 will start at pixel number 40.

OBJECT 1 DATA

CPU ADDR	DRAM WORD ADDR	PIXEL DATA	
2E14	01070H	7777H	Line 1
	...	7777H	
	...	7777H	
	...	7777H	
	0170EH	7777H	Line 2
	0170FH	7777H	Line 2
	01710H	7777H	Line 2
	01711H	7777H	Line 2
	...	...	
	...	...	
	...	...	
	...	...	
	...	...	
	...	...	
	...	...	
	...	...	
	01756H	7777H	Line 20
	01757H	7777H	...
	01758H	7777H	...
2EB2	01759H	7777H	...

OBJECT 2 DESCRIPTOR FIELD

HORIZONTAL 2 SCAN LINES/
240 PIXELS WIDE

CPU ADDR	DRAM WORD ADDR	CONTENTS	COMMENTS
0A10H	00508H	0000 011 00 00 00 00	Same as obj 0 and obj 1
0A12H	00509H	001111 00001 00 11 0	
		WIDTH = 001111	15 four word wide = 250 pixels wide
		X = 38	
0A14H	0050AH	0001 0111 0110 0000	Obj base addr. 01760H
0A16H	0050BH	0001 0111 0110 0000	Current obj entry = Obj base address (Initially)

OBJECT 2 DATA

CPU ADDR	DRAM WORD ADDR	PIXEL DATA	
2EC0H	01760H	5555H	Line 1
	...	5555H	
	...	5555H	
	...	5555H	
	...	5555H	
	...	5555H	60 Words
	...	...	
	...	...	
	...	5555H	
	...	...	
	...	...	
	0179C	5555H	Line 2
	...	5555H	
	...	5555H	
	...	...	
	...	...	
	...	...	
2FAEH	017D7	5555H	

OBJECT 3 DESCRIPTOR FIELD

TEXT 50 SCAN LINES/
16 CHARACTERS WIDE

CPU ADDR	DRAM WORD ADDR	CONTENTS	COMMENTS
0A18	0050CH	1010 1 10 0 0 0 0 0 0 0 100	
		Y: Slice No. = 1010	Start Slice Number
		C/B = 1	Character Object
		R1R0 = 10	8 Pixels/Character
		CRS = 0	Character Generator 0
		PSE = 0	Monospace Characters
		FAD = 0	1 Byte/Character
		OBL = 0	No blinking
		BLA = 0	Object is not turned off
		HCR = 0	Don't care
		TDE = 1	Transparent pixels
		CICO = 00	Default color bits
0A1A	0050DH	000010 0001010000	
		WIDTH = 000010	2 four word wide = 16 characters wide
		X = 0001010000	Object starts at X = 80
0A1C	0050EH	0001 0111 1101 1010	Object Base Address 017DAH
0A1FH	0050FH	0001 0111 1101 1010	Current Object Entry 017DAH

OBJECT 3 DATA

CPU ADDR	DRAM ADDR	ASCII CODE	
2FB4	017DA	5620	Line 1
...	017DB	4449	
...	017DC	4F45	
...	017DD	5320	
...	017DE	4F54	
...	017DF	4152	
...	017EO	4547	
...	017E1	2020	
2FC4	017E2	2020	Line 2
...	017E3	2020	
...	...	2020	
...	...	4E41	
...	...	2044	
...	...	2020	
...	...	2020	
...	...	2020	
2FD4	017EA	4420	Line 3
...	...	5349	
...	...	4C50	
...	...	5941	
...	...	4420	
...	...	5645	
...	...	4359	
...	...	2045	
2FE4	017F2	2020	Line 4
...	...	2020	
...	...	2020	
...	...	2020	
...	...	3238	
...	...	3137	
...	...	2036	
...	...	2020	
2FF4	017FA	2020	Line 5
...	...	2020	
...	...	2020	
...	...	4E49	
...	...	4554	
...	...	204C	
...	...	2020	
...	...	2020	

CHARACTER GENERATOR O

The pixel data for characters are stored in one of 2 character generators in the display RAM. Character generator 0 is used in this design. The base address of character generator 0 as obtained from R10, is 02000H in bank 0. Character height as defined in R1 is 10 scan lines. The character set 0 consists of 10 blocks of 256 words each. Block No. 1 contains the 1st slice of each of 256 characters, block 2 has the 2nd slice of all the 256 characters and so on. In this example, 26 alphabets and 10 numerals are defined. The VSDD addresses character generator as shown in Figure 7. Individual slices are addressed by concatenating the four bits of the character generator base address with the slice number Z and the ASCII code itself. Slice number 0 is the bottom scan line for the character and slice number H-1 is the top line. Each slice is encoded as a sequence of pixel bits. If a pixel bit is 1, then the pixel is given the foreground color. If the bit is 0, the pixel is displayed in background color. This is shown in Figure 8.

Figure 9 shows how the pixels are encoded for character "F" (ASCII code = 46H).

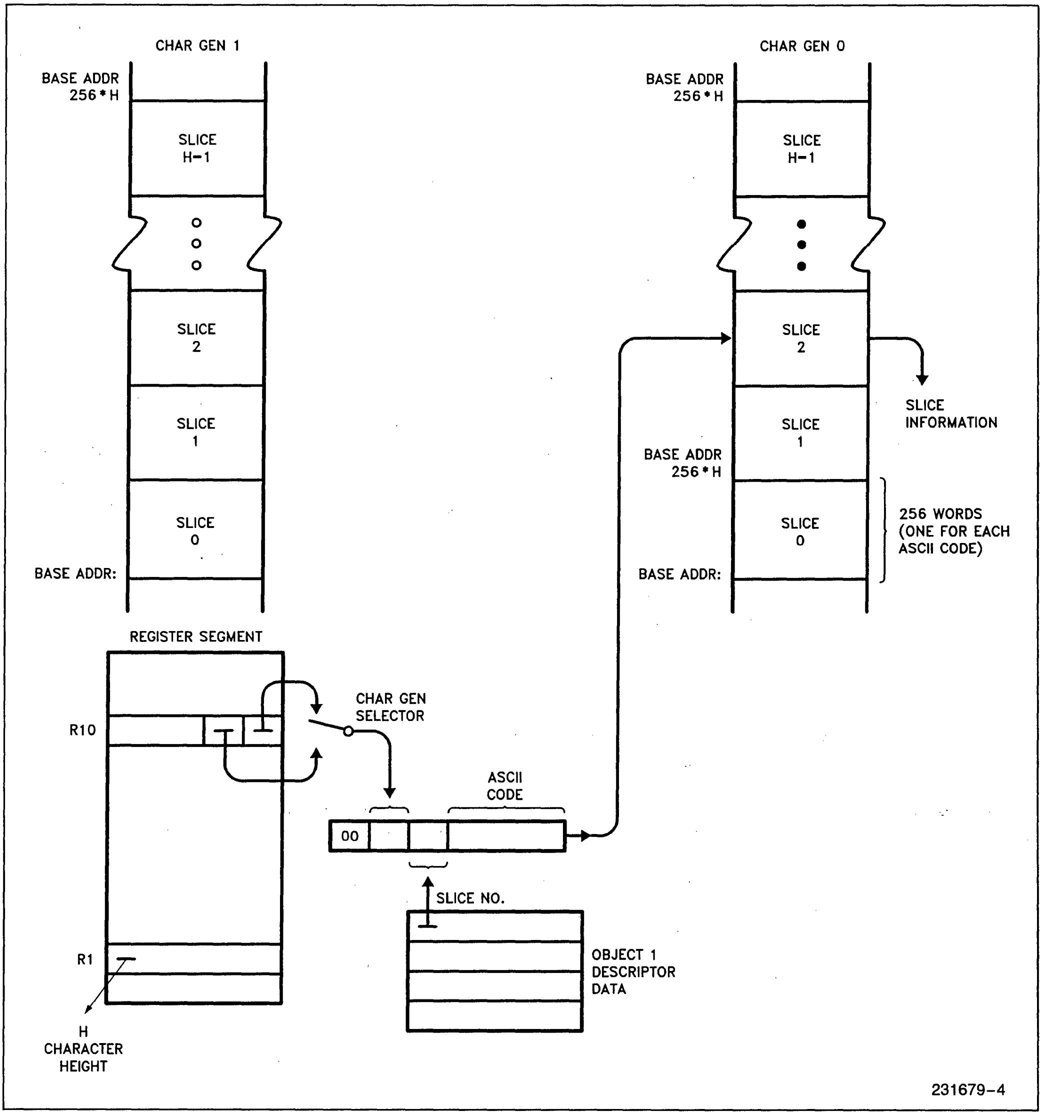
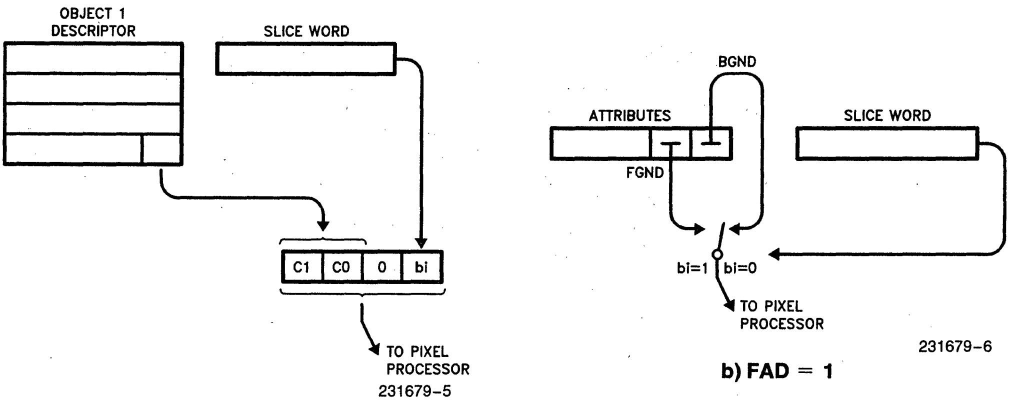


Figure 7. Addressing Character Slices



b) FAD = 1

a) FAD = 0

C1 C0 0 0—BGND Color
C1 C0 0 1—FGND Color

Figure 8. Character Generator Pixel Construction

	DRAM ADDRESS		PIXEL BITS								CPU ADDRESS	HEX DUMP
			P7	P6	P5	P4	P3	P2	P1	P0		
SLICE 9	02946H	00000000	0	0	0	0	0	0	0	0	0528CH	0000H
	02846H	00000000	0	1	1	1	1	1	1	0	0508CH	007EH
	02746H	00000000	0	0	0	0	0	1	1	0	04E8CH	0006H
	02646H	00000000	0	0	0	0	0	1	1	0	04C8CH	0006H
	02546H	00000000	0	0	0	1	1	1	1	0	04A8CH	001EH
	02446H	00000000	0	0	0	0	0	1	1	0	0488CH	0006H
	02346H	00000000	0	0	0	0	0	1	1	0	0468CH	0006H
	02246H	00000000	0	0	0	0	0	1	1	0	0448CH	0006H
	02146H	00000000	0	0	0	0	0	0	0	0	0428CH	0000H
SLICE 0	02046H	00000000	0	0	0	0	0	0	0	0	0408CH	0000H

Character Width = 8 Pixels
Character Height = 10 Scan Lines

Character Generator Base Address = 02000H

NOTE:
The leftmost pixel corresponds to the LSB in the slice word.

Figure 9. Bit Storage for Character "F"

The 80186 used in this design is resident on a single board computer known as SDV-186 board. This board is available from Red River Technology, Inc. in Addison, Texas. The VSDD board is connected to the 186 board via expansion connectors. The memory maps for the SDV-186 board and the VSDD are shown on the following pages. The CPU address space from 60000H–6FFFFH is used for 64K data window. This data window maps onto the data segment in the VSDD's memory space.

VSDD MEMORY MAP

CPU ADDR		VSDD DRAM ADDR
60000		00000
...	R0	...
...		...
...		...
...	CONTROL REGISTERS	...
...		...
...		...
6001E	R15	0000FH
60020	SCAN LINE 0 FLAGS	...
...		...
...		...
...	ACCESS TABLE	...
...		...
...		...
601B0H	SCAN LINE 200 FLAGS	...
6022A		...
...	BLANK	...
602FE		...
60300	COLOR 0000	00180H
...		...
...	COLOR LOOKUP TABLE	...
...		...
6031E	COLOR 1111	...
60320		...
...	BLANK	...
603FE		00500H
60A00		...
...	OBJECT DESCRIPTOR TABLE	...
...		...
60A18		...
...		...
...	BLANK	...
61FFE		01000H
62000		...
...		...
...	OBJECT DATA	...
...		...
63FFE		02000H
64000		...
...		...
...	CHAR GEN 0	...
...		...
65FFE		03000H
66000		...
...		...
...	CHAR GEN 1	...
...		...
67FFE		...
68000		...
...	BLANK	...
...		...

Memory Map of the SDV-186 Board

00000H-003FFH	INTERRUPT VECTORS	
00400H-0076FH	PROGRAM DATA	
00770H-007FFH	STACK	LCS/
00800H-00FFFH	MEMORY	
01000H-1FFFFH	VACANT	
20000H-3FFFFH	VACANT	
40000H-4FFFFH	64-K MEMORY	MCSO/
50000H-5FFFFH	64-K MEMORY	MCS1/
60000H-6FFFFH	64-K MEMORY (VACANT)	MCS2/
70000H-7FFFFH	64-K MEMORY (VACANT)	MCS3/
80000H-FBFFFH	VACANT	
FC000H-FFFFFH	MONITOR CODE	UCS/

APPENDIX A PROGRAMMING HORIZONTAL AND VERTICAL CONSTANTS FOR IBM MONITOR

The constants will be programmed for a resolution of 400 x 200 at 60 Hz, non-interlaced mode.

60 Hz gives a frame period of 16.67 ms.

For IBM Color Monitor, vertical blanking = 3 ms.

Active Vertical zone = 16.67 − 3 = 13.67 ms.

There are 200 lines in one frame.

$$\text{Line Time} = \frac{13.67}{200} = 68.35\ \mu s$$

$$\text{Horizontal Blanking} = 16\ \mu s$$

$$\text{Active Horizontal zone} = 68.35 - 16$$

$$= 52.35\ \mu s$$

$$\text{Horizontal Sync Frequency} = \frac{1}{\text{Line Time}}$$

$$= \frac{1}{68.35}$$

$$= 14.6\ \text{kHz}$$

$$\text{Pixel CLOCK PERIOD} = \frac{\text{Active horizontal time}}{\text{no. of pixels/scan line}}$$

$$= \frac{52.35\ \mu s}{400} = 130.1\ \text{ns}$$

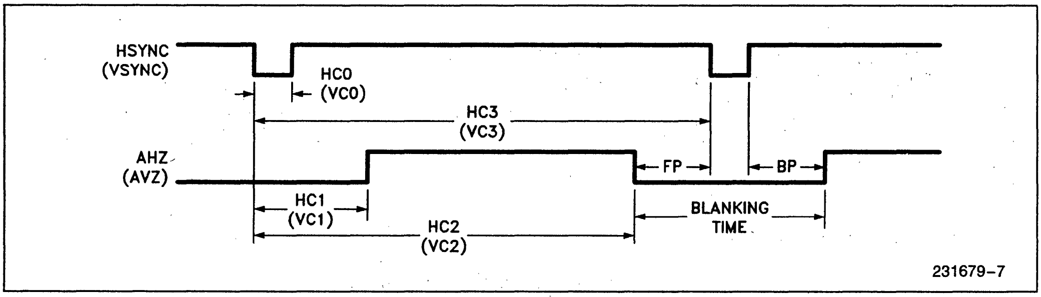
$$\text{PIXEL CLOCK} = \frac{1}{130.1} = 7.7\ \text{MHz}$$

We will use a pixel CLK of 8 mHz. As EVC = 0, System CLK is also 8 MHz.

Horizontal Blanking for the monitor = 16 μs.

This blanking time includes the front porch, sync width and the back porch.

RASTER TIMINGS



231679–7

HORIZONTAL CONSTANTS

For IBM color monitor, Hsync width = 4 μs

Horizontal blanking = 16 μs

Assume:

FP = 6 μs

BP = 6 μs

HCO = 4 μs

HC1 = 4 + 6 = 10 μs

HC2 = Active Horizontal Time + HC1

= 52.3 + 10 = 62.3 μs

HC3 = Line Time = 68.3 μs

For 8 MHz Video Clock, the period is 125 ns.

NOW, PSA = 0

GCLK PERIOD = 16 × 125 = 2000 ns
= 2 μs

HCO – HC3 are programmed in terms of GCLK

HCO = 2 GCLK PERIODS = 4 μs

HCO = 000001B

NOTE:
HC0–HC3 and VC0–VC3 values are offset by 1, i.e. if HC0 is 2, then time programmed is 3 GCLK periods. HC2 and HC3 had to be tweeked to obtain a steady display on the screen. The following values give a flicker-free display.

HC1 = 10 μs = 5 GCLK Periods

HC1 = 000100B

HC2 = 60 μs = 30 GCLK Periods

HC2 = 011101B

HC3 = 66 μs = 33 GCLK Periods

HC3 = 100000B

VERTICAL CONSTANTS

Vertical Blanking = 3 ms

Line Time = 33 GCLK Periods

= 66 μs

For IBM Monitor

VC0 = VSYNC Width = 198 μs

= 3 Line times

VC0 = 0000000010B

Assume, FP = 0.5 ms

BP = 2.3 ms

VC1 = VC0 + BP

= 0.198 + 2.3 = 2.5 ms = 37 Line times

VC1 = 0000100100B

VC2 = Active vertical time + VC1

= 13.67 + 2.5 = 16.17 ms = 237 Lines time

VC2 = 0011101100

VC3 = Vertical sweep rate = 16.67 ms = 245 Line times

VC3 = 0011110100

APPENDIX B CHARACTER GENERATOR 0

This character set is located in the VSDD's DRAM in bank O. It starts at 02000H. 26 alphabets, 10 numerals and a blank space are defined here. The ASCII code for the characters are as follows:

CHARACTER	ASCII	NUMERALS	ASCII CODE
A	41H	0	30H
B	42H	1	31H
C	43H	2	32H
D	44H	3	33H
E	45H	4	34H
F	46H	5	35H
G	47H	6	36H
H	48H	7	37H
I	49H	8	38H
J	4AH	9	39H
K	4BH		
L	4CH		
M	4DH		
N	4EH		
O	4FH		
P	50H		
Q	51H		
R	52H		
S	53H		
T	54H		
U	55H		
V	56H		
W	57H		
X	58H		
Y	59H		
Z	5AH		
BLANK SPACE	20H		

The character set has 10 blocks of 256 words each. All the locations except those corresponding to above 37 characters are blank in the display RAM. They are programmed with 0's. The hex dump for the characters is as follows:

SLICE 0		SLICE 1		SLICE 2		
DRAM LOCATION	HEX DUMP	DRAM LOCATION	HEX DUMP	DRAM LOCATION	HEX DUMP	CHARACTER
2020H	00000H	2120H	00000H	2220H	00000H	BLANK SPACE
2030	0000	2130	0000	2230	003C	0
2031	0000	2131	0000	2231	007E	1
2032	0000	2132	0000	2232	007E	2
2033	0000	2133	0000	2233	003C	3
2034	0000	2134	0000	2234	0060	4
2035	0000	2135	0000	2235	003C	5
2036	0000	2136	0000	2236	003C	6
2037	0000	2137	0000	2237	0018	7
2038	0000	2138	0000	2238	003C	8
2039	0000	2139	0000	2239	003C	9
2041	0000	2141	0000	2241	0066	A
2042	0000	2142	0000	2242	003E	B
2043	0000	2143	0000	2243	0038	C
2044	0000	2144	0000	2244	001E	D
2045	0000	2145	0000	2245	007E	E
2046	0000	2146	0000	2246	0006	F
2047	0000	2147	0000	2247	0038	G
2048	0000	2148	0000	2248	0066	H
2049	0000	2149	0000	2249	003C	I
204A	0000	214A	0000	224A	003C	J
204B	0000	214B	0000	224B	0066	K
204C	0000	214C	0000	224C	007E	L
204D	0000	214D	0000	224D	0066	M
204E	0000	214E	0000	224E	0046	N
204F	0000	214F	0000	224F	003C	O
2050	0000	2150	0000	2250	000C	P
2051	0000	2151	0000	2251	007C	Q
2052	0000	2152	0000	2252	0066	R
2053	0000	2153	0000	2253	003C	S
2054	0000	2154	0000	2254	0018	T
2055	0000	2155	0000	2255	003C	U
2056	0000	2156	0000	2256	0018	V
2057	0000	2157	0000	2257	003C	W
2058	0000	2158	0000	2258	0066	X
2059	0000	2159	0000	2259	0018	Y
205A	0000	215A	0000	225A	007E	Z

SLICE 3		SLICE 4		SLICE 5		
DRAM LOCATION	HEX DUMP	DRAM LOCATION	HEX DUMP	DRAM LOCATION	HEX DUMP	CHARACTER
2320H	0000H	2420H	0000H	2520H	0000H	SPACE
2330	0066	2430	0066	2530	006E	0
2331	0018	2431	0018	2531	0018	1
2332	0006	2432	000C	2532	0030	2
2333	0066	2433	0060	2533	0038	3
2334	0060	2434	007E	2534	0066	4
2335	0066	2435	0060	2535	0060	5
2336	0066	2436	0066	2536	003E	6
2337	0018	2437	0018	2537	0018	7
2338	0066	2438	0066	2538	003C	8
2339	0066	2439	0060	2539	007C	9
2341	0066	2441	007E	2541	0066	A
2342	0066	2442	0066	2542	003E	B
2343	006C	2443	0006	2543	0006	C
2344	0036	2444	0066	2544	0066	D
2345	0006	2445	0006	2545	001E	E
2346	0006	2446	0006	2546	001E	F
2347	006C	2447	0046	2547	0066	G
2348	0066	2448	0066	2548	007E	H
2349	0018	2449	0018	2549	0018	I
234A	0066	244A	0060	254A	0060	J
234B	0036	244B	000E	254B	0006	K
234C	007E	244C	0006	254C	0006	L
234D	0066	244D	0066	254D	0066	M
234E	0066	244E	0076	254E	007E	N
234F	0066	244F	0066	254F	0066	O
2350	0006	2450	0006	2550	003E	P
2351	0026	2451	0036	2551	0066	Q
2352	0036	2452	001E	2552	003E	R
2353	0066	2453	0060	2553	003C	S
2354	0018	2454	0018	2554	0018	T
2355	0066	2455	0066	2555	0066	U
2356	0018	2456	003E	2556	0024	V
2357	007E	2457	005A	2557	005A	W
2358	0066	2458	003C	2558	0018	X
2359	0018	2459	0018	2559	003C	Y
235A	0006	245A	000C	255A	0018	Z

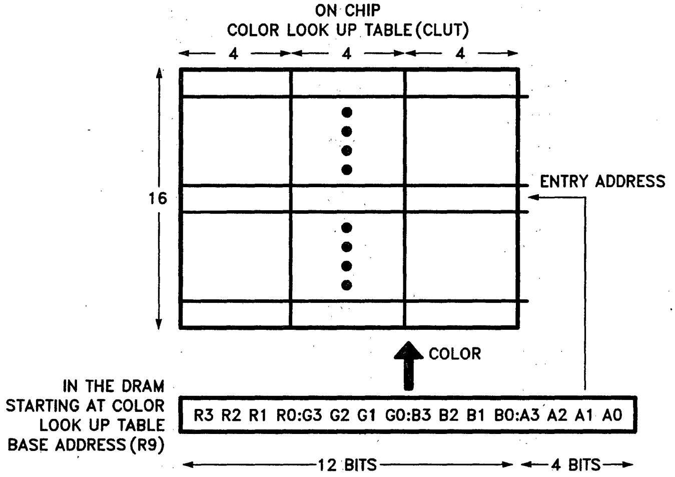
SLICE 6		SLICE 7		SLICE 8		
DRAM LOCATION	HEX DUMP	DRAM LOCATION	HEX DUMP	DRAM LOCATION	HEX DUMP	CHARACTER
2620H	0000H	2720H	0000H	2820H	0000H	SPACE
2630	0076	2730	0066	2830	003C	0
2631	001C	2731	0018	2831	0018	1
2632	0060	2732	0066	2832	003C	2
2633	0060	2733	0066	2833	003C	3
2634	0078	2734	0070	2834	0060	4
2635	003E	2735	0002	2835	007E	5
2636	0006	2736	0066	2836	003C	6
2637	0030	2737	0066	2837	007E	7
2638	0066	2738	0066	2838	003C	8
2639	0066	2739	0066	2839	003C	9
2641	003C	2741	003C	2841	0018	A
2642	0066	2742	0066	2842	003E	B
2643	0006	2743	006C	2843	0038	C
2644	0066	2744	0036	2844	001E	D
2645	0006	2745	0006	2845	007E	E
2646	0006	2746	0006	2846	007E	F
2647	0006	2747	006C	2847	0038	G
2648	0066	2748	0066	2848	0066	H
2649	0018	2749	0018	2849	003C	I
264A	0060	274A	0060	284A	0070	J
264B	000E	274B	0036	284B	0066	K
264C	0006	274C	0006	284C	0006	L
264D	007E	274D	0066	284D	0024	M
264E	006E	274E	0066	284E	0062	N
264F	0066	274F	0066	284F	003C	O
2650	0066	2750	0066	2850	003E	P
2651	0066	2751	0066	2851	003C	Q
2652	0066	2752	0066	2852	003E	R
2653	0006	2753	0066	2853	003C	S
2654	0018	2754	0018	2854	007E	T
2655	0066	2755	0066	2855	0066	U
2656	0066	2756	0042	2856	0042	V
2657	0042	2757	0042	2857	0042	W
2658	003C	2758	0066	2858	0066	X
2659	003C	2759	0066	2859	0066	Y
265A	0030	275A	0060	285A	007E	Z

SLICE 9		
DRAM LOCATION	HEX DUMP	CHARACTER
2920H	0000H	SPACE
2930	0000	0
2931	0000	1
2932	0000	2
2933	0000	3
2934	0000	4
2935	0000	5
2936	0000	6
2937	0000	7
2938	0000	8
2939	0000	9
2941	0000	A
2942	0000	B
2943	0000	C
2944	0000	D
2945	0000	E
2946	0000	F
2947	0000	G
2948	0000	H
2949	0000	I
294A	0000	J
294B	0000	K
294C	0000	L
294D	0000	M
294E	0000	N
294F	0000	O
2950	0000	P
2951	0000	Q
2952	0000	R
2953	0000	S
2954	0000	T
2955	0000	U
2956	0000	V
2957	0000	W
2958	0000	X
2959	0000	Y
295A	0000	Z

APPENDIX C ANALOG OPERATION

In the example described in the application note, digital video outputs were used.

For analog operation, DEI (digitally encoded color information) bit in register R0 is set to 0. The on-chip color look table (CLUT) is loaded with 16 entries from the external DRAM. These 16 words of data are stored in the memory starting at color look up table base address. (Register R9). Each entry is 16 bits long—with lowest 4 bits specifying the address of the entry in the CLUT and upper 12 bits specifying the color as shown in Figure 10. Four bit pixel code is used to address the CLUT. The pixel code is matched with the lowest four bits of the CLUT RAM and the pixel is given the color specified by the upper 12 bits. The address entries need not be sequential from 0–15 but they can be random. The color corresponding to address 0010 in the CLUT is reserved for the background color.

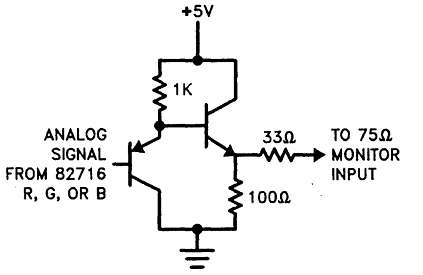

ON CHIP
COLOR LOOK UP TABLE (CLUT)
4
4
4
16
ENTRY ADDRESS
COLOR
IN THE DRAM STARTING AT COLOR LOOK UP TABLE BASE ADDRESS (R9)
R3 R2 R1 R0:G3 G2 G1 G0:B3 B2 B1 B0:A3 A2 A1 A0
12 BITS
4 BITS

231679–8

R3–R0: Red color component
G3–G0: Green color component
B3–B0: Blue color component
A3–A0: Entry address

Figure 10. Filling the CLUT

The color look up table outputs—4 bits/color—drive 3 internal DACs (digital-to-analog converter). RGB signals are generated by the DACs. An externally supplied reference voltage (VREF) is used to drive the DACs. The value of VREF should be between 0 and 2V. As DACs have high output resistance, external analog buffers are used to interface them to low input resistance monitors as shown in Figure 11.



231679-9

Figure 11. Buffering 82716 Analog Output to Low Input Resistance Monitor

APPENDIX D
CLOCKING SCHEMES

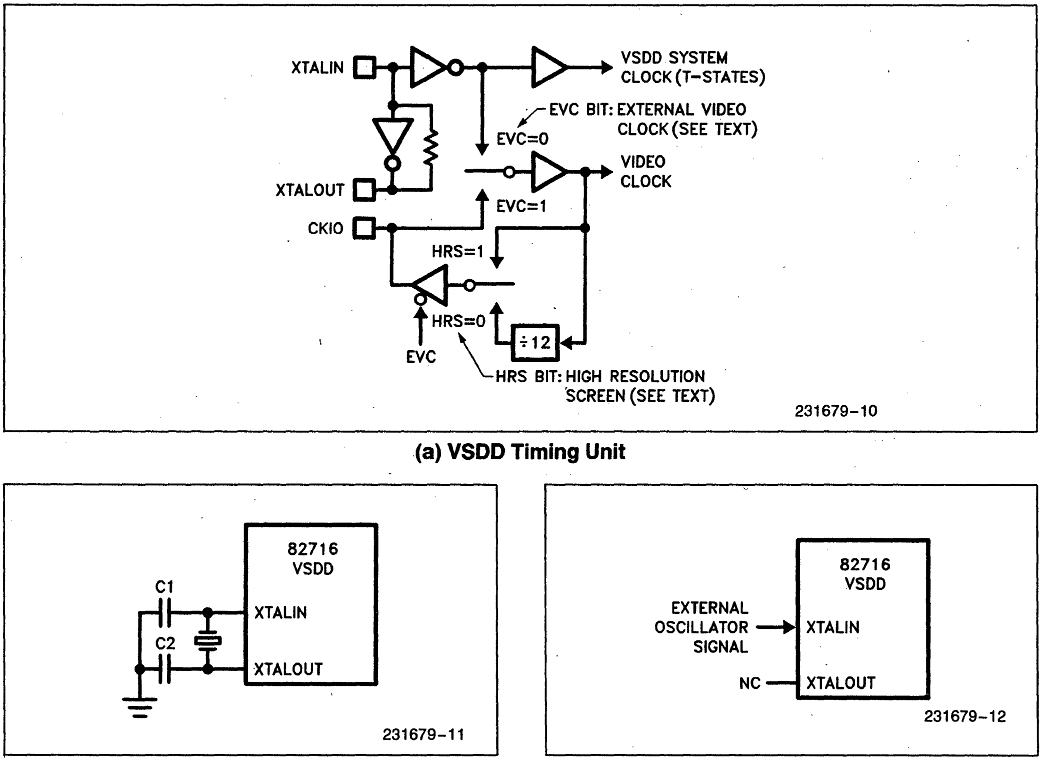
The VSDD uses two clock signals: system clock and video clock. The video clock can be derived from the system clock or may be external.

The system clock is generated by an internal oscillator using an external crystal at XTALIN and XTALOUT pins. The crystal frequency can be between 5 MHz and 15 MHz.

An external clock may also be fed to the XTALIN pin (instead of using a crystal). For example, CLKOUT pin of 80186 can be used to drive the VSDD system clock.

If an independent video (dot) clock is desired (EVC = 1) CKIO is used to input this clock. Maximum video clock frequency can be 25 MHz. EVC should be set to zero if video clock is derived from system clock.

Figure 12 explains various clocking schemes.


XTALIN
XTALOUT
CKIO
VSDD SYSTEM CLOCK (T-STATES)
EVC BIT: EXTERNAL VIDEO CLOCK (SEE TEXT)
EVC=0
VIDEO CLOCK
EVC=1
HRS=1
HRS=0
÷12
EVC
HRS BIT: HIGH RESOLUTION SCREEN (SEE TEXT)
231679-10
82716 VSDD
C1
C2
XTALIN
XTALOUT
231679-11
82716 VSDD
EXTERNAL OSCILLATOR SIGNAL
XTALIN
NC
XTALOUT
231679-12

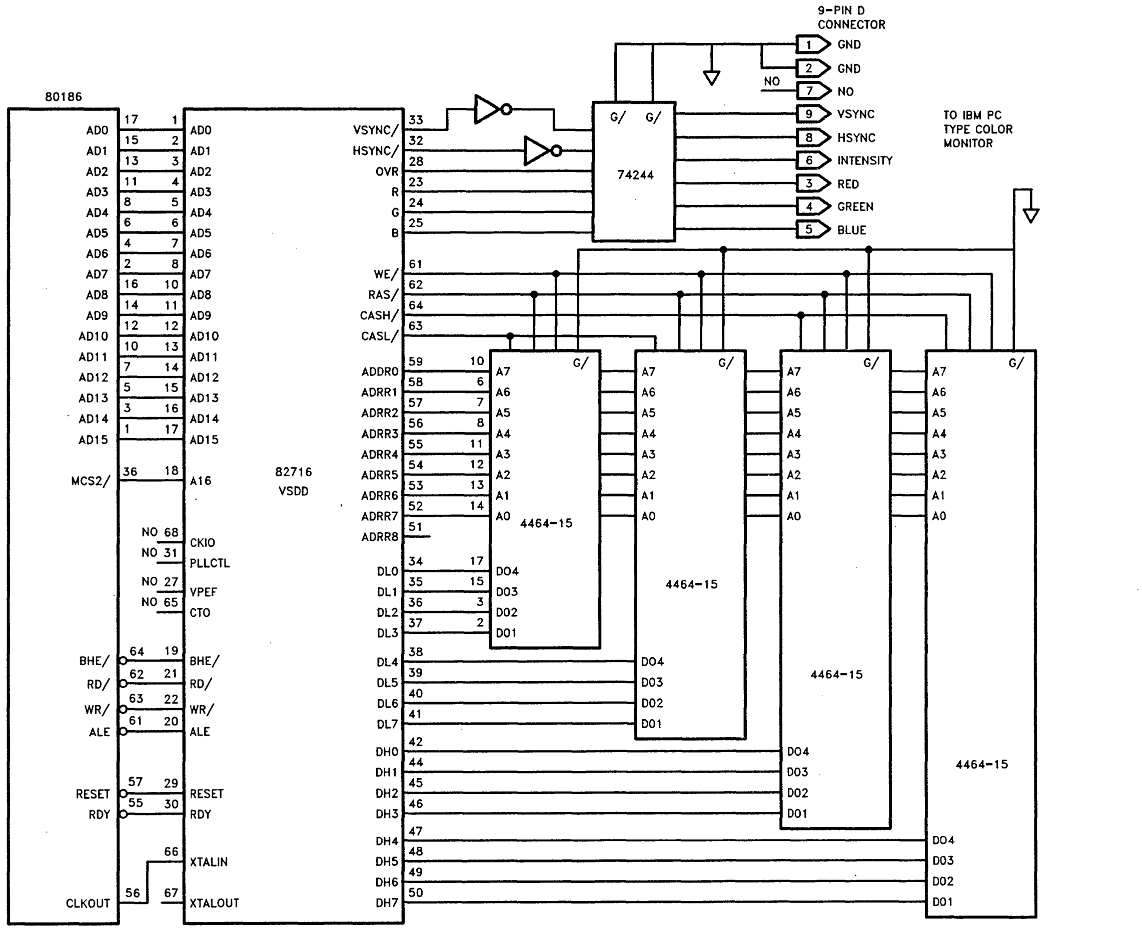
(a) VSDD Timing Unit

(b) With Crystal

(c) With External Clock

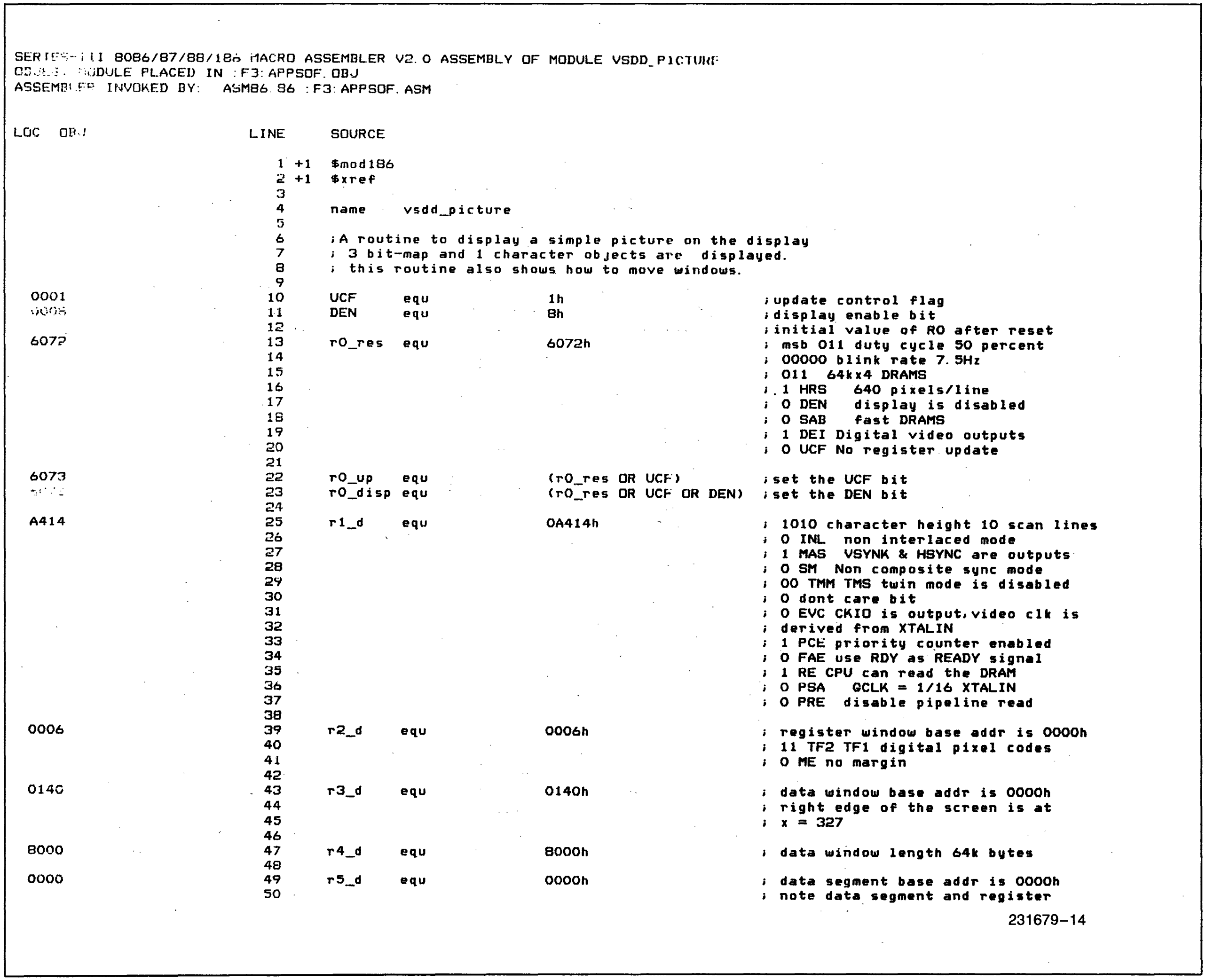
Figure 12. Clocking Schemes

APPENDIX E



231679-13

APPENDIX F

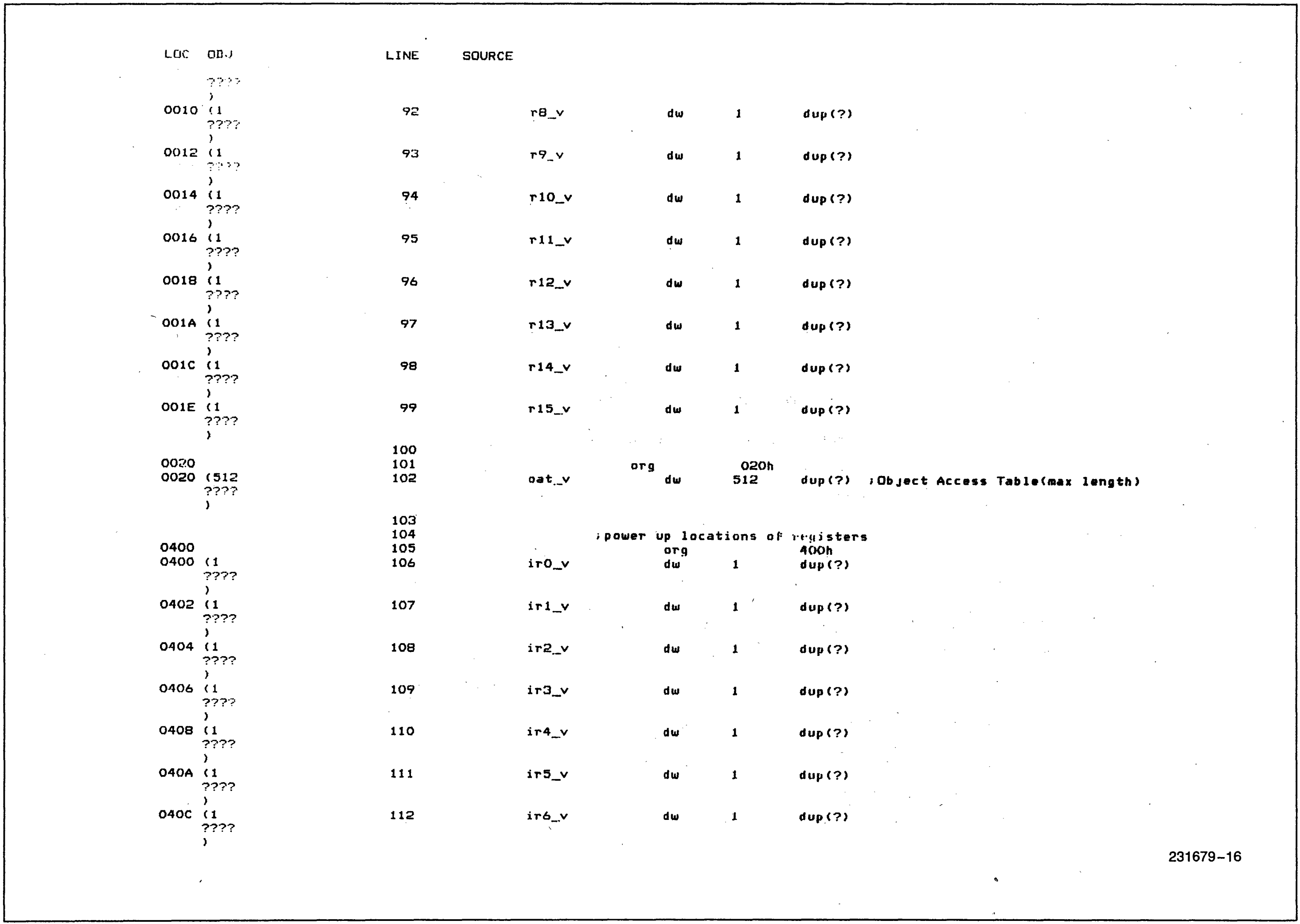


```
SERIES-III 8086/87/88/186 MACRO ASSEMBLER V2.0 ASSEMBLY OF MODULE VSDD_PICTURE
OBJECT MODULE PLACED IN :F3:APPSOF.OBJ
ASSEMBLER INVOKED BY:   ASM86.86 :F3:APPSOF.ASM

LOC  OBJ                  LINE     SOURCE

                             1 +1  $mod186
                             2 +1  $xref
                             3
                             4     name    vsdd_picture
                             5
                             6     ;A routine to display a simple picture on the display
                             7     ; 3 bit-map and 1 character objects are  displayed.
                             8     ; this routine also shows how to move windows.
                             9
 0001                       10     UCF      equ          1h                         ;update control flag
 0008                       11     DEN      equ          8h                         ;display enable bit
                            12                                                      ;initial value of RO after reset
 6072                       13     r0_res   equ          6072h                      ; msb 011 duty cycle 50 percent
                            14                                                      ; 00000 blink rate 7.5Hz
                            15                                                      ; 011  64kx4 DRAMS
                            16                                                      ;.1 HRS   640 pixels/line
                            17                                                      ; 0 DEN   display is disabled
                            18                                                      ; 0 SAB   fast DRAMS
                            19                                                      ; 1 DEI Digital video outputs
                            20                                                      ; 0 UCF No register update
                            21
 6073                       22     r0_up    equ          (r0_res OR UCF)            ;set the UCF bit
 607B                       23     r0_disp  equ          (r0_res OR UCF OR DEN)     ;set the DEN bit
                            24
 A414                       25     r1_d     equ          0A414h                     ; 1010 character height 10 scan lines
                            26                                                      ; 0 INL  non interlaced mode
                            27                                                      ; 1 MAS  VSYNK & HSYNC are outputs
                            28                                                      ; 0 SM  Non composite sync mode
                            29                                                      ; 00 TMM TMS twin mode is disabled
                            30                                                      ; 0 dont care bit
                            31                                                      ; 0 EVC CKIO is output,video clk is
                            32                                                      ; derived from XTALIN
                            33                                                      ; 1 PCE priority counter enabled
                            34                                                      ; 0 FAE use RDY as READY signal
                            35                                                      ; 1 RE CPU can read the DRAM
                            36                                                      ; 0 PSA   GCLK = 1/16 XTALIN
                            37                                                      ; 0 PRE  disable pipeline read
                            38
 0006                       39     r2_d     equ          0006h                      ; register window base addr is 0000h
                            40                                                      ; 11 TF2 TF1 digital pixel codes
                            41                                                      ; 0 ME no margin
                            42
 0140                       43     r3_d     equ          0140h                      ; data window base addr is 0000h
                            44                                                      ; right edge of the screen is at
                            45                                                      ; x = 327
                            46
 8000                       47     r4_d     equ          8000h                      ; data window length 64k bytes
                            48
 0000                       49     r5_d     equ          0000h                      ; data segment base addr is 0000h
                            50                                                      ; note data segment and register
```

231679-14

```
LOC  OBJ                 LINE     SOURCE

                          51                                                   ; segment overlap. In the overlapped
                          52                                                   ; region register segment will
                          53                                                   ; overwrite  the data segment
                          54                                                  ; 00 BS1 BS0 bank 0
                          55
  000A                    56     r6_d      equ               000Ah             ; CPU is allowed 10 accesses during each
                          57                                                   ; line building process.
                          58
  0500                    59     r7_d      equ               0500h             ; object descriptor table starts at
                          60                                                   ; 0500h in bank 0
                          61
  0010                    62     r8_d      equ               0010h             ; access table at 0010h
                          63
  0180                    64     r9_d      equ               0180h             ; color look up table base addr
                          65
  0023                    66     r10_d     equ               0023h             ; char gen0 at 2000h
                          67                                                   ; char gen1 at 3000h
                          68
  0010                    69     r11_d     equ               0010h
                          70
  0402                    71     r12_d     equ               0402h             ; HC0, VC0
                          72
  1024                    73     r13_d     equ               1024h             ; HC1, VC1
                          74
  74EC                    75     r14_d     equ               74ECh             ; HC2, VC2
                          76
  80F4                    77     r15_d     equ               80F4h             ; HC3, VC3
                          78
                          79     ; allocate memory for register and data segments
                          80
                          81
----                      82     video_vsdd      segment          at 6000H
                          83
0000 (1                   84               r0_v           dw        1       dup(?)
     ????
     )
0002 (1                   85               r1_v           dw        1       dup(?)
     ????
     )
0004 (1                   86               r2_v           dw        1       dup(?)
     ????
     )
0006 (1                   87               r3_v           dw        1       dup(?)
     ????
     )
0008 (1                   88               r4_v           dw        1       dup(?)
     ????
     )
000A (1                   89               r5_v           dw        1       dup(?)
     ????
     )
000C (1                   90               r6_v           dw        1       dup(?)
     ????
     )
000E (1                   91               r7_v           dw        1       dup(?)
```

231679-15

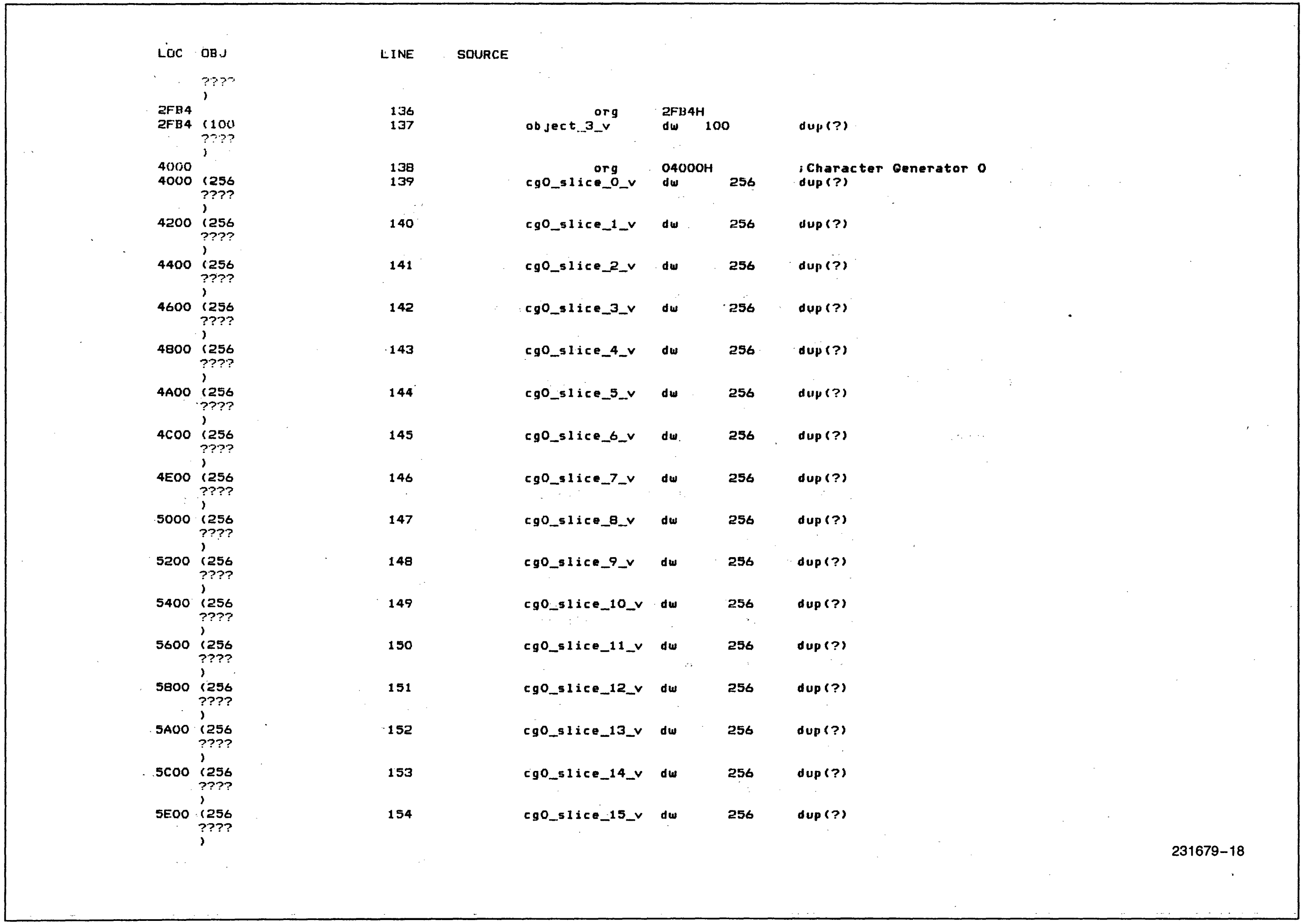


```
LOC  OBJ                   LINE     SOURCE

     ????
     )
0010 (1                      92            r8_v          dw       1        dup(?)
     ????
     )
0012 (1                      93            r9_v          dw       1        dup(?)
     ????
     )
0014 (1                      94            r10_v         dw       1        dup(?)
     ????
     )
0016 (1                      95            r11_v         dw       1        dup(?)
     ????
     )
0018 (1                      96            r12_v         dw       1        dup(?)
     ????
     )
001A (1                      97            r13_v         dw       1        dup(?)
     ????
     )
001C (1                      98            r14_v         dw       1        dup(?)
     ????
     )
001E (1                      99            r15_v         dw       1        dup(?)
     ????
     )
                            100
0020                        101                       org       020h
0020 (512                   102            oat_v         dw       512      dup(?)  ;Object Access Table(max length)
     ????
     )
                            103
                            104                  ;power up locations of registers
0400                        105                          org               400h
0400 (1                     106            ir0_v         dw       1        dup(?)
     ????
     )
0402 (1                     107            ir1_v         dw       1        dup(?)
     ????
     )
0404 (1                     108            ir2_v         dw       1        dup(?)
     ????
     )
0406 (1                     109            ir3_v         dw       1        dup(?)
     ????
     )
0408 (1                     110            ir4_v         dw       1        dup(?)
     ????
     )
040A (1                     111            ir5_v         dw       1        dup(?)
     ????
     )
040C (1                     112            ir6_v         dw       1        dup(?)
     ????
     )
```

231679-16

```
LOC  OBJ              LINE    SOURCE

040E (1                113              ir7_v            dw     1       dup(?)
     ????
     )
0410 (1                114              ir8_v            dw     1       dup(?)
     ????
     )
0412 (1                115              ir9_v            dw     1       dup(?)
     ????
     )
0414 (1                116              ir10_v           dw     1       dup(?)
     ????
     )
0416 (1                117              ir11_v           dw     1       dup(?)
     ????
     )
0418 (1                118              ir12_v           dw     1       dup(?)
     ????
     )
041A (1                119              ir13_v           dw     1       dup(?)
     ????
     )
041C (1                120              ir14_v           dw     1       dup(?)
     ????
     )
041E (1                121              ir15_v           dw     1       dup(?)
     ????
     )
0300                   122                      org      300H
0300 (16               123              clut_v           dw     16      dup(?)  ;colour lookup table
     ????
     )
0A00                   124                      org      0A00H
0A00 (4                125              odt0_v           dw     4       dup(?)  ;Object descriptor table
     ????
     )
0A08 (4                126              odt1_v           dw     4       dup(?)
     ????
     )
0A10 (4                127              odt2_v           dw     4       dup(?)
     ????
     )
0A18 (4                128              odt3_v           dw     4       dup(?)
     ????
     )
                       129
2000                   130                      org      2000H                  ;3.6K bytes
2000 (1800             131              object_0_v       dw     1800    dup(?)  ;4 bits/pixel 75*96 bit mapped
     ????
     )
2E14                   132                      org      2E14H
2E14 (80               133              object_1_v       dw       80    dup(?)
     ????
     )
2EC0                   134                      org      2EC0H
2EC0 (120              135              object_2_v       dw      120    dup(?)
```

231679-17



```
LOC   OBJ                  LINE     SOURCE

      ????
      )
2FB4                        136                 org      2FB4H
2FB4  (100                  137           object_3_v     dw    100         dup(?)
      ????
      )
4000                        138                 org      04000H            ;Character Generator 0
4000  (256                  139           cg0_slice_0_v  dw        256     dup(?)
      ????
      )
4200  (256                  140           cg0_slice_1_v  dw        256     dup(?)
      ????
      )
4400  (256                  141           cg0_slice_2_v  dw        256     dup(?)
      ????
      )
4600  (256                  142           cg0_slice_3_v  dw        256     dup(?)
      ????
      )
4800  (256                  143           cg0_slice_4_v  dw        256     dup(?)
      ????
      )
4A00  (256                  144           cg0_slice_5_v  dw        256     dup(?)
      ????
      )
4C00  (256                  145           cg0_slice_6_v  dw        256     dup(?)
      ????
      )
4E00  (256                  146           cg0_slice_7_v  dw        256     dup(?)
      ????
      )
5000  (256                  147           cg0_slice_8_v  dw        256     dup(?)
      ????
      )
5200  (256                  148           cg0_slice_9_v  dw        256     dup(?)
      ????
      )
5400  (256                  149           cg0_slice_10_v dw        256     dup(?)
      ????
      )
5600  (256                  150           cg0_slice_11_v dw        256     dup(?)
      ????
      )
5800  (256                  151           cg0_slice_12_v dw        256     dup(?)
      ????
      )
5A00  (256                  152           cg0_slice_13_v dw        256     dup(?)
      ????
      )
5C00  (256                  153           cg0_slice_14_v dw        256     dup(?)
      ????
      )
5E00  (256                  154           cg0_slice_15_v dw        256     dup(?)
      ????
      )
```

231679-18

```
LOC  OBJ              SOURCE

----                  video_vsdd      ends

----                  video_data              segment

0000 20564944454F20         object_3_data                       db      ' VIDEO STORAGE  '
     53544F52414745
     2020

0010 2020202020414E                                             db        '     AND        '
     44202020202020
     2020

0020 20444953504C41                                             db      ' DISPLAY DEVICE '
     59204445564943
     4520

0030 20202020383237                                             db      '    82716       '
     31362020202020
     2020

0040 20202020494E54                                             db      '    INTEL
     454C0920202020
     20

                      ;    Data for character generator 0
                      ;    Characters are 10 scan lines high
                      ;    Slice #0,1 and 9 are empty(0's)
                      ;    26 alphabets and 10 numbers are defined

                      ;    slice information for 10 numbers



004F 3C                     numbers_data                db      3Ch, 7Eh, 7Eh, 3Ch, 60h ;slice 2
0050 7E
0051 7E
0052 3C
0053 60
0054 3C                                                 db      3Ch, 3Ch, 18h, 3Ch, 3Ch
0055 3C
0056 18
0057 3C
0058 3C
0059 66                                                 db      66h, 18h, 06h, 66h, 60h ;slice 3
005A 18
005B 06
005C 66
005D 60
005E 66                                                 db      66h, 66h, 18h, 66h, 66h
005F 66
0060 18
0061 66
0062 66
```

231679-19

```
LOC  OBJ          LINE     SOURCE

0063 66            184                                db      66h, 18h, 0Ch, 60h, 7Eh ;slice 4
0064 18
0065 0C
0066 60
0067 7E
0068 60            185                                db      60h, 66h, 18h, 66h, 60h
0069 66
006A 18
006B 66
006C 60
006D 6E            186                                db      6Eh, 18h, 30h, 38h, 66h ;slice 5
006E 18
006F 30
0070 38
0071 66
0072 60            187                                db      60h, 3Eh, 18h, 3Ch, 7Ch
0073 3E
0074 18
0075 3C
0076 7C
0077 76            188                                db      76h, 1Ch, 60h, 60h, 78h ;slice 6
0078 1C
0079 60
007A 60
007B 78
007C 3E            189                                db      3Eh, 06h, 30h, 66h, 66h
007D 06
007E 30
007F 66
0080 66
0081 66            190                                db      66h, 18h, 66h, 66h, 70h ;slice 7
0082 18
0083 66
0084 66
0085 70
0086 02            191                                db      02h, 66h, 66h, 66h, 66h
0087 66
0088 66
0089 66
008A 66
008B 3C            192                                db      3Ch, 18h, 3Ch, 3Ch, 60h ;slice 8
008C 18
008D 3C
008E 3C
008F 60
0090 7E            193                                db      7Eh, 3Ch, 7Eh, 3Ch, 3Ch
0091 3C
0092 7E
0093 3C
0094 3C
                   194
                   195
                   196     ;       slice information for 26 alphabets
                   197
                   198             ;character_set_0
```

231679-20

```
LOC  OBJ            LINE    SOURCE

                     199                                    ;slices 0, 1 and 9 are 0's (empty)
                     200
0095 66              201            slice_2_d       db      66H, 3EH, 38H, 1EH, 7EH, 06H, 38H, 66H
0096 3E
0097 38
0098 1E
0099 7E
009A 06
009B 38
009C 66
009D 3C              202                            db      3CH, 3CH, 66H, 7EH, 66H, 46H, 3CH, 0CH
009E 3C
009F 66
00A0 7E
00A1 66
00A2 46
00A3 3C
00A4 0C
00A5 7C              203                            db      7CH, 66H, 3CH, 18H, 3CH, 18H, 3CH, 66H
00A6 66
00A7 3C
00A8 18
00A9 3C
00AA 18
00AB 3C
00AC 66
00AD 18              204                            db      18H, 7EH
00AE 7E
00AF 66              205            slice_3_d       db      66H, 66H, 6CH, 36H, 06H, 06H, 6CH, 66H
00B0 66
00B1 6C
00B2 36
00B3 06
00B4 06
00B5 6C
00B6 66
00B7 18              206                            db      18H, 66H, 36H, 7EH, 66H, 66H, 66H, 06H
00B8 66
00B9 36
00BA 7E
00BB 66
00BC 66
00BD 66
00BE 06
00BF 26              207                            db      26H, 36H, 66H, 18H, 66H, 18H, 7EH, 66H
00C0 36
00C1 66
00C2 18
00C3 66
00C4 18
00C5 7E
00C6 66
00C7 18              208                            db      18H, 06H
00C8 06
00C9 7E              209            slice_4_d       db      7EH, 66H, 06H, 66H, 06H, 06H, 46H, 66H
```

231679-21

```
LOC  OBJ          LINE     SOURCE

00CA 66
00CB 06
00CC 66
00CD 06
00CE 06
00CF 46
00D0 66
00D1 18            210                                db      18H, 60H, 0EH, 06H, 66H, 76H, 66H, 06H
00D2 60
00D3 0E
00D4 06
00D5 66
00D6 76
00D7 66
00D8 06
00D9 36            211                                db      36H, 1EH, 60H, 18H, 66H, 3CH, 5AH, 3CH
00DA 1E
00DB 60
00DC 18
00DD 66
00DE 3C
00DF 5A
00E0 3C
00E1 18            212                                db      18H, 0CH
00E2 0C
00E3 66            213          slice_5_d             db      66H, 3EH, 06H, 66H, 1EH, 1EH, 66H, 7EH
00E4 3E
00E5 06
00E6 66
00E7 1E
00E8 1E
00E9 66
00EA 7E
00EB 18            214                                db      18H, 60H, 06H, 06H, 66H, 7EH, 66H, 3EH
00EC 60
00ED 06
00EE 06
00EF 66
00F0 7E
00F1 66
00F2 3E
00F3 66            215                                db      66H, 3EH, 3CH, 18H, 66H, 24H, 5AH, 18H
00F4 3E
00F5 3C
00F6 18
00F7 66
00F8 24
00F9 5A
00FA 18
00FB 3C            216                                db      3CH, 18H
00FC 18
00FD 3C            217          slice_6_d             db      3CH, 66H, 06H, 66H, 06H, 06H, 06H, 66H
00FE 66
00FF 06
0100 66
```

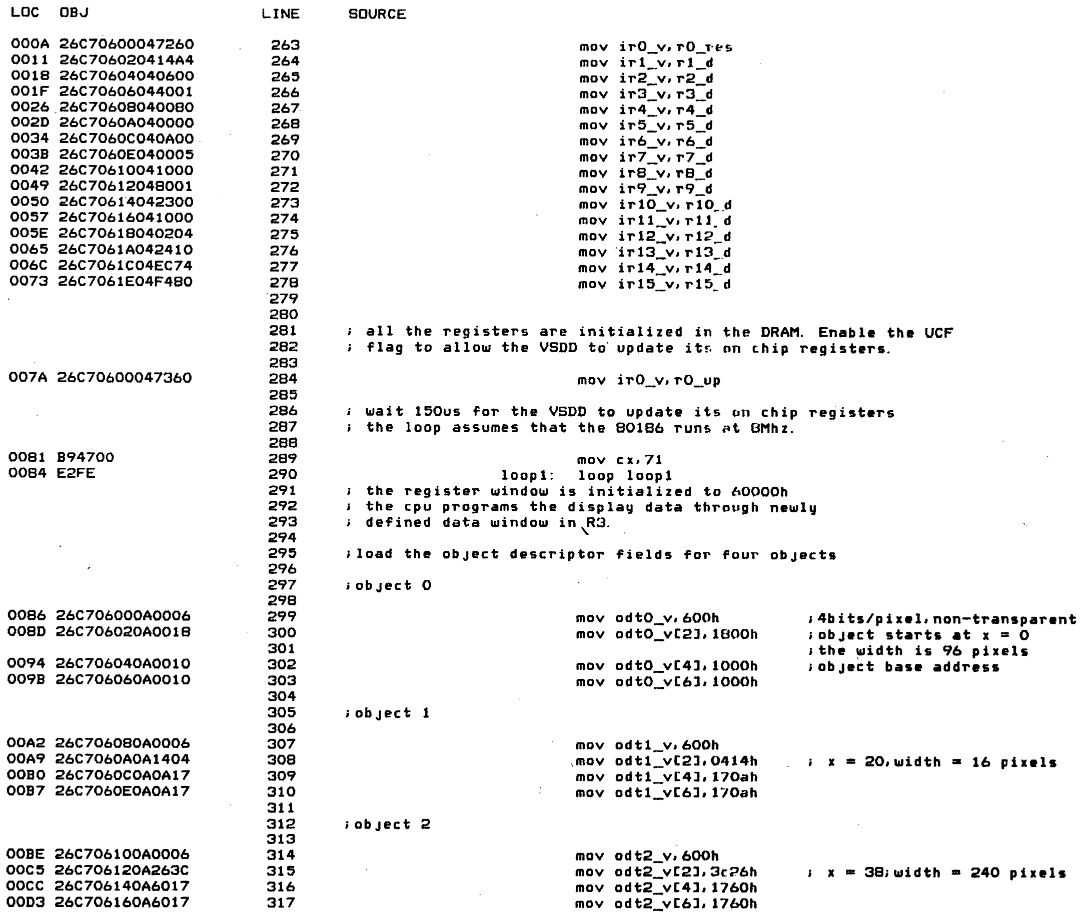
231679–22

```
LOC  OBJ          LINE     SOURCE

0101 06
0102 06
0103 06
0104 66
0105 18           218                                db      18H, 60H, 0EH, 06H, 7EH, 6EH, 66H, 66H
0106 60
0107 0E
0108 06
0109 7E
010A 6E
010B 66
010C 66
010D 66           219                                db      66H, 66H, 06H, 18H, 66H, 66H, 42H, 3CH
010E 66
010F 06
0110 18
0111 66
0112 66
0113 42
0114 3C
0115 3C           220                                db      3CH, 30H
0116 30
0117 3C           221          slice_7_d             db      3CH, 66H, 6CH, 36H, 06H, 06H, 6CH, 66H
0118 66
0119 6C
011A 36
011B 06
011C 06
011D 6C
011E 66
011F 18           222                                db      18H, 60H, 36H, 06H, 66H, 66H, 66H, 66H
0120 60
0121 36
0122 06
0123 66
0124 66
0125 66
0126 66
0127 66           223                                db      66H, 66H, 66H, 18H, 66H, 42H, 42H, 66H
0128 66
0129 66
012A 18
012B 66
012C 42
012D 42
012E 66
012F 66           224                                db      66H, 60H
0130 60
0131 18           225          slice_8_d             db      18H, 3EH, 38H, 1EH, 7EH, 7EH, 38H, 66H
0132 3E
0133 38
0134 1E
0135 7E
0136 7E
0137 38
```

231679-23

```
LOC  OBJ                    LINE     SOURCE

0138 66
0139 3C                      226                                      db       3CH, 70H, 66H, 06H, 24H, 62H, 3CH, 3EH
013A 70
013B 66
013C 06
013D 24
013E 62
013F 3C
0140 3E
0141 3C                      227                                      db       3CH, 3EH, 3CH, 7EH, 66H, 42H, 42H, 66H
0142 3E
0143 3C
0144 7E
0145 66
0146 42
0147 42
0148 66
0149 66                      228                                      db       66H, 7EH
014A 7E
                             229
----                         230     video_data              ends
                             231
                             232
                             233
                             234     ; monitor segment. The GSP board monitor starts at OFFFOh
----                         235     monitor                 segment          at       OFFFOh
                             236
0000                         237                                      org 0
0000 (1                      238             reset           dw               1 dup(?)         ; a label
     ????
     )
                             239
----                         240     monitor                 ends
                             241     ;-------------------------------------------------------------------
                             242     ;          Main routine. This routine loads the DRAM with access table :
                             243     ;          the object descriptor table,the character generator and the :
                             244     ;          actual object data. The VSDD is also initialized.          !
                             245     ;--------------------------------------------------------------------
                             246
----                         247     prog_code               segment
                             248
                             249                     assume  cs:prog_code, ds:video_data, es:video_vsdd
                             250
0000                         251     simple_display          proc     far
0000                         252     start:
                             253
0000 B8----          R       254                             mov ax,video_data       ;initialize the data segment
0003 8ED8                    255                             mov ds,ax
                             256
0005 B80060                  257                             mov ax,video_vsdd       ;initialize the extra
0008 8EC0                    258                             mov es,ax                              ;segment
                             259
                             260     ; initialize the register segment. the register
                             261     ; segment is located at 0400h after reset.
                             262
```

231679-24



```
LOC  OBJ                  LINE     SOURCE

000A 26C70600047260        263                           mov ir0_v,r0_res
0011 26C706020414A4        264                           mov ir1_v,r1_d
0018 26C70604040600        265                           mov ir2_v,r2_d
001F 26C70606044001        266                           mov ir3_v,r3_d
0026 26C70608040080        267                           mov ir4_v,r4_d
002D 26C7060A040000        268                           mov ir5_v,r5_d
0034 26C7060C040A00        269                           mov ir6_v,r6_d
003B 26C7060E040005        270                           mov ir7_v,r7_d
0042 26C70610041000        271                           mov ir8_v,r8_d
0049 26C70612048001        272                           mov ir9_v,r9_d
0050 26C70614042300        273                           mov ir10_v,r10_d
0057 26C70616041000        274                           mov ir11_v,r11_d
005E 26C70618040204        275                           mov ir12_v,r12_d
0065 26C7061A042410        276                           mov ir13_v,r13_d
006C 26C7061C04EC74        277                           mov ir14_v,r14_d
0073 26C7061E04F480        278                           mov ir15_v,r15_d
                           279
                           280
                           281     ; all the registers are initialized in the DRAM. Enable the UCF
                           282     ; flag to allow the VSDD to update its on chip registers.
                           283
007A 26C70600047360        284                           mov ir0_v,r0_up
                           285
                           286     ; wait 150us for the VSDD to update its on chip registers
                           287     ; the loop assumes that the 80186 runs at 8Mhz.
                           288
0081 B94700                289                           mov cx,71
0084 E2FE                  290                   loop1:  loop loop1
                           291     ; the register window is initialized to 60000h
                           292     ; the cpu programs the display data through newly
                           293     ; defined data window in R3.
                           294
                           295     ;load the object descriptor fields for four objects
                           296
                           297     ;object 0
                           298
0086 26C706000A0006        299                           mov odt0_v,600h          ;4bits/pixel,non-transparent
008D 26C706020A0018        300                           mov odt0_v[2],1800h      ;object starts at x = 0
                           301                                                    ;the width is 96 pixels
0094 26C706040A0010        302                           mov odt0_v[4],1000h      ;object base address
009B 26C706060A0010        303                           mov odt0_v[6],1000h
                           304
                           305     ;object 1
                           306
00A2 26C706080A0006        307                           mov odt1_v,600h
00A9 26C7060A0A1404        308                           mov odt1_v[2],0414h      ; x = 20,width = 16 pixels
00B0 26C7060C0A0A17        309                           mov odt1_v[4],170ah
00B7 26C7060E0A0A17        310                           mov odt1_v[6],170ah
                           311
                           312     ;object 2
                           313
00BE 26C706100A0006        314                           mov odt2_v,600h
00C5 26C706120A263C        315                           mov odt2_v[2],3c26h      ; x = 38;width = 240 pixels
00CC 26C706140A6017        316                           mov odt2_v[4],1760h
00D3 26C706160A6017        317                           mov odt2_v[6],1760h
```

231679-25

```
LOC  OBJ                   SOURCE


                           ;object 3

00DA 26C706180A04AC                                mov odt3_v,0AC04h       ;character object
                                                                           ;8 pixels/character
                                                                           ;transparent pixel
00E1 26C7061A0A5008                                mov odt3_v[2],0850h     ;x = 80, Width = 16
                                                                           ; characters
00E8 26C7061C0ADA17                                mov odt3_v[4],17DAh
00EF 26C7061E0ADA17                                mov odt3_v[6],17DAh


                           ; set up the object data

00F6 BA0200                                        mov dx,2
00F9 BB0000                                        mov bx,0

                           ;object 0
00FC B90807                                        mov cx,24*75            ; number of data words -
                                                                                   ;75 lines,24 words(96 pixels)
00FF B88888                                        mov ax,8888h            ;pixel data

0102 2689870020            fill_obj_0:              mov object_0_v[bx],ax
0107 03DA                                          add bx,dx
0109 E2F7                  loop fill_obj_0

                           ;object 1

010B BB0000                                        mov bx,0
010E B95000                                        mov cx,4*20             ;number of data words
0111 B87777                                        mov ax,7777h            ;pixel data

0114 268987142E            fill_obj_1:             mov object_1_v[bx],ax
0119 03DA                                          add bx,dx
011B E2F7                  loop fill_obj_1

                           ;object 2

011D BB0000                                        mov bx,0
0120 B93C00                                        mov cx,15*4
0123 B85555                                        mov ax,5555h

0126 268987C02E            fill_obj_2:             mov object_2_v[bx],ax
012B 03DA                                          add bx,dx
012D E2F7                  loop fill_obj_2

                           ;object 3 - character object

012F BB0000                                        mov bx,0
0132 B92800                                        mov cx,40                       ;total 80 characters
                                                                                   ;in the object,2/word

0135 8B07                  fill_obj_3:             mov ax,word ptr object_3_data[bx]  ; read the ASCII code
                                                                                          ; for 2 characters
```

231679-26

```
LOC  OBJ                  SOURCE

0137 268987B42F                                    mov object_3_v[bx],ax            ; write the data
013C 03DA                                          add bx,dx
013E E2F5                 loop fill_obj_3

                          ;load the charactor generator

0140 B80000                                        mov ax,0
0143 BB0000                                        mov bx,0
                          ;load the numbers

0146 BE6000                                        mov si,30h*2          ;store at proper ASCII
                                                                         ;location. Note cpu space
                                                                         ;byte addressable
0149 B90A00                                        mov cx,10             ;10 numbers
014C BA0700                                        mov dx,7              ; 7 slices

014F                      write_a_number:

014F 8A474F                                        mov al,numbers_data[bx]          ;read data byte
0152 2689840044                                    mov cg0_slice_2_v[si],ax         ;write data word in
                                                                                    ;the DRAM
0157 43                                            inc bx                           ; next byte
0158 83C602                                        add si,2                         ; next location in
                                                                                    ; the DRAM
015B E2F2                 loop write_a_number

015D 81C6EC01                                      add si,(256*2)-20                ;next slice
0161 B90A00                                          mov cx,10
0164 4A                                            dec dx

0165 75E8                 jnz write_a_number

                          ;store the 26 alphabets

0167 B80000                                        mov ax,0
016A BB0000                                        mov bx,0
016D B91A00                                        mov cx,26             ; 26 alphabets
0170 BE8200                                        mov si,41h*2          ;proper offset into
                                                                         ;character generator
0173 BA0700                                        mov dx,7              ; 7 slices

0176                      write_a_character:

0176 8A879500                                      mov al,slice_2_d[bx]             ;read data byte
017A 2689840044                                    mov cg0_slice_2_v[si],ax         ; write a word
017F 43                                            inc bx                           ; next byte
0180 83C602                                        add si,2                         ;next location

0183 E2F1                 loop write_a_character

0185 B91A00                                        mov cx,26
0188 81C6CC01                                      add si,(256*2)-52            ;next slice
018C 4A                                            dec dx
018D 75E7                 jnz write_a_character

```

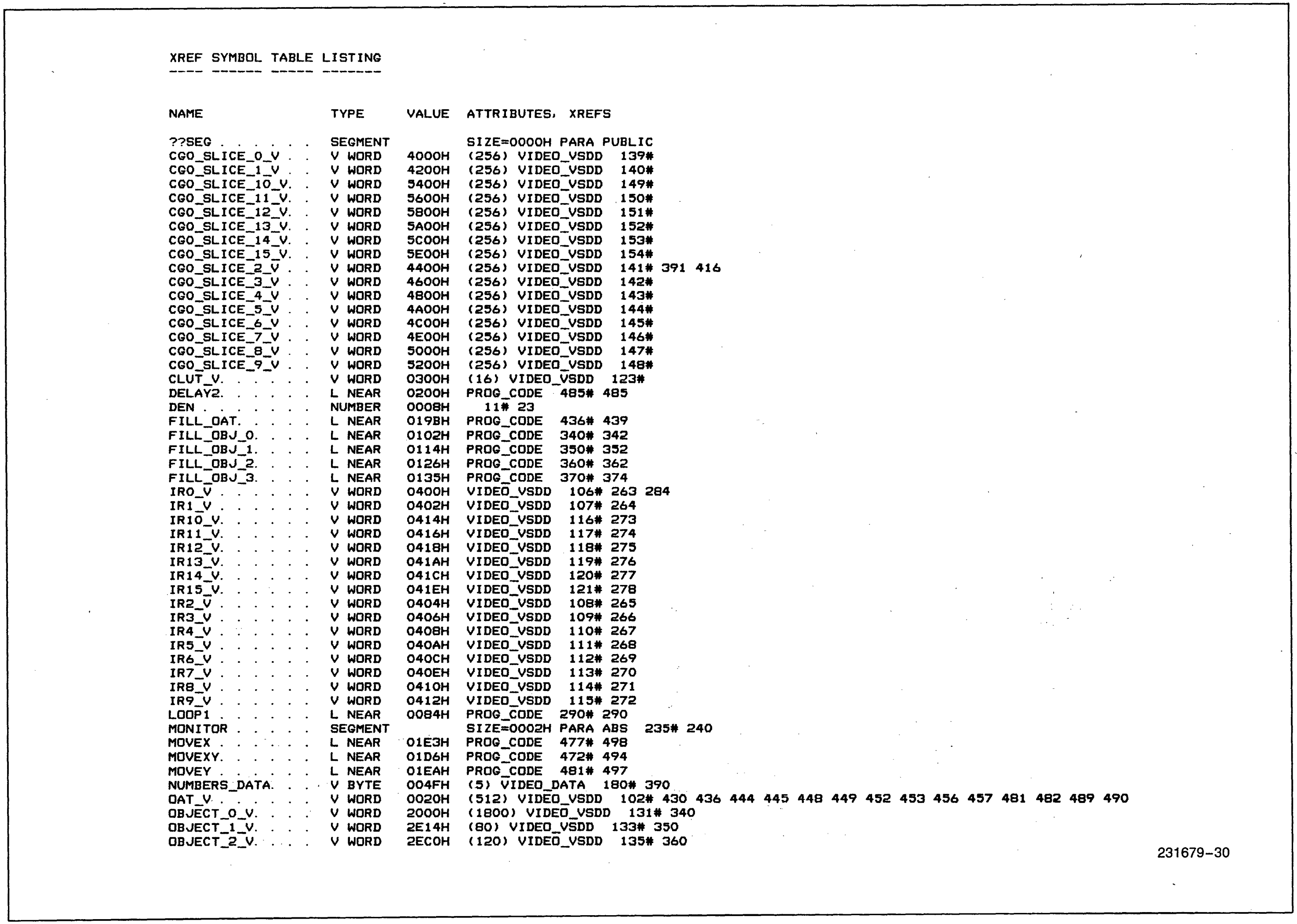
231679-27

```
LOC  OBJ                SOURCE

                        ;load the access table

018F BB0000                                   mov bx,0
0192 B90002                                   mov cx,length oat_v
0195 B8FFFF                                   mov ax,0ffffh
0198 BA0200                                   mov dx,2

                        ; fill the access table with all 1s

019B 26894720           fill_oat:             mov oat_v[bx],ax
019F 03DA                                     add bx,dx

01A1 E2F8               loop fill_oat

                        ;enable the objects

01A3 B8FEFF                                   mov ax,0fffeh
01A6 26A32000                                 mov oat_v,ax            ;enable object 0 at line 0
01AA 26A3B600                                 mov oat_v[75*2],ax  ;disable object 0 at line 75

01AE B8FDFF                                   mov ax,0fffdh
01B1 26A30601                                 mov oat_v[115*2],ax ;enable object 1 at line 115
01B5 26A32C01                                 mov oat_v[134*2],ax ;disable object 1 at line 131

01B9 B8FBFF                                   mov ax,0fffbh
01BC 26A33801                                 mov oat_v[140*2],ax ;enable object 2 at line 140
01C0 26A33A01                                 mov oat_v[141*2],ax ;disable object 2 at line 141

01C4 B8F7FF                                   mov ax,0fff7h
01C7 26A34800                                 mov oat_v[20*2],ax  ;enable object 3 at line 20
01CB 26A3AC00                                 mov oat_v[70*2],ax  ;disable object 3 at line 70


                        ; the display data is initialized by the 80186. Now enable the
                        ; set the display enable bit (DEN) in the VSDD to enable the
                        ; display.

01CF 26C70600007B60                           mov r0_v,r0_disp   ; the register segment  is now located
                                                                 ; at 60000h


                        ; a simple routine to scroll objects horizontally and vertically
                        ; object 0 is moved horozontally while object 1 is moved vertic-
                        ; ally

01D6                      movexy:
01D6 26C706020A0018                           mov odt0_v[2],1800h
01DD BA4201                                   mov dx,322              ; maximum value of x for obj 0
01E0 BB0000                                   mov bx,0                ; start y value for obj 1

01E3 268306020A01               movex:        add odt0_v[2],1 ;mov obj 0 by 2 pixels in x direction
01E9 4A                                       dec dx


01EA 26816720FDFF              movey:  and oat_v[bx],0fffdh             ;object 1 start
```

231679-28

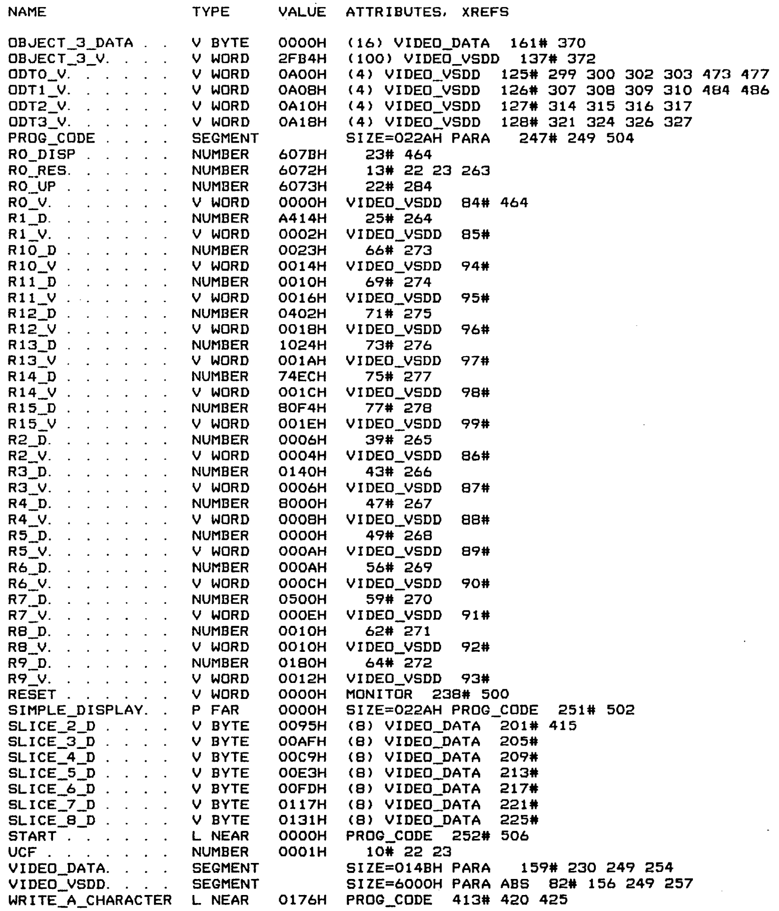
```
LOC  OBJ                    LINE     SOURCE

01F0 26816748FDFF            482                                and oat_v[bx+40],0fffdh         ;object 1 stop
01F6 B97017                  483                                mov cx,6000                     ;delay counter
01F9 26C706080A0006          484                                mov odt1_v,600h                 ;turn the object on
0200 E2FE                    485             delay2: loop delay2
0202 26C706080A1006          486                                mov odt1_v,610h         ;turn the object off
                             487                                                        ; before disabling the
                             488                                                        ; access table
0209 26814F20F2FF            489                                or  oat_v[bx],0fff2h    ; reset the access table
020F 26814F48F2FF            490                                or  oat_v[bx+40],0fff2h ; to original values
0215 83C302                  491                                add bx,2
0218 81FB3403                492                                cmp bx,820              ; max value of y is 410
                             493
021C 74B8                    494                                je  movexy              ; if y = max then start
                             495                                                        ; at top of frame
021E 83FA00                  496                                   cmp dx,0
0221 74C7                    497                                je movey                ; if x = max then finish y move
0223 EBBE                    498                                jmp movex               ; else continue with x move
                             499
0225 EA0000F0FF              500                                jmp far ptr     RESET   ; go back to the board monitor
                             501
                             502     simple_display             endp
                             503
----                         504     prog_code                          ends
                             505
                             506                                end     start
```

231679-29



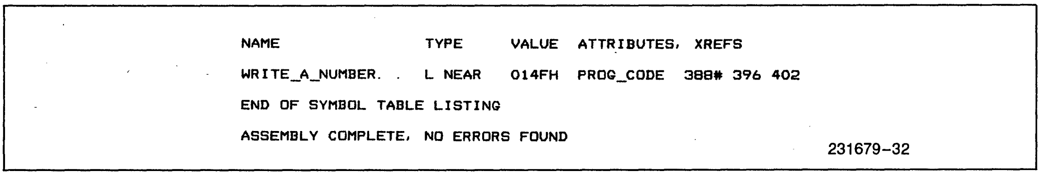
```
XREF SYMBOL TABLE LISTING
---- ------ ----- -------


NAME                 TYPE      VALUE  ATTRIBUTES, XREFS

??SEG . . . . . . .  SEGMENT          SIZE=0000H PARA PUBLIC
CGO_SLICE_0_V . . .  V WORD    4000H  (256) VIDEO_VSDD  139#
CGO_SLICE_1_V . . .  V WORD    4200H  (256) VIDEO_VSDD  140#
CGO_SLICE_10_V. . .  V WORD    5400H  (256) VIDEO_VSDD  149#
CGO_SLICE_11_V. . .  V WORD    5600H  (256) VIDEO_VSDD  150#
CGO_SLICE_12_V. . .  V WORD    5800H  (256) VIDEO_VSDD  151#
CGO_SLICE_13_V. . .  V WORD    5A00H  (256) VIDEO_VSDD  152#
CGO_SLICE_14_V. . .  V WORD    5C00H  (256) VIDEO_VSDD  153#
CGO_SLICE_15_V. . .  V WORD    5E00H  (256) VIDEO_VSDD  154#
CGO_SLICE_2_V . . .  V WORD    4400H  (256) VIDEO_VSDD  141# 391 416
CGO_SLICE_3_V . . .  V WORD    4600H  (256) VIDEO_VSDD  142#
CGO_SLICE_4_V . . .  V WORD    4800H  (256) VIDEO_VSDD  143#
CGO_SLICE_5_V . . .  V WORD    4A00H  (256) VIDEO_VSDD  144#
CGO_SLICE_6_V . . .  V WORD    4C00H  (256) VIDEO_VSDD  145#
CGO_SLICE_7_V . . .  V WORD    4E00H  (256) VIDEO_VSDD  146#
CGO_SLICE_8_V . . .  V WORD    5000H  (256) VIDEO_VSDD  147#
CGO_SLICE_9_V . . .  V WORD    5200H  (256) VIDEO_VSDD  148#
CLUT_V. . . . . . .  V WORD    0300H  (16) VIDEO_VSDD  123#
DELAY2. . . . . . .  L NEAR    0200H  PROG_CODE  485# 485
DEN . . . . . . . .  NUMBER    0008H    11# 23
FILL_OAT. . . . . .  L NEAR    019BH  PROG_CODE  436# 439
FILL_OBJ_0. . . . .  L NEAR    0102H  PROG_CODE  340# 342
FILL_OBJ_1. . . . .  L NEAR    0114H  PROG_CODE  350# 352
FILL_OBJ_2. . . . .  L NEAR    0126H  PROG_CODE  360# 362
FILL_OBJ_3. . . . .  L NEAR    0135H  PROG_CODE  370# 374
IR0_V . . . . . . .  V WORD    0400H  VIDEO_VSDD  106# 263 284
IR1_V . . . . . . .  V WORD    0402H  VIDEO_VSDD  107# 264
IR10_V. . . . . . .  V WORD    0414H  VIDEO_VSDD  116# 273
IR11_V. . . . . . .  V WORD    0416H  VIDEO_VSDD  117# 274
IR12_V. . . . . . .  V WORD    0418H  VIDEO_VSDD  118# 275
IR13_V. . . . . . .  V WORD    041AH  VIDEO_VSDD  119# 276
IR14_V. . . . . . .  V WORD    041CH  VIDEO_VSDD  120# 277
IR15_V. . . . . . .  V WORD    041EH  VIDEO_VSDD  121# 278
IR2_V . . . . . . .  V WORD    0404H  VIDEO_VSDD  108# 265
IR3_V . . . . . . .  V WORD    0406H  VIDEO_VSDD  109# 266
IR4_V . . . . . . .  V WORD    0408H  VIDEO_VSDD  110# 267
IR5_V . . . . . . .  V WORD    040AH  VIDEO_VSDD  111# 268
IR6_V . . . . . . .  V WORD    040CH  VIDEO_VSDD  112# 269
IR7_V . . . . . . .  V WORD    040EH  VIDEO_VSDD  113# 270
IR8_V . . . . . . .  V WORD    0410H  VIDEO_VSDD  114# 271
IR9_V . . . . . . .  V WORD    0412H  VIDEO_VSDD  115# 272
LOOP1 . . . . . . .  L NEAR    0084H  PROG_CODE  290# 290
MONITOR . . . . . .  SEGMENT          SIZE=0002H PARA ABS  235# 240
MOVEX . . . . . . .  L NEAR    01E3H  PROG_CODE  477# 498
MOVEXY. . . . . . .  L NEAR    01D6H  PROG_CODE  472# 494
MOVEY . . . . . . .  L NEAR    01EAH  PROG_CODE  481# 497
NUMBERS_DATA. . . .  V BYTE    004FH  (5) VIDEO_DATA  180# 390
OAT_V . . . . . . .  V WORD    0020H  (512) VIDEO_VSDD  102# 430 436 444 445 448 449 452 453 456 457 481 482 489 490
OBJECT_0_V. . . . .  V WORD    2000H  (1800) VIDEO_VSDD  131# 340
OBJECT_1_V. . . . .  V WORD    2E14H  (80) VIDEO_VSDD  133# 350
OBJECT_2_V. . . . .  V WORD    2EC0H  (120) VIDEO_VSDD  135# 360
```

231679-30



```
NAME               TYPE     VALUE  ATTRIBUTES, XREFS

OBJECT_3_DATA . .  V BYTE   0000H  (16) VIDEO_DATA  161# 370
OBJECT_3_V. . . .  V WORD   2FB4H  (100) VIDEO_VSDD  137# 372
ODT0_V. . . . . .  V WORD   0A00H  (4) VIDEO_VSDD  125# 299 300 302 303 473 477
ODT1_V. . . . . .  V WORD   0A08H  (4) VIDEO_VSDD  126# 307 308 309 310 484 486
ODT2_V. . . . . .  V WORD   0A10H  (4) VIDEO_VSDD  127# 314 315 316 317
ODT3_V. . . . . .  V WORD   0A18H  (4) VIDEO_VSDD  128# 321 324 326 327
PROG_CODE . . . .  SEGMENT         SIZE=022AH PARA   247# 249 504
R0_DISP . . . . .  NUMBER   607BH    23# 464
R0_RES. . . . . .  NUMBER   6072H    13# 22 23 263
R0_UP . . . . . .  NUMBER   6073H    22# 284
R0_V. . . . . . .  V WORD   0000H  VIDEO_VSDD  84# 464
R1_D. . . . . . .  NUMBER   A414H    25# 264
R1_V. . . . . . .  V WORD   0002H  VIDEO_VSDD  85#
R10_D . . . . . .  NUMBER   0023H    66# 273
R10_V . . . . . .  V WORD   0014H  VIDEO_VSDD  94#
R11_D . . . . . .  NUMBER   0010H    69# 274
R11_V . . . . . .  V WORD   0016H  VIDEO_VSDD  95#
R12_D . . . . . .  NUMBER   0402H    71# 275
R12_V . . . . . .  V WORD   0018H  VIDEO_VSDD  96#
R13_D . . . . . .  NUMBER   1024H    73# 276
R13_V . . . . . .  V WORD   001AH  VIDEO_VSDD  97#
R14_D . . . . . .  NUMBER   74ECH    75# 277
R14_V . . . . . .  V WORD   001CH  VIDEO_VSDD  98#
R15_D . . . . . .  NUMBER   80F4H    77# 278
R15_V . . . . . .  V WORD   001EH  VIDEO_VSDD  99#
R2_D. . . . . . .  NUMBER   0006H    39# 265
R2_V. . . . . . .  V WORD   0004H  VIDEO_VSDD  86#
R3_D. . . . . . .  NUMBER   0140H    43# 266
R3_V. . . . . . .  V WORD   0006H  VIDEO_VSDD  87#
R4_D. . . . . . .  NUMBER   8000H    47# 267
R4_V. . . . . . .  V WORD   0008H  VIDEO_VSDD  88#
R5_D. . . . . . .  NUMBER   0000H    49# 268
R5_V. . . . . . .  V WORD   000AH  VIDEO_VSDD  89#
R6_D. . . . . . .  NUMBER   000AH    56# 269
R6_V. . . . . . .  V WORD   000CH  VIDEO_VSDD  90#
R7_D. . . . . . .  NUMBER   0500H    59# 270
R7_V. . . . . . .  V WORD   000EH  VIDEO_VSDD  91#
R8_D. . . . . . .  NUMBER   0010H    62# 271
R8_V. . . . . . .  V WORD   0010H  VIDEO_VSDD  92#
R9_D. . . . . . .  NUMBER   0180H    64# 272
R9_V. . . . . . .  V WORD   0012H  VIDEO_VSDD  93#
RESET . . . . . .  V WORD   0000H  MONITOR  238# 500
SIMPLE_DISPLAY. .  P FAR    0000H  SIZE=022AH PROG_CODE  251# 502
SLICE_2_D . . . .  V BYTE   0095H  (8) VIDEO_DATA  201# 415
SLICE_3_D . . . .  V BYTE   00AFH  (8) VIDEO_DATA  205#
SLICE_4_D . . . .  V BYTE   00C9H  (8) VIDEO_DATA  209#
SLICE_5_D . . . .  V BYTE   00E3H  (8) VIDEO_DATA  213#
SLICE_6_D . . . .  V BYTE   00FDH  (8) VIDEO_DATA  217#
SLICE_7_D . . . .  V BYTE   0117H  (8) VIDEO_DATA  221#
SLICE_8_D . . . .  V BYTE   0131H  (8) VIDEO_DATA  225#
START . . . . . .  L NEAR   0000H  PROG_CODE  252# 506
UCF . . . . . . .  NUMBER   0001H    10# 22 23
VIDEO_DATA. . . .  SEGMENT         SIZE=014BH PARA   159# 230 249 254
VIDEO_VSDD. . . .  SEGMENT         SIZE=6000H PARA ABS  82# 156 249 257
WRITE_A_CHARACTER  L NEAR   0176H  PROG_CODE  413# 420 425
```

231679-31



```
NAME              TYPE     VALUE  ATTRIBUTES, XREFS

WRITE_A_NUMBER. . L NEAR   014FH  PROG_CODE  388# 396 402

END OF SYMBOL TABLE LISTING

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

231679-32

October 1987

82786 Hardware Configuration

Order Number: 292007-003

1.0 INTRODUCTION

The 82786 is an intelligent coprocessor capable of creating and displaying high performance graphics. Both drawing and display functions are integrated into a single VLSI chip to provide an inexpensive solution for bitmapped graphics subsystems.

This application note is intended to show, through examples, use of the 82786 and the hardware interfaces between the 82786 and the rest of the system. Because the 82786 integrates many functions onto one chip, the hardware design of a graphics system is greatly simplified.

2.0 OVERVIEW

Internally, the 82786 consists of two independent processors.

— Graphics Processor: executes high-level line drawing, character drawing and bit-block-transfer commands to create and modify bitmaps in memory

— Display Processor: displays portions of bitmaps in regions on the CRT called windows

Figure 1 illustrates these processors and their hardware interfaces.

— Graphics Memory Interface: connects dedicated graphics memory to the 82786

— System Bus Interface: connects CPU and system memory to the 82786

— Video Interface: connects the 82786 to CRT or other display

The video interface is controlled directly by the Display Processor. The other interfaces are controlled by the 82786 Bus Interface Unit (BIU). The BIU connects the internal Graphics and Display Processors to the CPU and system memory as well as to the graphics memory through the internal DRAM/VRAM controller.

2.1 Dedicated Graphics Memory

The dedicated graphics memory provides the 82786 with very fast access to memory without contention with the CPU and system memory. Typically, the bitmaps to be drawn and displayed, the character fonts, and the command lists for the 82786 processors are all stored in this memory. In some instances it is desirable to have the Graphics Processor command lists stored in system memory.

The 82786 contains a complete DRAM/VRAM controller on-chip which interfaces directly with a wide variety of DRAMs without external logic. This direct connection not only reduces chip count but also allows the 82786 to perform very fast burst accesses to the DRAMs. The DRAM/VRAM controller can take advantage of the quick burst-mode sequential accesses made possible by Page Mode, Fast Page Mode (sometimes called Ripplemode™), and Static Column DRAMs. In addition, interleaved DRAM/VRAM arrays are fully supported by the on-chip DRAM/VRAM controller allowing even faster burst access.

2.2 System Bus Interface

The system bus interface connects the CPU and its system memory to the 82786 and its graphics memory.

The most common 82786 configuration (shown in Figure 1) allows the CPU to access the system memory while the 82786 accesses its dedicated graphics memory simultaneously. It also allows the CPU to access the graphics memory and for the 82786 to access the system memory (but not simultaneously). The system bus

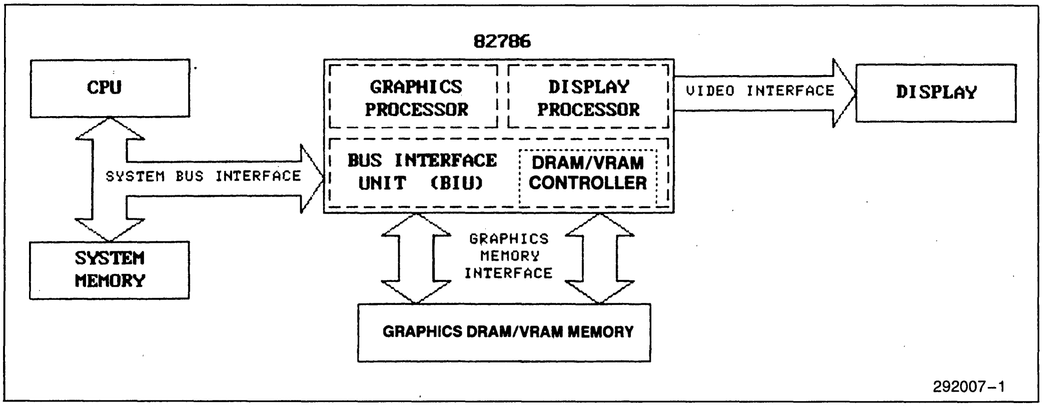


Figure 1. 82786 System Block Diagram

connects the 82786 graphics subsystem to the system CPU and memory. If DMA capability is also provided in the system, it interfaces to the 82786 exactly as the CPU does. The interface allows accesses in two directions.

- — Slave Mode: CPU or DMA read or write access of the 82786 internal registers or dedicated graphics memory through the 82786
- — Master Mode: 82786 read or write access to system memory

Therefore, any processor (CPU, DMA, Graphics and Display Processors) can access both the system memory and the graphics memory. The 82786 BIU arbitrates between both of the internal 82786 processors as well as the external processor (CPU or DMA) to decide which processor gets access of the bus.

The CPU software accesses both system and graphics memory in an identical manner (except that the specific memory addresses are different). Therefore the actual location of the memory (whether in system or graphics memory) is transparent to the software. However, the CPU can access the system memory faster than the graphics memory because there is less contention with the Graphics and Display Processors. When the CPU accesses the 82786, the 82786 runs in slave mode.

In slave mode, the 82786 looks like an intelligent DRAM/VRAM controller to the CPU (Figure 2). The CPU can chip-select the 82786 and the 82786 will acknowledge when the cycle is complete by generating a READY signal for the CPU.

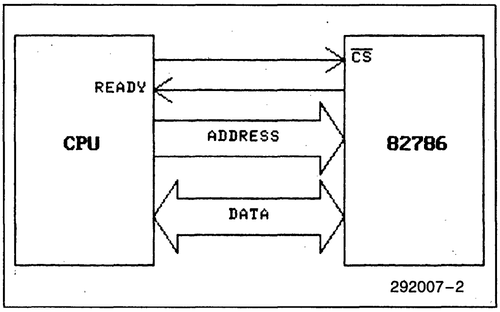


Figure 2. Slave Bus Cycle

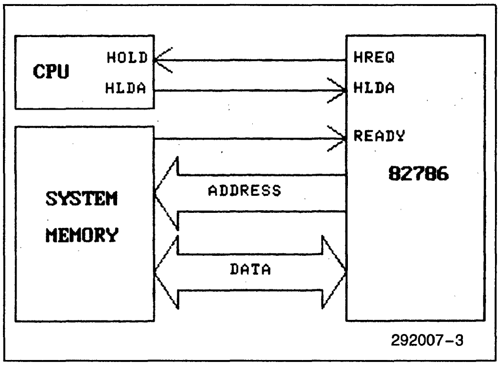


Figure 3. Master Bus Cycle

Conversely, the 82786 Graphics and Display Processors access both system memory and graphics memory in an identical manner. However, they can access graphics memory faster than system memory because there is less contention with the CPU. When the 82786 accesses system memory, the 82786 runs in master mode.

In master mode, the 82786 looks like a second CPU controlling the local bus (Figure 3). The 82786 activates HOLD to request control of the system bus. When the CPU acknowledges the HLDA line, then the 82786 will take over the bus. When the 82786 is through with the bus, it will release HOLD and the CPU can remove HLDA to regain control of the bus.

The 82786 system bus interface is optimized to interface to an 80286 synchronously (using the same bus clock). As a synchronous slave it interprets the 80286 status lines directly and performs the requested bus accesses. As a master it generates 80286 style bus signals.

The 82786 system bus may alternatively be set up to interface asynchronously to virtually any processor. In this mode, read and write signals are used when slave accesses are performed.

2.3 Video Interface

The 82786 supports two different video interfaces in order to support both standard DRAMs and dual port video DRAMs/(VRAMs). When using standard

DRAMs the 82786 reads the video data from memory and internally serializes the video data to generate the serial video data stream up to 25 MHz. When using VRAMs the 82786 loads the VRAM shift register periodically; then the shift register and external logic generate the serial video data stream.

With standard DRAMs displays up to 640 by 480 by 8 resolution can be generated at 60 Hz noninterlaced refresh. With VRAMs displays up to 2048 by 1936 by 8 can be generated at 60 Hz without interlacing.

In addition, horizontal and vertical sync signals and a blanking signal are provided and may be programmed to satisfy the requirements of nearly any CRT.

In the standard DRAM mode all of the logic to support the advanced capabilities of the Display Processor such as panning, zooming, windowing, and switching between various bits/pixel in various windows is contained internally in the 82786. Provision is also made for the addition of up to four external color look-up tables.

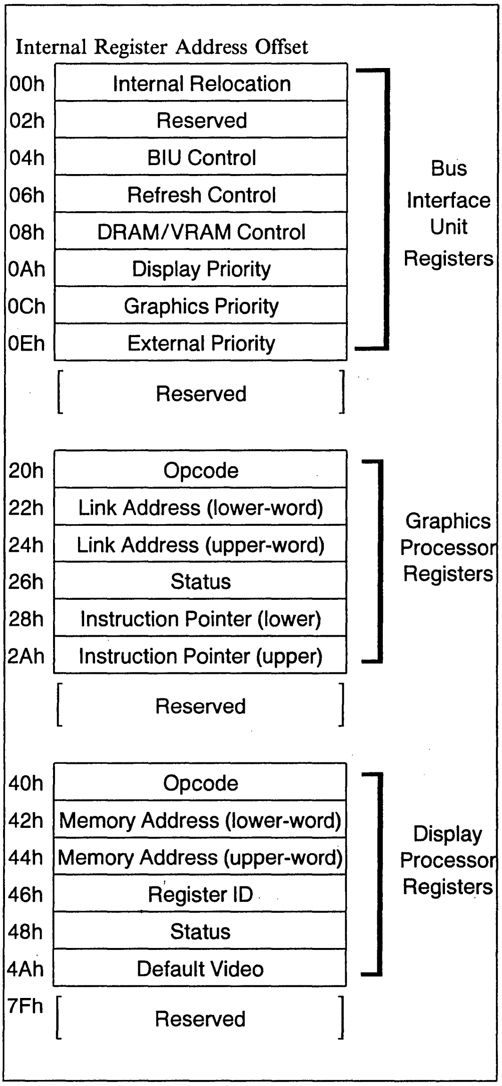
Higher resolution displays (dot clock rates greater than 25 MHz) can also be implemented by using external logic to trade-off bits/pixel for dot clock rates. Also, multiple 82786s can be used together for even greater performance.

2.4 82786 Internal Registers

A 64 word (128 byte) direct-mapped register block is contained internally in the 82786 (Figure 4). Software may locate this register block to the beginning of any 128 even byte boundary anywhere in the 82786 I/O or memory address space. No matter where these registers are mapped, they are only accessible by the external CPU. The Graphics and Display Processors can not access these registers.

Registers, located at specified offsets within this block, allow programming of the BIU and Graphics and Display Processors. The Graphics and Display Processors also have other registers which are only accessible through commands to these processors. These commands are initiated by writing into the corresponding opcode and address registers within this 128 byte register block.

All of these registers are described in detail in the 82786 data sheet. Do not use "Reserved" Registers. When these reserved registers are read, the data returned is indeterminate. Reserved Registers should only be written as zeros to ensure compatibility with future products.



Internal Register Address Offset

00h	Internal Relocation	Bus Interface Unit Registers
02h	Reserved	
04h	BIU Control	
06h	Refresh Control	
08h	DRAM/VRAM Control	
0Ah	Display Priority	
0Ch	Graphics Priority	
0Eh	External Priority	
	Reserved	
20h	Opcode	Graphics Processor Registers
22h	Link Address (lower-word)	
24h	Link Address (upper-word)	
26h	Status	
28h	Instruction Pointer (lower)	
2Ah	Instruction Pointer (upper)	
	Reserved	
40h	Opcode	Display Processor Registers
42h	Memory Address (lower-word)	
44h	Memory Address (upper-word)	
46h	Register ID	
48h	Status	
4Ah	Default Video	
7Fh	Reserved	

Figure 4. 82786 Internal Registers

3.0 DEDICATED GRAPHICS MEMORY INTERFACE

The 82786 contains a full DRAM/VRAM controller on-chip which allows it to be connected directly to arrays of DRAMs without external logic.

A wide range of DRAM configurations are possible for x 1, x 4 and x 8 bit wide DRAMs. Both Page Mode and

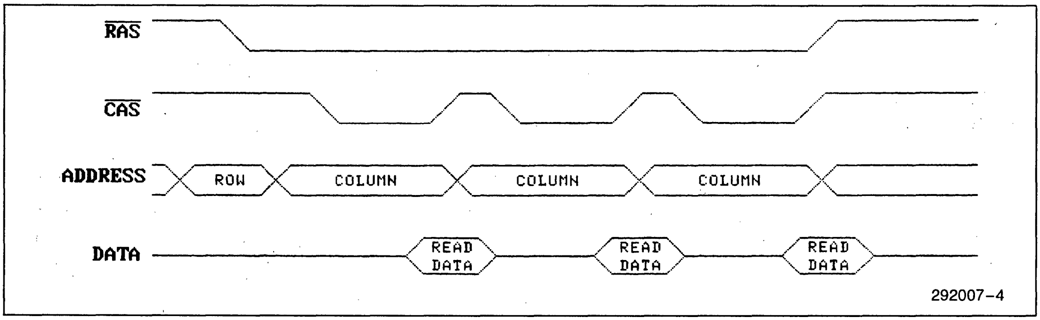


Figure 5. Fast Page Mode Burst-Access Read Cycle

Fast-page-mode burst accesses for block transfers are supported directly by the 82786 to take advantage of the fast sequential addressing capability of DRAMs (see Figure 5). Once the DRAM is set-up with the row address, the column addresses can be quickly scanned in for several burst-accesses to the same page. With the 82786, Fast Page Mode bursts for block transfers run at twice the speed of page mode.

Interleaving of two banks of DRAMs is also supported directly by the 82786. For a sequential burst access, DRAM cycles for both banks can be initiated. Then, during the burst access, the 82786 can alternate accesses between the two banks, thus cutting the effective DRAM access time in half (see Figure 6).

Static Column DRAMs can also be used to get the same performance as Fast Page Mode. The only difference between the two types is that Static Column DRAMs do not latch the column address, whereas, Fast Page Mode DRAMs do latch the column address on the falling edge of CAS. In noninterleaved configurations, Static Column DRAMs can directly replace Fast Page Mode. However, in an interleaved configuration, the column address must be latched externally for Static Column DRAMs.

The following table shows the burst-access rate of these various configurations for a 10 MHz 82786.

	Page Mode	Fast Page Mode and Static Column
Noninterleaved:	10 Mbyte/sec (2 cycles)	20 Mbyte/sec (1 cycle)
Interleaved:	20 Mbyte/sec (1 cycle)	30 Mbyte/sec (0.5 cycle)

The other cycle times, and speeds at 10 MHz, are the same for all DRAM configurations:

Single Reads	3 cycles	300 ns
Single Writes	3 cycles	300 ns
Read-Modify-Writes	4 cycles	400 ns
Burst-Access Set-Up	2 cycles	200 ns
Refresh	3 cycles	300 ns

All burst-accesses for block transfers perform an even number of 16-bit word accesses.

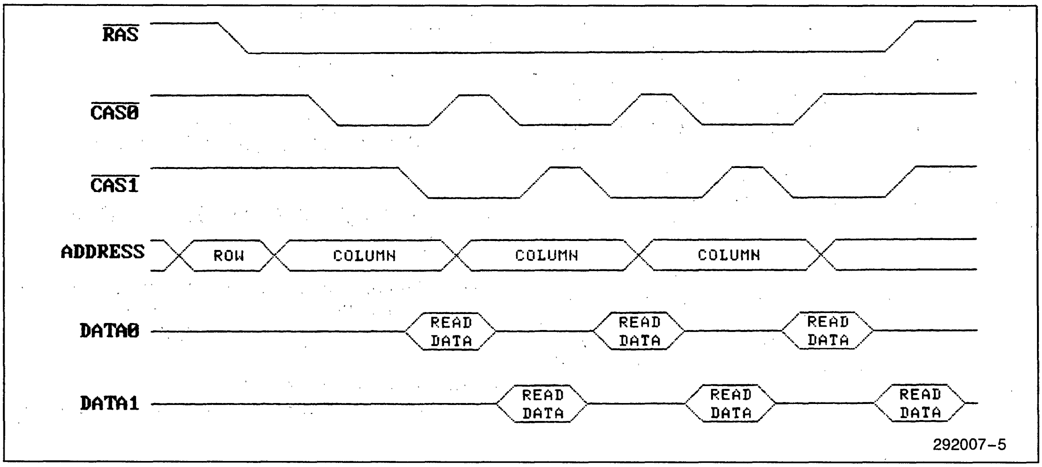


Figure 6. Interleaved Fast-Page-Mode Burst-Access Read Cycle

Burst-accesses for block transfers are used by all Display Processor memory accesses except the operand for LD_REG and DMP_REG commands. Block-read accesses are used by the Graphics Processor for command-block fetching and to fetch the character fonts. The Graphics Processor uses a block-read followed by a block-write for the read-modify-write operations of BitBlt, Scan_Line, and Character drawing. All other pixel drawing commands use single read-modify-write cycles.

3.1 DRAM Configurations

Up to 4 rows per bank, and 1 noninterleaved or 2 interleaved banks are supported (see Figure 7). Each bank must always be 16 bits wide. If only one noninterleaved bank is used, it must be bank 0 (using $\overline{\text{CAS0}}$ and $\overline{\text{BEN0}}$). If interleaving is used, both banks must have the same number of rows. In either case, if only one row is used, it must be row 0 (using $\overline{\text{RAS0}}$). For only two rows, row 0 and 1 are used ($\overline{\text{RAS0}}$ and $\overline{\text{RAS1}}$). Similarly, three rows use row 0, 1, and 2.

The 82786 can directly drive up to 32 DRAM/VRAM chips. One 82786 pin shares two DRAM functions DRA9/$\overline{\text{RAS3}}$. These functions are never both used in the same configuration. DRA9 is only used by 1M x 1 DRAMs, which limit the number of rows to only two due to both addressing (4 Megabytes) and drive (32 chips) limitations.

Figure 8 shows a full connection diagram for thirty-two 64K x 4 DRAMs. Two interleaved banks of four rows each are used. Unlike most DRAM/VRAM controllers, no impedance-matching resistors are usually needed between the 82786 chip and the DRAM/VRAM chips. The impedance-matching for most configurations is handled internally by the 82786. This is also the connections required for x 4 VRAMs which use the $\overline{\text{BEN}}$ signal to control their $\overline{\text{OE}}/\overline{\text{DT}}$ input which is used to determine when to load their internal shift register (Figure 9).

If Static Column DRAMs are used in an interleaved configuration, an external latch is required to latch the column address for the second bank (Figure 8a). The 82786 can directly drive up to thirty-two DRAM devices. For configurations requiring more than thirty-two devices, external buffering must be used.

DRAMs with separate data-in and data-out pins (such as the x 1 DRAMs) require a tristate buffer for the data-out lines of each bank. (All of the rows within each bank may share the same tristate buffer). Figure 10 shows a full connection diagram for thirty-two 256K x 1 DRAMs including the tristate buffers. Two interleaved banks of one row each are used. This is a special case for the RAS lines. Normally $\overline{\text{RAS0}}$ would drive all of the DRAMs in both banks for the one row as in Figure 7. However, because the RAS lines have drive capability for only 16 DRAMs, both $\overline{\text{RAS0}}$ and $\overline{\text{RAS1}}$ are used. The 82786 recognizes this special case and automatically drives $\overline{\text{RAS1}}$ identically to $\overline{\text{RAS0}}$.

The other special DRAM case is using two rows of x 1 DRAMs in a noninterleaved configuration. This configuration has the advantage that only one bank of transceivers is required, but burst access time is reduced by half from the previous example. Normally, $\overline{\text{CAS0}}$ would be used to drive all 32 DRAMs, but because of drive limitations, both $\overline{\text{CAS0}}$ and $\overline{\text{CAS1}}$ are used, (one for each bank). Again the 82786 recognizes this special case and automatically drives $\overline{\text{CAS1}}$ identically to $\overline{\text{CAS0}}$.

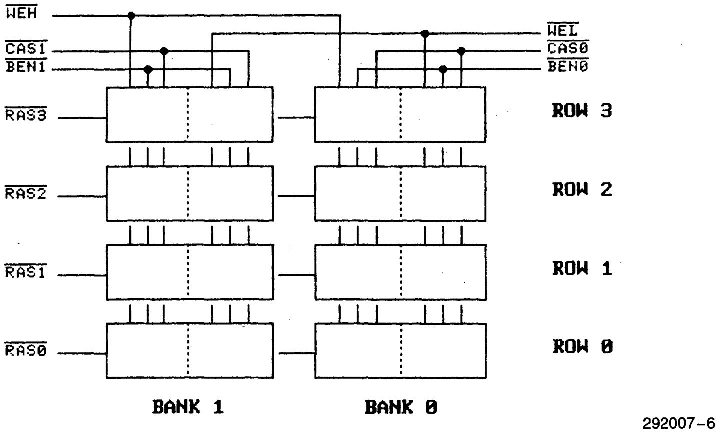
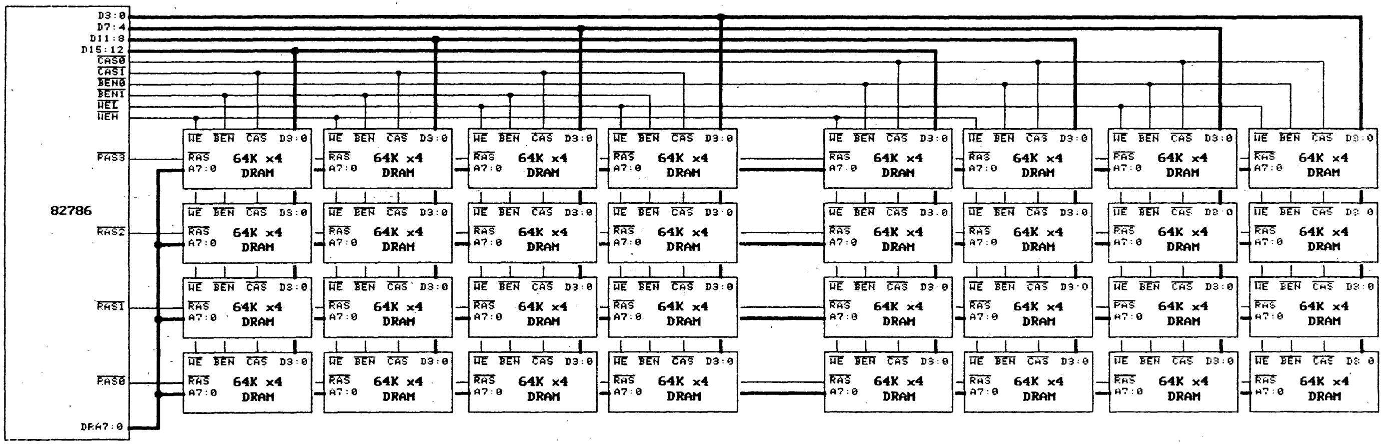
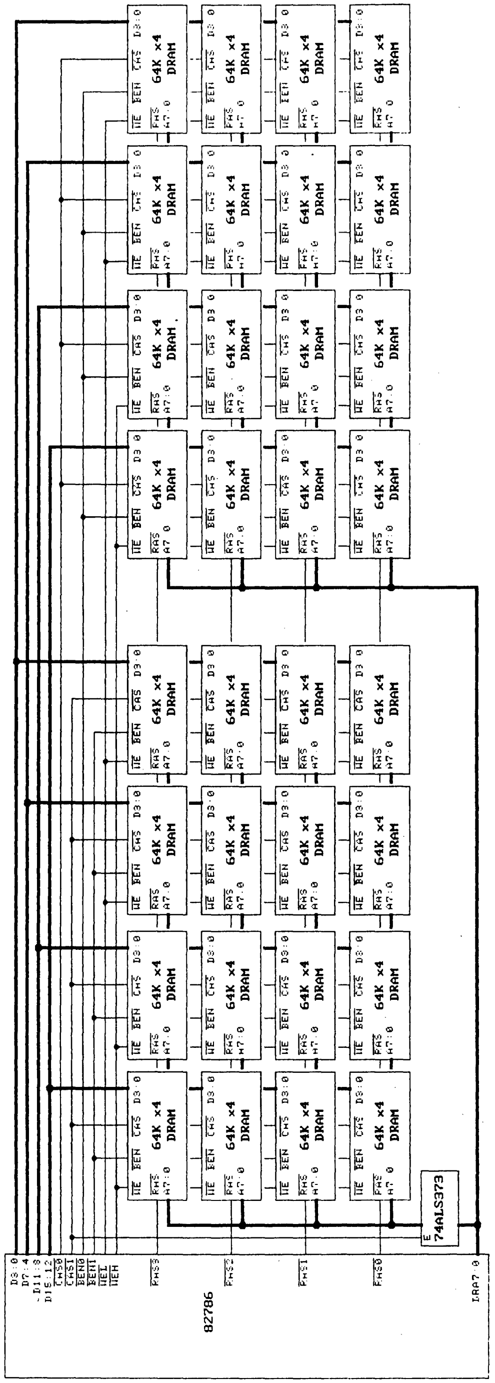


Figure 7. 82786 Supports up to 4 Rows of 2 Interleaved Banks of DRAMs 64K x 4 Video RAMs with 82876 1 Row, 2 Banks, 4 Bits/Pixel



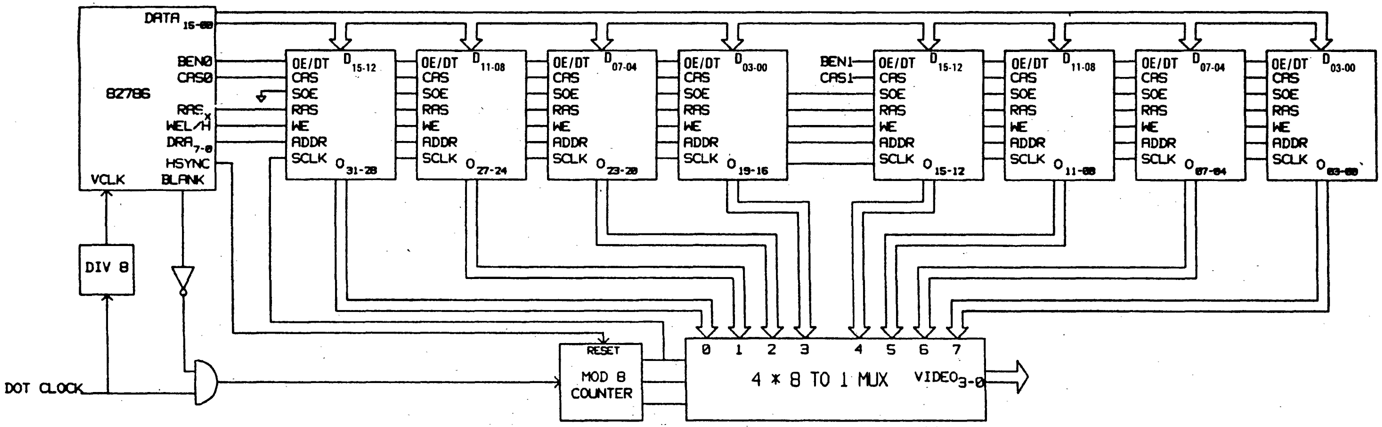
292007-7

Figure 8. 82786 Driving 4 Rows of Two Interleaved Banks of 64K x 4 DRAMS



292007-8

Figure 8a. 82786 Driving 4 Rows of Two Interleaved Banks of 64K x 4 Static Column DRAMs



292007-9

Figure 8b. 64K x 4 Video Rams with 82786 1 Row, 2 Banks, 4 Bits/Pixel

The table in Figure 11 shows all the possible configurations for 64K bit, 256K bit and 1M DRAMs.

3.2 DRAM Timing Parameters

Care should be taken to ensure that all of the timings of the DRAMs used, fit with those in the 82786 data sheet. To make the comparisons easier, the names of the parameter in the 82786 data sheet correspond to the names in most DRAM data sheets. In addition, the parameters have been broken into the same four groups used by most DRAM data sheets.

The critical parameters for page mode DRAMs are generally:

Single rd/wrt/RMW	Single wrt	RMW	Page rd/wrt
Tcac Trp Trcd Trah Tasc Ton	Trwl Tcwl	Tds(rw) Toff	Tds(i)

Some of the 82786 parameters may not be found in all Page Mode data sheets. If no corresponding DRAM parameters for Tcaa or Tcar are specified, then the 82786 spec may be ignored. The reason is that, if no such DRAM parameters exists, then the resulting minimum values for these parameters are at most:

$$Tcaa = Tasc + Tcac$$
$$Tcar = Tasc + Trsh$$

Then as long as the Tasc, Tcac, and Trsh specs fit, the 82786 timings guarantee Tcaa and Tcar to fit.

A third parameter that may not be found in all Page Mode data sheets is Ton. If x 1 DRAMs are used, the external data transceiver is responsible for meeting this and the DRAM is not required to meet this spec. If, however, x 4 or x 8 DRAMs are used, without the data transceiver, care must be taken to ensure that this spec is met.

The critical parameters for Fast Page Mode and Static Column DRAMs are generally:

Single rd/wrt/RMW	Single wrt	RMW	Fast-page-mode rd/wrt

Trp
Trah
Tasc
	Trwl
	Tcwl
		Tds(rw)
		Toff
			Tcp
			Tcaa
			Tcap
			Tds(n)
			Tcah(i)
			Tds(i)
			Tdh(i)
			Ton(ri)

For interleaved Static Column DRAMs, the address latch delay must be added to the DRAM parameters corresponding to the row and column addresses. These parameters are:

— Tasr
— Tasc
— Tcaa

For all types of x1 DRAMs, Page Mode, Fast Page Mode and Static Column, the transceiver delay must be added to the DRAM parameters which correspond to read-data. These parameters are:

— Trac
— Tcac
— Tcaa

Notice that all of the 82786 DRAM timings are specified relative to the bus clock (CLK). This has two implications. First, a slower bus clock can be used to allow the 82786 to use slower DRAMs. Secondly, many of the parameters are determined by the duty cycle of the bus clock (as their specification is dependent on clock high or low time). A slightly nonsymmetric clock, such as the clock for the 80286, can be used for the 82786 CLK, but care should be taken to examine the effects on the DRAM timing. In some circumstances, it may be advantageous to use a slightly nonsymmetric clock.

Some of the specifications are relative to the 82786 clock period (Tc), while others are relative to a specific phase (THigh, TLow).

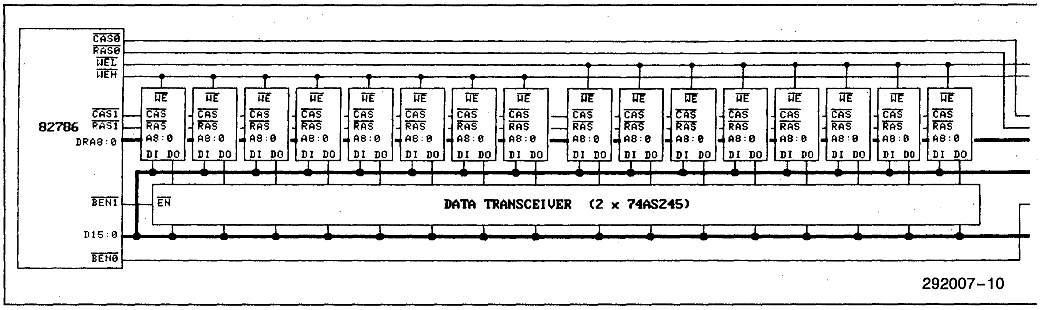


Figure 10. Two Interleaved Banks of 256K x 1 DRAMs

	Non-Interleaved				Interleaved			
	1-row	2-rows	3-rows	4-rows	1-row	2-rows	3-rows	4-rows
64K x1	128K 16	256K 32	384K 48*	512K 64*	256K 32	512K 64*	768K 96*	1024K 128*
16K x4	32K 4	64K 8	96K 12	128K 16	64K 8	128K 16	192K 24	256K 32
8K x8	16K 2	32K 4	48K 6	64K 8	32K 4	64K 8	96K 12	128K 16
256K x 1	512K 16	1024K 32	1536K 48*	2048K 64*	1024K 32	2048K 64*	3072K 96*	4096K 128*
64K x4	128K 4	256K 8	384K 12	512K 16	256K 8	512K 16	768K 24	1M 32
32K x8	64K 2	128K 4	192K 6	256K 8	128K 4	256K 8	384K 12	512K 16
1M x1	2M 16	4M 32	—	—	4M 32	—	—	—
256K x4	512K 4	1M 8	1.5M 12	2M 16	1M 8	2M 16	3M 24	4M 32
128K x8	256K 2	512K 4	768K 6	1M 8	512K 4	1M 8	1.5M 12	2M 16

*Requires external buffering

Figure 11. Possible DRAM configurations for 64K, 256K and 1 Mbit DRAMs. The top number in each box is total memory size in bytes, the bottom is the number of DRAM chips required.

Look at this example. Suppose you use 51C256H Fast-page-mode DRAMs with the 82786 as in Figure 10. First, look at the critical parameters shown above. Since it is not possible to create a precisely 50% duty cycle clock, you must consider clocks with a few percent tolerance. The table compares the 82786 using several clock frequencies and duty cycle tolerances with two versions of the 51C256H. The table is ordered with the tightest timings first.

From the table, you can see that the fast 120 ns access DRAMs can be used with the 82786 with a 10 MHz clock with as much as a 40%–60% duty cycle skew. The slower DRAMs can be used at 9 MHz with a tighter 45%–55% duty cycle skew or at 8 MHz with a 40%–60% skew.

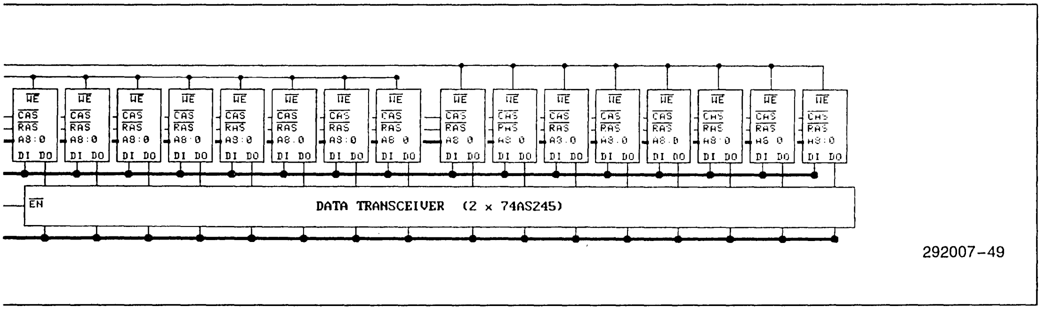


Figure 10. Two Interleaved Banks of 256K x 1 DRAMs (Continued)

Parameter			82786 Specifications				51C256H DRAM Specs	
			10 MHz 45–55%	10 MHz 40–60%	9 MHz 45–55%	8 MHz 40–60%	−12 120 ns	−15 150 ns
Tdh	Min	Tph	22.5	20	25	25	20	25
Toff	Max	T1 + 3	25.5	23	28	28	20	25
Tcah	Min	Tch + 2	26.5	22	24.5	27	15	20
Tcp	Min	Tcl − 5	17.5	15	20	20	10	10
Tds	Min	Tcl − 8	14.5	12	17	17	0	0
Tcaa	Max	2Tc − 27	73	73	83	98	55	70
Tcap	Max	2Tc − 21	79	79	89	104	60	75
Tasc	Min	Tcl − 5	17.5	15	17.5	17.5	5	5
Trp	Min	2Tc − 5	95	95	105	120	70	85
Trwl	Min	Tc − 9	41	41	46	53.5	25	30
Tcwl	Min	Tc − 12	38	38	43	50.5	25	30
Trah	Min	Tc + 3	53	53	58	65.5	15	20
Ton	Max	Tc − 24	26	26	31	38.5	25	30

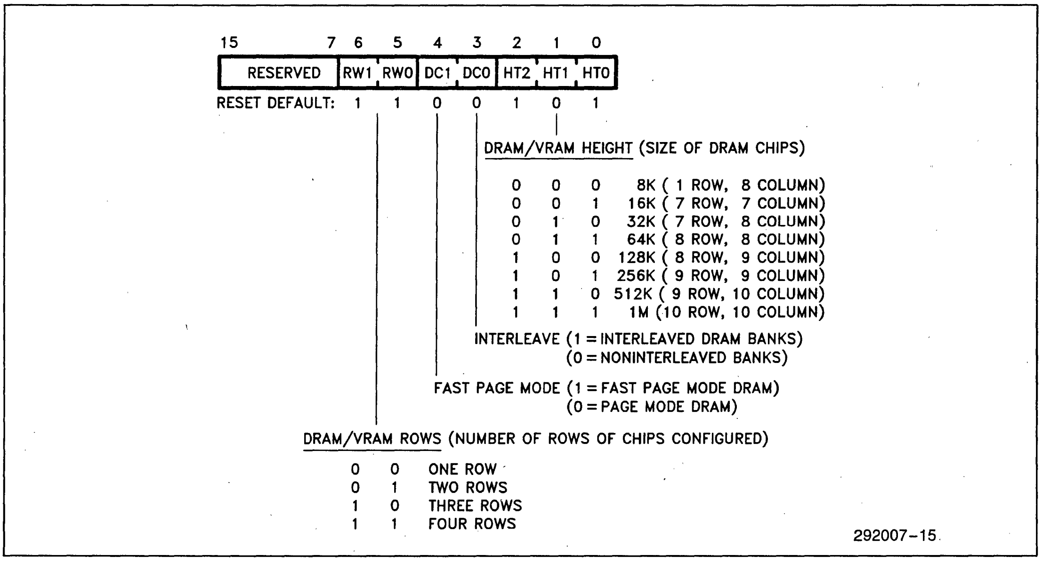
Because these x 1 DRAMs require transceivers between their data outputs and the 82786, the transceiver delays must also be considered. The two parameters in the table above, that are affected are Tcaa and Tcap. The transceiver delay must be added to the DRAM access time for these parameters. This implies that the data-in to data-out time of the transceivers must be 18 ns or less for the 10 MHz −120 ns case and the 8 MHz −150 ns case. The delay must be 28 ns or less for the 9 MHz −150 ns case and the 8 MHz −150 ns case.

3.3 Initializing the DRAM Controller

Two of the 82786 Internal Registers are used to configure the DRAM/VRAM Controller. Both of the registers are typically set once during initialization and then never changed. The DRAM/VRAM Control Register is set to indicate the configuration of the DRAMs/VRAMs used. The DRAM/VRAM Refresh Control Register is set to indicate the frequency of refresh cycles. Once programmed, the settings can be write-protected using the write-protect bits discussed in Section 4.2.

It is recommended that all fields of the DRAM/VRAM Control Register be written simultaneously to avoid illegal combinations. Also, no DRAM accesses should be attempted until the DRAM/VRAM Control Register has been set. For the configuration in Figure 10 using one row of 256K Fast Page Mode DRAMs in two interleaved banks:

DRAM/VRAM Control - Internal Register Offset 08h


15 7 6 5 4 3 2 1 0
RESERVED RW1 RW0 DC1 DC0 HT2 HT1 HT0
RESET DEFAULT: 1 1 0 0 1 0 1

DRAM/VRAM HEIGHT (SIZE OF DRAM CHIPS)

0	0	0	8K (1 ROW, 8 COLUMN)
0	0	1	16K (7 ROW, 7 COLUMN)
0	1	0	32K (7 ROW, 8 COLUMN)
0	1	1	64K (8 ROW, 8 COLUMN)
1	0	0	128K (8 ROW, 9 COLUMN)
1	0	1	256K (9 ROW, 9 COLUMN)
1	1	0	512K (9 ROW, 10 COLUMN)
1	1	1	1M (10 ROW, 10 COLUMN)

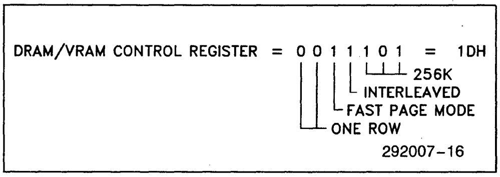
INTERLEAVE (1 = INTERLEAVED DRAM BANKS)
(0 = NONINTERLEAVED BANKS)

FAST PAGE MODE (1 = FAST PAGE MODE DRAM)
(0 = PAGE MODE DRAM)

DRAM/VRAM ROWS (NUMBER OF ROWS OF CHIPS CONFIGURED)

0	0	ONE ROW
0	1	TWO ROWS
1	0	THREE ROWS
1	1	FOUR ROWS

292007-15



292007-16

DRAM/VRAM Refresh Control, Internal Register Offset 06H, is set to 18 as shown below.

15	6	5	4	3	2	1	0
Reserved	Refresh Scalar						
RESET Default:	0	1	0	0	1	0	

The 82786 CLK input is internally divided by 16 and then divided by the Refresh Scalar + 1 in the DRAM/VRAM Refresh Control Register to determine the time between refresh cycles. Only one row of each DRAM/VRAM is refreshed at a time so refresh of the entire DRAM/VRAM requires 128, 256, 512 or 1024 of these refresh cycles depending on the number of rows in the DRAM/VRAM.

For example, the 51C256H DRAMs require a complete refresh every 4 ms (Tref). These DRAMs consist of 512 address rows of 512 address columns. However, for refresh purposes, only 256 row addresses (A0–A7) need to be refreshed within the 4 ms refresh time. The A8 input is not used for refresh cycles. (The 82786 maintains a full 10-bit refresh address, the upper 2 bits are simply not used in this configuration). Assuming a 10 MHz 82786 CLK, we can determine the value for the DRAM/VRAM Refresh Control register as follows:

$$\text{Refresh Count} = \frac{\text{Tref} \times \text{CLK}}{16 \times \text{Refresh\_Rows}} - 1$$

$$= \frac{4\text{ ms} \times 20\text{ MHz}}{16 \times 256} - 1 = 18.53$$

The result should always be rounded down, so the DRAM/VRAM Refresh Control Register should be programmed with 18. This result is dependent only on the DRAM/VRAM type and the 82786 CLK frequency. The configuration of the DRAM/VRAM chips does not matter.

There is a latency time between the refresh request generated by this count and the actual refresh cycle. The refresh will always occur as soon as the current bus cycle finishes. Refresh cycles can interrupt block transfers, but only at double-word boundaries. The worst case is if a refresh request occurs just after the 82786 receives HLDA to begin a master mode block transfer. The 82786 must complete two master cycles before the refresh cycles can be performed. During this latency, further refresh requests may be generated. The 82786 contains a refresh request queue that allows up to three refresh requests to be pending. As soon as the bus is freed, all queued refresh cycles will be run consecutively.

For the above example, refresh requests are generated every 15.2 μs which is derived using the following formula.

$$\text{Request\_time} = \frac{16 \times (\text{RefreshCount} + 1)}{\text{CLK}}$$

$$= \frac{16 \times (18 + 1)}{20\ \text{MHz}} = 15.2\ \mu\text{s}$$

The amount of latency that the DRAMs will tolerate for each row is:

$$\text{Allowed\_Latency} = \text{Tref} - (\text{RequestTime} \times \text{Refresh\_Rows})$$
$$= 4\ \text{ms} - (15.2\ \mu\text{s} \times 256) = 108.8\ \mu\text{s}$$

But the real latency limit is that the 82786 allows only three requests to be queued:

$$\text{Maximum\_Latency} = \text{Queue\_Size} \times \text{Refresh\_Time}$$
$$= 3 \times 15.2 = 45.6\ \mu\text{s}$$

Therefore, the maximum number of wait-states allowed for a 82786 master mode transfer is:

$$\text{Wait\_States} = ((\text{Maximum\_Latency} \times \text{PCLK}) - \text{overhead})/\text{bus-cycles}$$
$$((45.6\ \mu\text{s} \times 10\ \text{MHz}) - 7\ \text{cycles})/2 = 224$$

Clearly, in this situation, refresh latency is not a problem. If the system memory caused the 82786 to delay over 224 wait-states for a master-mode access, not only would DRAM/VRAM refresh be missed, but the display refresh would also be lost.

The 82786 always issues three refresh cycles following a RESET. Besides these first three refresh cycles, the 82786 does not perform any other DRAM/VRAM initialization after cold or warm-reset. If the DRAMs/VRAMs require other initialization cycles, the CPU should either perform dummy cycles to the DRAM/VRAM or wait until the refresh counter has requested enough refresh cycles to occur.

If the DRAM/VRAM Refresh Control Register is set to all ones, refresh cycles are disabled.

4.0 SYSTEM BUS INTERFACE

The 82786 system bus structure allows the 82786 to be easily connected to a variety of CPUs. The 82786 can act as both a slave and a master to the CPU's bus. As a slave, the CPU or DMA can perform read and write cycles to the 82786 Internal Registers or to the 82786 DRAM/VRAM. As a master, the 82786 Graphics and Display Processors can perform read and write cycles to the CPU's system memory.

The 82786 bus can operate in three different modes to handle various CPU interfaces. The 82786 determines which mode to use by sampling the $\overline{\text{BHE}}$ and MIO pins during RESET:

	$\overline{\text{BHE}}$	MIO
Synchronous 80286 bus	1	0
Synchronous 80186 bus	1	1
Asynchronous bus	0	X

For synchronous 80286 interfaces, the Reset and Clock inputs into the 80286 and 82786 must be common. For synchronous interfaces to 80186, the 80186 CLKIN must be the same as the 82786 CLK (so an external clock source must be used). The $\overline{\text{RES}}$ input into the 80186 must meet a set up and hold time with respect to the CLKIN. The RESET for the 82786 should be generated from the $\overline{\text{RES}}$ (for 80186) by delaying $\overline{\text{RES}}$ by one CLKIN cycle and inverting it. This ensures that the 82786 ph1 is coincident with 80186 CLKOUT low.

These pin states are easy to achieve for the synchronous modes. During RESET, the 80286 always drives $\overline{\text{BHE}}$ high and MIO low.

CPUs with timings different from the 80286 must use asynchronous mode (however, CPUs such as the 80386 can easily generate 80286 style timings). Care should be taken in this case to ensure $\overline{\text{BHE}}$ is low during RESET.

In each of these three modes it is possible to configure the 82786 to allow both master and slave accesses or to simplify the logic to allow only slave access. In master mode, the 82786 always generates 80286 style bus signals.

If the 82786 is used as a master, it will activate its HREQ line when it needs to become the bus master to access system memory. It waits until HLDA is received and then begins driving the system bus. Once HLDA is received, a 10 MHz 82786 can perform system bus accesses at the following rate (assuming 0 wait-states).

single reads/writes	4 cycles	5 Mbyte/sec
read-modify-writes	6 cycles	3.3 Mbyte/sec
burst-access read/write	2 cycles	10 Mbyte/sec

The 82786 will begin the first master bus access on the cycle after HLDA is activated. The only delay is the time between when the 82786 activates HREQ and the system can release the bus and return HLDA. Most synchronous CPUs require a minimum of three cycles between the time HOLD is activated until they can return HLDA. The 82786 will keep HREQ activated until it no longer has more accesses to perform to system memory (until either the next 82786 access is to the dedicated graphics DRAM/VRAM or until neither the Graphics or Display Processors require the bus.). Once the 82786 is done using the system bus, it will remove HREQ and is able to immediately access its Graphics DRAM/VRAM on the next cycle.

It is potentially possible for the 82786 to require the system bus for a lengthy period of time. For example, if the 82786 has been programmed to give the Graphics Processor high priority, and the Graphics Processor executes a command that requires a lot of access to system memory, then the system bus could potentially be held by the 82786 for several consecutive accesses. Drawing a long vector into a bitmap residing in system memory is such a command. In this case, the maximum time the 82786 can potentially keep the system bus is determined by the frequency of DRAM/VRAM refresh cycles programmed into DRAM/VRAM Refresh Control Register.

If the CPU needs to regain control of the bus before the 82786 is done, it may remove HLDA early. The 82786 will then complete the current access and remove HREQ to indicate to the CPU that it may now takeover control of the bus. If the 82786 still requires more access to the system bus, it will re-activate HREQ two cycles after it had removed it and wait until the next HLDA. Since the 82786 removes HREQ for only two cycles, it is important that the CPU recognize it immediately. Otherwise a lock-out condition will occur in which the CPU is waiting for the 82786 to remove HREQ and the 82786 is waiting for the CPU to issue HLDA. This is not a problem for the synchronous interfaces. Extra logic may be required to prevent this situation if the 82786 is used as a master in an asynchronous interface and HLDA is ever removed prematurely, especially if the CPU clock is significantly slower than the 82786 clock.

4.1 Memory Map

Figure 12 shows the memory map as it appears to both the 82786 Graphics and Display Processors. These processors both use a 22-bit address which provides for up to 4 Megabytes of address space. They are only allowed to make memory accesses so no I/O map is applicable.

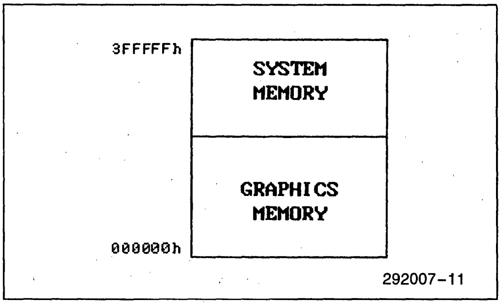


Figure 12. Memory Map for Graphics and Display Processors

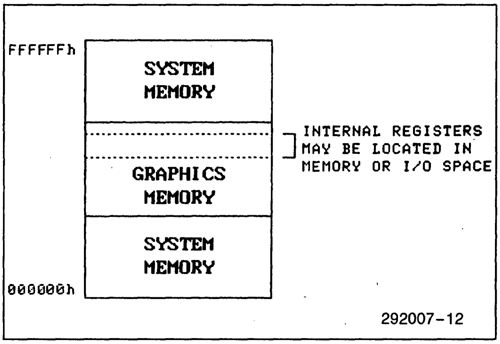


Figure 13. Memory Map for System CPU

The 82786 dedicated graphics DRAM/VRAM always starts at location 000000h and grows upwards. The upper address depends on the amount of DRAM/VRAM memory configured in the DRAM/VRAM Control Register. The system bus memory begins where the DRAM/VRAM ends and continues to the highest addressable memory location 3FFFFFh.

The memory map as it appears to the system CPU is shown in Figure 13. The area that the 82786 Graphics DRAM/VRAM is mapped into can be anywhere in the CPU address space and is completely defined by the address decode logic of the CPU system. Normally only the space for the configured graphics memory is mapped into CPU address space. If addresses above the configured graphics memory are mapped into the CPU address space, and the CPU writes to addresses above the configured 82786 memory, the write is ignored. If it reads from these locations, the data returned is undefined.

The 82786 Internal Registers may be configured to reside in memory or I/O address space. If configured to reside in memory, then they will override a 128 byte area of the 82786 memory address space for external (CPU) accesses. The Internal Registers are only accessible by the external CPU and therefore are never found in the 82786 Graphics or Display Processor memory maps.

Suppose the 82786 is configured with 1 Megabyte of Graphics DRAM/VRAM and is used in an 80286 system. A possible memory map and connection diagram is shown in Figure 14. All of the 82786 memory is mapped into the 80286 address space. Also, a 3 Megabyte portion of the 80286 system memory is mapped into the 82786. Since the 80286 has two more address bits than the 82786, a tristate buffer is used to supply the top two address bits when 82786 enters master mode.

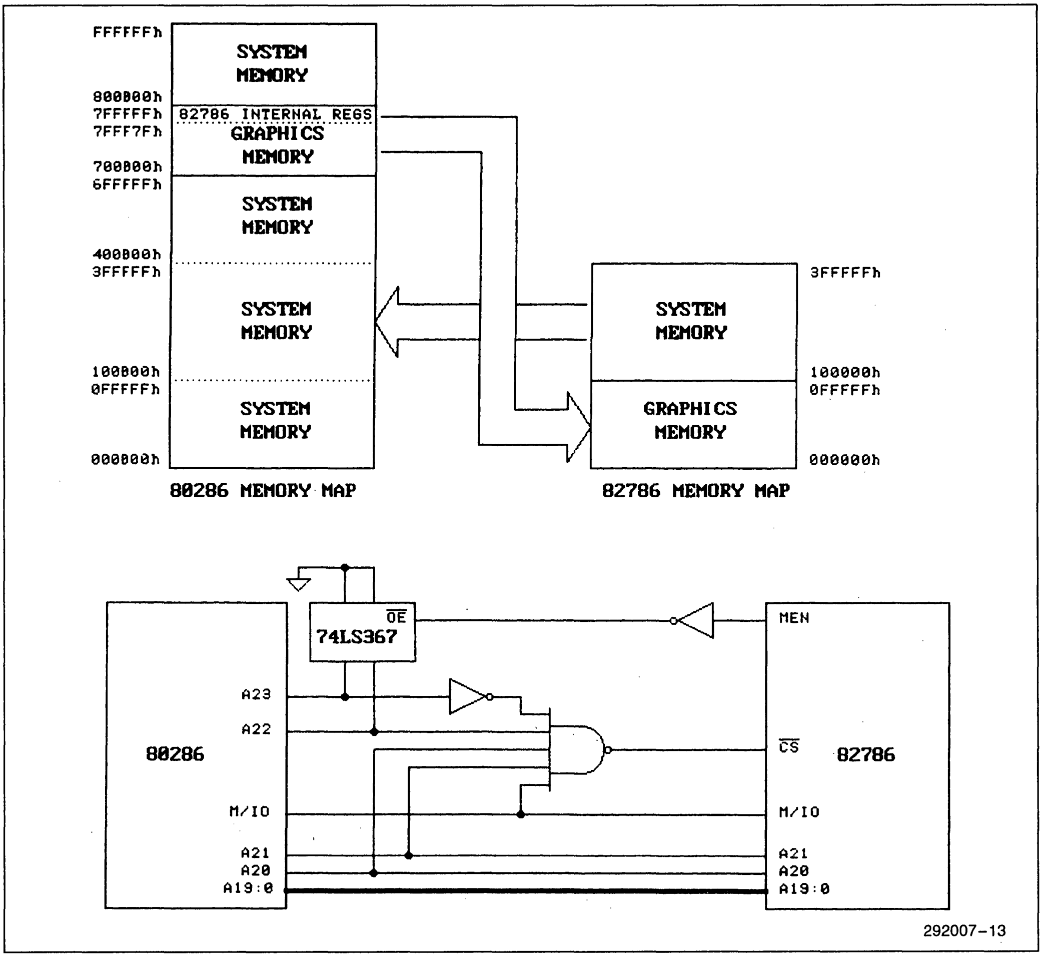


Figure 14. Possible Memory Mapping for 80286/82786 82786 Internal Registers are Memory Mapped

Notice that the same memory corresponds to one set of memory addresses for the CPU and a different set of memory address for the 82786 Graphics and Display Processors. Although it is possible to make these addresses match, it is not necessary as long as the controlling CPU software understands the relationship and makes the simple conversion. Often it is not desirable to make the addresses match. For example, most CPUs use the lowest memory addresses for special purposes, such as for interrupt vectors. If the lowest CPU memory were 82786 memory rather than the faster (for CPU access) system memory, then these operations would execute significantly slower.

Even though the real addresses don't match, the operating system for a CPU such as the 80286 could make the CPUs virtual addresses map easily to the 82786 real addresses.

The 82786 Internal Registers may either be memory or I/O mapped. If they are memory mapped over the Graphics DRAM/VRAM, the CPU will not be able to access the 128 bytes of DRAM/VRAM which they cover, (although the Graphics and Display Processors can). If they are memory mapped above the Graphics DRAM/VRAM (over nonconfigured memory), then they will not prevent the CPU from accessing any of the 82786 memory, but they must be included in the CPU memory space that the address decoder allocates for the 82786. The 82786 Internal Registers may be I/O mapped, so they do not overlap any memory, however the CPU chip select logic for the 82786 becomes slightly larger. Figure 15 shows a circuit similar to Figure 14, except the registers are I/O mapped. Memory mapping the Internal Registers allows the software slightly more flexibility in accessing the registers.

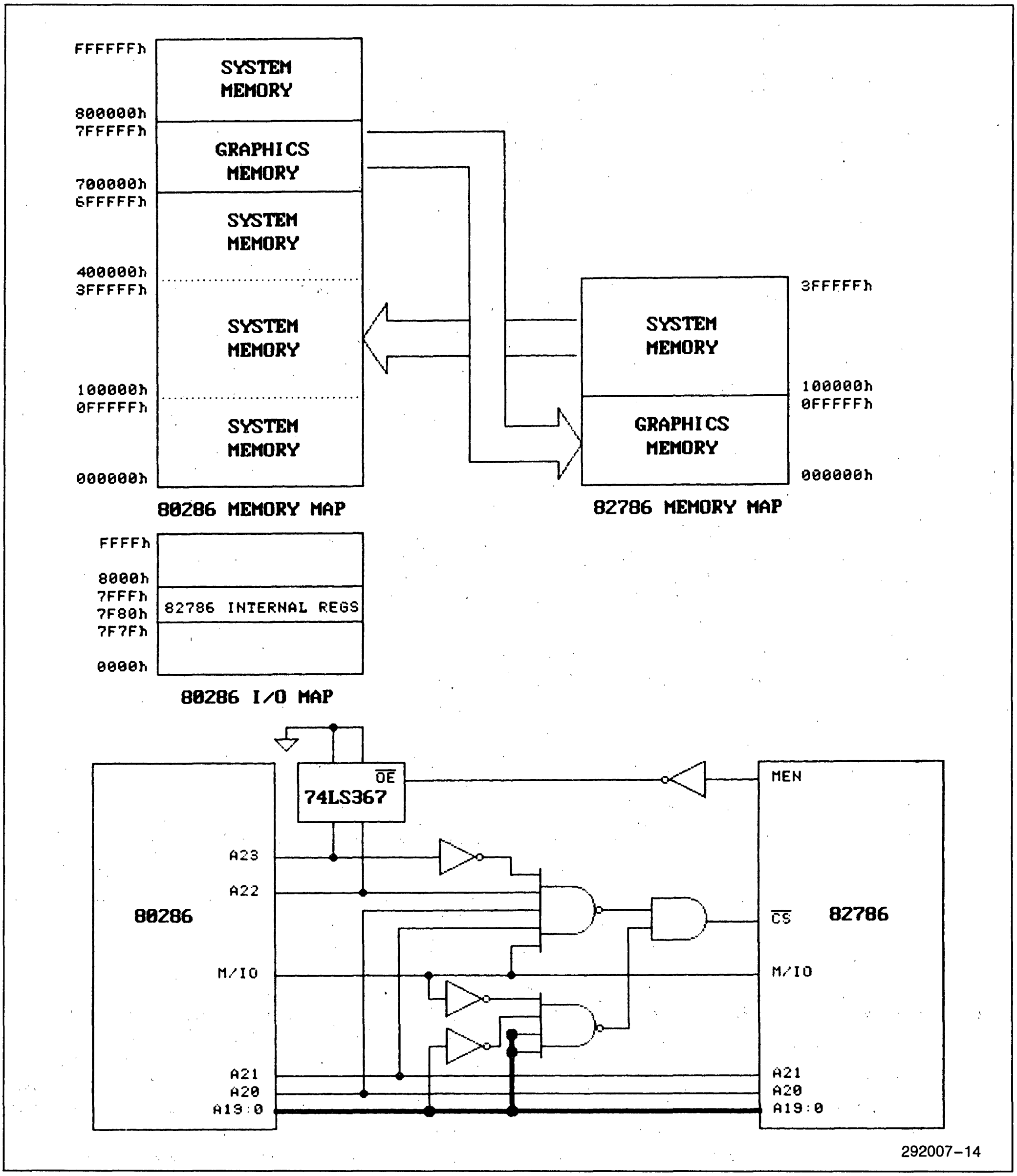


Figure 15. Possible Memory Mapping for 80286/82786
82786 Internal Registers are I/O Mapped

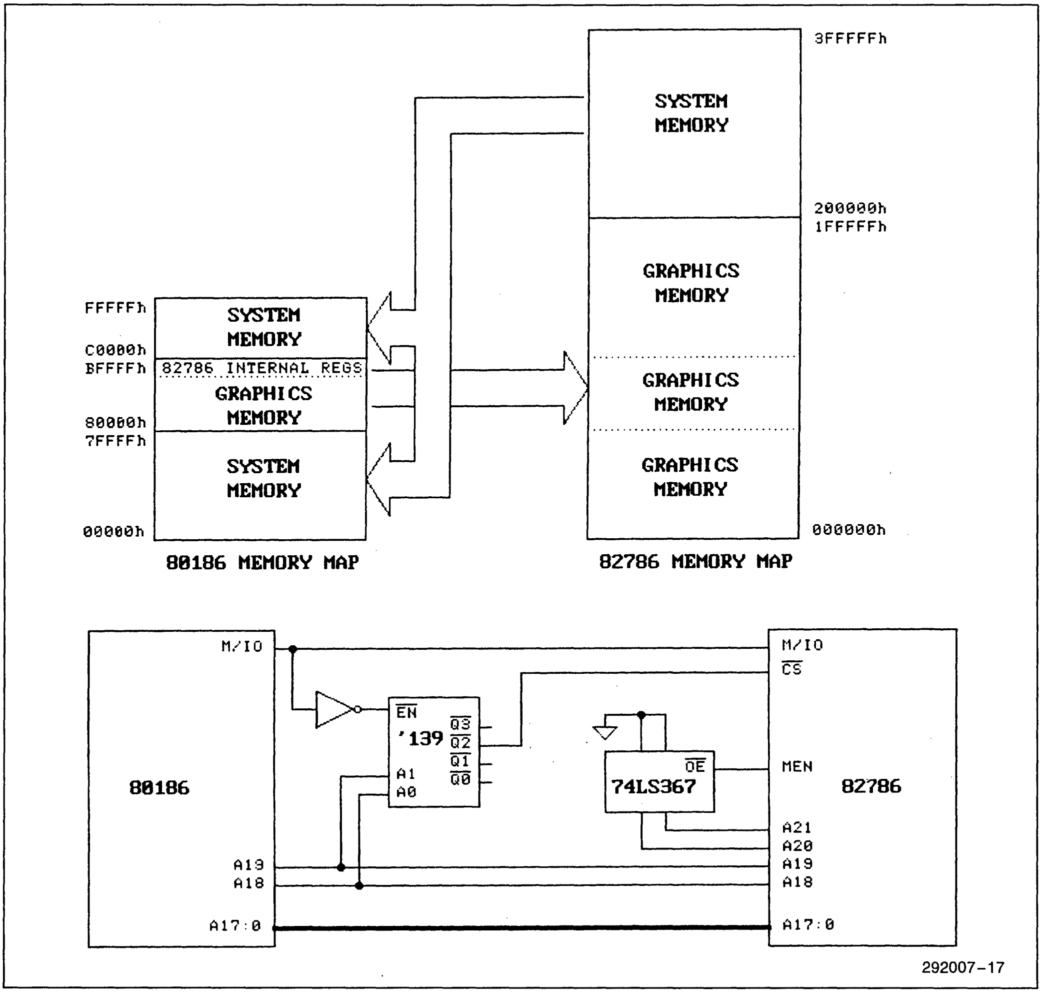


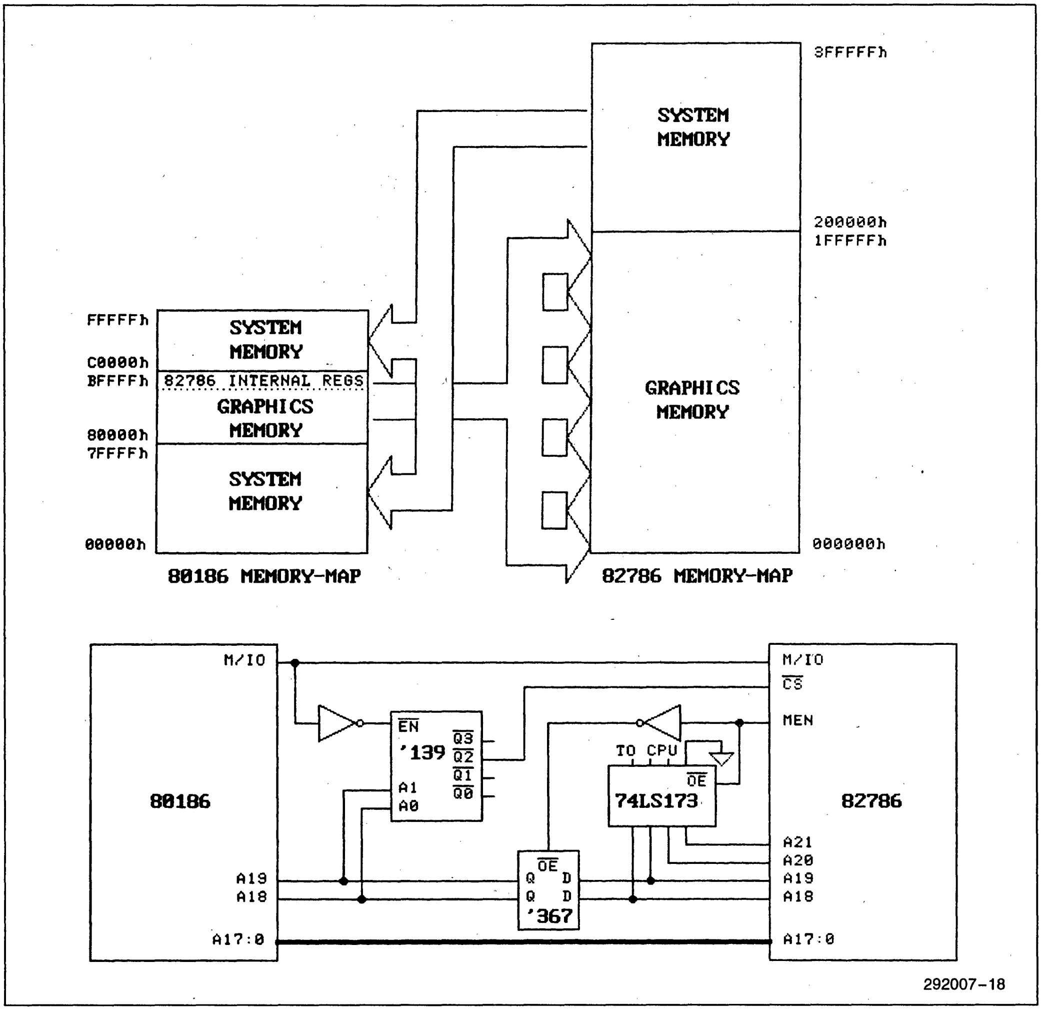
Figure 16. Possible Memory Mapping for 80186/82786

Because graphics memory can be quite large, some system designs might not allow all of the configured Graphics DRAM/VRAM to be directly mapped by the CPU. For example, if the 82786 has 2 Megabytes of Graphics DRAM/VRAM and is used with a 80186 processor, which can only address 1 Megabyte, then the 80186 can not directly access all of the 82786 memory. In this case the CPU can be permitted to only access a portion of the graphics DRAM/VRAM. Figure 16 shows a memory map and connection diagram for such a system. Since the 82786 has two more address bits than the 80186, a tristate buffer is used to supply the two highest address bits when the 82786 is in slave mode.

In many cases the CPU does not require access to all of the graphics memory. For example, many situations will not require the CPU to directly access the bitmaps. If the CPU must gain access to the graphics memory which is not directly mapped to the CPU, the 82786 Graphics Processor can be instructed (using the BitBlt command) to move portions of the Graphics memory to and from the area accessible by the CPU.

Alternatively, the Graphics DRAM/VRAM areas can be bank switched to allow the CPU direct access at any portion of the graphics memory. Figure 17 shows the use of an I/O port (74LS173 latch) to which the CPU can write the highest 3 bits of the address for the 82786 slave accesses.

In both Figures 16 and 17, it is possible for the 82786 in master mode to access the CPU memory addresses that



292007-18

Figure 17. Possible Memory Mapping for 80186/82786 Bank-Switching Allows 80186 to Access All 82786 Memory

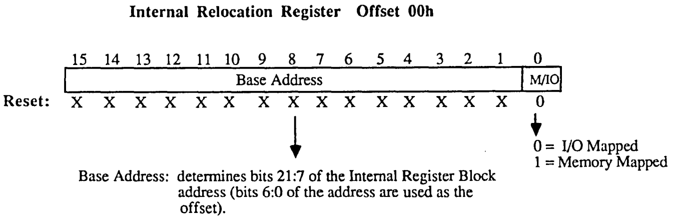
correspond to the 82786 slave addresses. In this case, the circuit will generate a 82786 chip-select, but the 82786 will not respond to this chip-select while it remains in master mode. As long as the READY logic goes high (it may not since the 82786 will not perform the slave-access) then the 82786 will complete the master-mode cycle. By the time the 82786 returns to slave-mode, the chip-select will have gone away.

4.2 BIU Registers

Within the 82786 Internal Register block, the registers at offsets 00h–0Fh are used by the Bus Interface Unit to control the system configuration (Figure 4). These registers are normally set once during power-up intialization and never changed.

Two of these registers, DRAM/VRAM Refresh Control and DRAM/VRAM Control have already been discussed in Section 3.3. The rest of the registers are discussed in this section.

The Internal Relocation Register defines the location of the 82786 Internal Registers anywhere in the 82786 memory or I/O address space.


Internal Relocation Register Offset 00h
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Base Address
M/IO
Reset: X X X X X X X X X X X X X X X 0
0 = I/O Mapped
1 = Memory Mapped
Base Address: determines bits 21:7 of the Internal Register Block address (bits 6:0 of the address are used as the offset).

292007-B6

After RESET, any CPU slave I/O address to the 82786 (which activates the 82786 Chip-Select) will access the Internal Register Block. During initialization, a write to the Internal Relocation Register should be performed to locate the Internal Register Block at specific even byte memory or I/O address. Once the write to the Internal Relocation Register occurs, the 82786 Internal Register Block no longer occupies all of 82786 I/O space, rather it is restricted to just the 128 memory or I/O bytes specified. The Internal Registers can be located anywhere accessible by the CPU. However, if they are memory-mapped and located over configured graphics memory, they will take precedence over the memory for CPU accesses to those addresses. Graphics or Display Processor accesses to these addresses will still be directed to DRAM/VRAM. For example, writing the value of 03F8h locates the Internal Registers at I/O addresses FE00h – FE7Fh.

```
03F8h = 00 0000 1111 1110 0   0
                        |     |
                        |   I/O mapped
        Base Address 00FE00h(offsets 0-7Fh)
```

Note that the address written to the Internal Relocation Register determines the memory or I/O address that is required to be placed on the 82786 address pins during a CPU access to the 82786 Internal Registers. The actual CPU address used may be different, and is dependent on the chip select and memory mapping logic described in Section 4.1.

There are four sources of requests for the 82786 bus:

— DRAM/VRAM refresh
— Display Processor
— Graphics Processor
— External Processor (CPU or DMA slave accesses)

The DRAM/VRAM refresh requests are always top priority. That is, once the DRAM/VRAM refresh request is made, the 82786 bus will complete the current bus access and then perform the DRAM/VRAM refresh. Three BIU registers are used to set the priorities of the other three bus requests. Two priority values are used:

FPL - First Priority Level - priority used when processor first requests bus.

SPL - Subsequent Priority Level - priority used for processor to maintain bus during a block transfer. If a block transfer is interrupted, this is also the priority used to regain the bus to complete the burst access.

When a processor first requests the 82786 bus, its FPL value is used. The processor with the highest priority gets access to the bus. Once the bus is granted, the first access occurs. If a multiple-word block transfer is performed the SPL value is then used as the priority to maintain the bus for subsequent cycles. As long as no other processor of higher priority requests the bus, the burst-access is allowed to continue to completion. If a higher priority request is made, the block transfer will be suspended and the bus granted to the new request. The suspended block transfer will not get the bus back until its SPL value is again the highest priority request.

A separate register is used to program the priority for each of the three processors. Because the External Processor can not perform block transfers, no External SPL value is required for it.

Display Priority - Internal Register Offset 0Ah

15	6	5	4	3	2	1	0
Reserved		FPL			SPL		
RESET Default:		1	1	0	0	1	1

Graphics Priority - Internal Register Offset 0Ch

15	6	5	4	3	2	1	0
Reserved		FPL			SPL		
RESET Default:		1	0	1	0	1	0

External Priority - Internal Register Offset 0Eh

15	6	5	4	3	2	1	0
Reserved		FPL			Reserved		
RESET Default:		1	1	1			

All of the priorities are programmable values from 0 to 7, with 7 being the highest priority. If two processors that are programmed with the same priority both request the bus, the priority in which the bus will be granted for the two will be (from highest to lowest):

— Display Processor
— Graphics Processor
— External Processor

There are two exceptions to these programmable priorities. If the CPU makes a slave request while one of the 82786 processors makes a master request, the CPU's request will always be handled first by the 82786 regardless of priority settings. This is necessary to prevent the lock-out situation where the CPU will not grant HLDA until it completes the bus access to the 82786 and the 82786 will not complete the CPU bus cycle until the higher priority master cycle completes. The other exception is refresh cycles; they always will be handled while the 82786 is in a HLDA loop.

The values programmed into these priority registers should be selected carefully. There is a performance penalty whenever a block transfer is interrupted. However, if block transfers are not interrupted, then it is possible that one processor must wait a long time to get the bus while another is finishing. A balance between overall bus performance and maximum tolerable latency must be made.

For example, if the Display Processor is not given high enough priority, it may not always be able to fetch the bitmapped display data fast enough to keep up with the CRT. When this happens, the Display Processor will not be able to send the correct video data to the CRT and will instead place the value in the FldColor Register on the VDATA pins. To prevent this, the Display Processor can be programmed for the highest priority (after DRAM/VRAM refresh).

The Display Processor internally contains a FIFO which is used to buffer the bitmap data to be displayed. The FIFO consists of 32-double-words of 32 bits each. Each FIFO double-word contains the results of a 32-bit fetch from the bitmap memory. A double-word can therefore contain as many as 32 pixels, or as few as 1 pixel (such as at window borders).

Display Control Processor Register Block 2 TripPt Register controls when this FIFO is loaded. If the TripPt value is set at 16, the Display Processor waits until the FIFO is half empty (only 16 double-words left) before it requests a new block transfer to refill the FIFO. The block transfer request will not end until the FIFO is again full (although the block transfer may be interrupted by a higher priority request). If the TripPt value is set at 20, the Display Processor will begin requesting a new block transfer after only 12 FIFO double-words are emptied (20 left remaining). A low TripPt value generates fewer but longer block transfers and therefore the overall Display Processor bus efficiency is increased. However, a low TripPt value also requires that the bus latency be smaller. A low TripPt value means that there are less double-words left in the FIFO when the bus request is made. If the FIFO drains completely before the bus has been granted, then the FldColor Register value will be used from the current pixel through the end of the current scan line. The TripPt may be programmed to 16 or 20 using the Display Processor LD_REG or LD_ALL commands.

The Display Processor also keeps busy during Blank times. During Vertical Blank time it performs any command loaded into its Opcode Register. During Horizontal Blank time it loads a new Strip Descriptor if necessary and begins fetching the first pixels on the line. The descriptor fetch begins as soon as the last pixel of the last line has been placed in the FIFO. If the Display Processor priority is not high enough to allow these fetches during Blank time, then again part of the display can not be generated correctly and FldColor will be used. Two bits in the Display Processor Status Register can be used to determine if the Display Processor ever gets behind:

bit-5 – DOV - Descriptor Overrun - set if strip descriptor fetch does not complete by the time horizontal blanking ends.

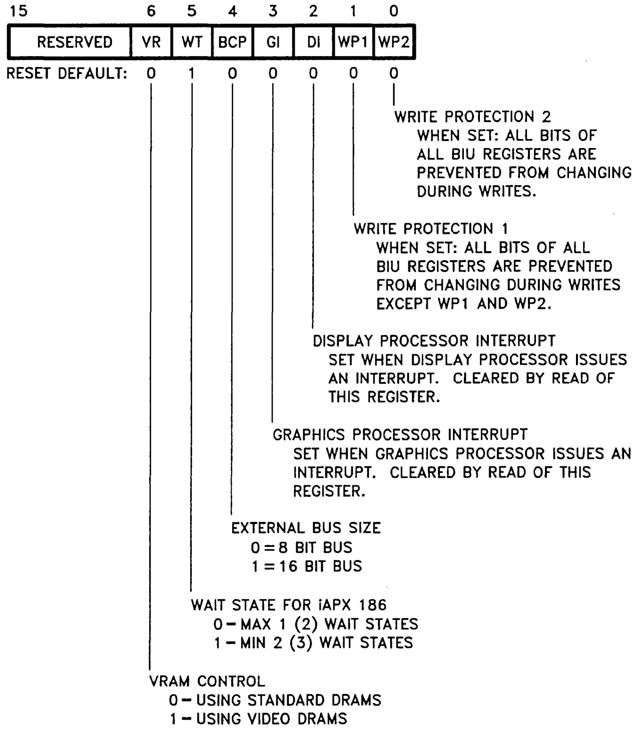
bit-4 – FMT - FIFO Empty - set if the Display FIFO empties.

Both bits are reset after reading the Status Register.

The setting of the External Priority Register can greatly affect the performance of the external CPU when it performs an access to the 82786. Unless the External Priority is greater than the Graphics Processor, whenever the Graphics Processor is busy with a command stream that demands significant bus bandwidth, the CPU may have to wait a significant amount of time before it can complete an access to the 82786. The CPU waits for the 82786 in the middle of a bus access until the 82786 returns the READY signal. During this wait time, the CPU will not be able to process anything, including interrupts. Of course, if the application is very graphics intensive and the CPU throughput is of lesser concern, then the Graphics Processor can be programmed with a higher priority.

Use the following priority values during your initial design. Once the system is working properly, you may wish to tweak the values for optimum performance. The optimum values are dependent on the CPU and video speeds as well as the CPU and graphics instruction mix and the window arrangement. In most cases, these registers will be initialized once and never changed. It may be advantageous in some specialized applications to adjust these values when the application changes modes.

	FPL	SPL
Display Processor	6	6
Graphics Processor	2	2
External Processor	4	
Trip Point		20


15
6
5
4
3
2
1
0
RESERVED
VR
WT
BCP
GI
DI
WP1
WP2
RESET DEFAULT: 0 1 0 0 0 0 0
WRITE PROTECTION 2
WHEN SET: ALL BITS OF ALL BIU REGISTERS ARE PREVENTED FROM CHANGING DURING WRITES.
WRITE PROTECTION 1
WHEN SET: ALL BITS OF ALL BIU REGISTERS ARE PREVENTED FROM CHANGING DURING WRITES EXCEPT WP1 AND WP2.
DISPLAY PROCESSOR INTERRUPT
SET WHEN DISPLAY PROCESSOR ISSUES AN INTERRUPT. CLEARED BY READ OF THIS REGISTER.
GRAPHICS PROCESSOR INTERRUPT
SET WHEN GRAPHICS PROCESSOR ISSUES AN INTERRUPT. CLEARED BY READ OF THIS REGISTER.
EXTERNAL BUS SIZE
0 = 8 BIT BUS
1 = 16 BIT BUS
WAIT STATE FOR iAPX 186
0 – MAX 1 (2) WAIT STATES
1 – MIN 2 (3) WAIT STATES
VRAM CONTROL
0 – USING STANDARD DRAMS
1 – USING VIDEO DRAMS

292007-19

The BIU Control Register contains a miscellaney of bits.

After the BIU Registers have been initialized, the WP1 and WP2 bits can be used to protect all of the BIU Registers (82786 Internal Register offsets 00h – 0Fh) from being rewritten. This will prevent faulty software from going wild and placing the 82786 into an unwanted state. Once WP1 is set, the only way to change the BIU registers is to reset WP1 first. Once WP2 is set, there is no way for the software to modify the BIU registers until a 82786 hardware RESET is performed.

After the 82786 causes an interrupt, the GI and DI interrupt bits are used to allow the software to determine whether the Graphics or Display Processor caused the interrupt. It is possible that both of these bits may be set if both processors have caused an interrupt by the time the interrupt handler reads this register. In this case, both interrupts should be handled by the interrupt handler.

Although it is not absolutely necessary to allow the 82786 to interrupt the CPU, it is very desirable. Graphics Processor interrupts can inform the software when it has completed all the commands as well as to report error conditions. Display Processor interrupts can inform the software when a new display field has begun. A new command can then be loaded into the Display Processor to be executed before the next display field. This facilitates operations such as smooth scrolling and blinking. The only hardware requirement to permit 82786 interrupts is that the 82786 INTR pin is tied to one of the interrupt controller inputs.

Although the 82786 always uses 16 bits, the 82786 can be used with both 8 and 16 bit processors. For an 8-bit CPU, separate transceivers are required for the low and high bytes to the 82786 (Figure 18). In both 8 and 16 bit modes, graphics memory may be accessed a byte at a time. Although the 82786 internal registers may be read a byte at a time, they all are considered to be 16 bits (even if some of the bits aren't used) and must always be written in 2-byte even-word pairs. In 16-bit mode, they must be written as a 16-bit word. In 8-bit mode, first the lower (even-address) byte is written and then the upper (odd-address) byte is written. With an 8-bit processor such as the 8088, both of the following assembly routines may be used to load the 16-bit BIUControl Register with AX.

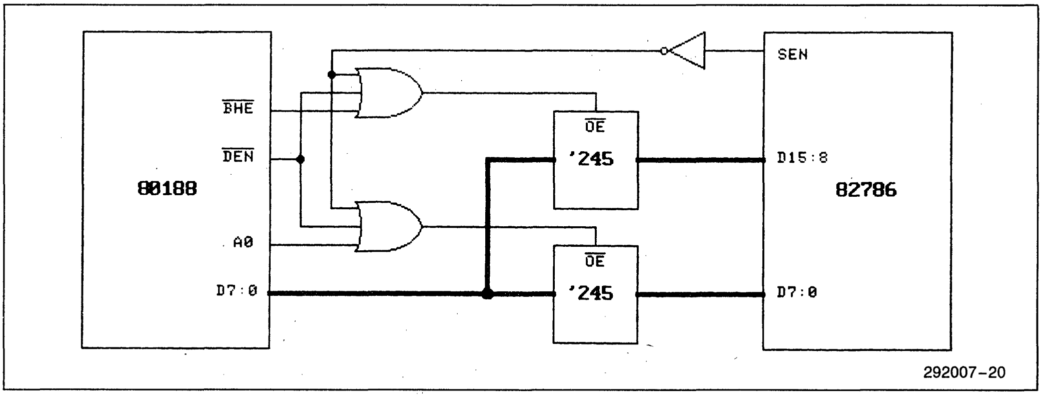


Figure 18. 8-Bit CPU Uses Two Data Transceivers to Connect to 82786

```
mov dx,BIUControl
out dx,al       ;write AL into low-byte of BIUControl
mov al,ah
inc dx
out dx,al       ;write AH into high-byte of BIUControl

or:

mov dx,BIUControl
out dx,ax       ;write AX into BIUControl word
```

In 8-bit mode, an even-byte write to an 82786 Internal Register does not change any of the 82786 Internal Registers, the data is simply saved until an odd-byte write to a 82786 internal register is performed. Then both the high and low bytes are written into that register. In effect, the even-byte address is ignored and an odd byte write will write into the register both the odd-byte data and whatever even byte data was last written, into the register address specified by the odd-byte access. There is no limit to the amount of time allowed between the even byte and corresponding odd-byte writes. An odd-byte write that is not preceded by an even byte will be ignored.

The 82786 always comes up in 8-bit mode after RESET. This means that a 16-bit CPU should change the BCP bit to one. It must perform two byte-wide accesses to do this. The following initialization code can be used.

```
mov dx,BIUControl
mov al,30h
out dx,al            ;write 30h into low-byte of BIUControl

xor al,al
inc dx
out dx,al            ;write 00h into high-byte of BIUControl

mov dx,InternalRelocation
mov ax,03F8h
out dx,ax            ;write 03F8h into InternalRelocation word
```

The 82786 is first placed in 16-bit mode (using two 8-bit writes), then the 82786 Internal Registers are located at the desired address (which is done with a 16-bit write). Next, the DRAM/VRAM and priority registers should be initialized. Byte-wide writes into the 82786 Internal Registers can not be performed while BCP = 1.

All the 82786 master mode operations are 16 bits wide independent of the BCP bit. This means that system memory must be accessible 16-bits at a time if master mode is to be used. The WT bit is set to 1 on reset. The VR bit is reset to 0 at reset.

4.3 80286 Synchronous Interface

The 82786 has been optimized for the 80286, which minimizes the interface logic requirements. Figure 19 shows a 82786 connected synchronously to an 80286. Much of the logic, such as the 82288, chip-select, and ready, can be shared by the rest of the 80286 system.

This configuration allows both master and slave accesses. The data transceivers allow the 80286 to access the 82786 and graphics memory and the 82786 to access the 80286 system memory. They also provide the isolation required to allow the 80286 to access system memory while the 82786 accesses graphics memory simultaneously. The tristate buffer 74LS367 is used to pull the 80286 upper address lines, COD/INTA, $\overline{\text{LOCK}}$ and PEACK to their proper states during master mode. If any of these signals are not used by the rest of the system, they need not be driven by a tristate buffer.

If master mode is not required, MEN will stay low and three of the four gates driving the data transceivers can be eliminated. Also, the tristate buffer, which is only used in master mode, may be eliminated. HREQ should be left open and the 82786 HLDA pin should be tied to ground so that the 82786 will never enter master mode.

Both the 80286 and the 82786 internally divide-by-two the CLK input and use both phases. For the 82786 to run correctly with the 80286, these phases must be correlated correctly. This can easily be done by observing the setup and hold times for rising RESET for both chips (see 80286 data sheet specifications and 82786 data sheet specifications). The 82C284 chip will meet this requirement.

Depending on the CLK speed and the type of DRAM/VRAM used, the 82786 may have very stringent CLK duty cycle requirements (see Section 3.2). It may not be possible to use the internal oscillator of the 82C284 chip but it may be possible to use an external oscillator to drive the 82C284 external clock (EFI) pin.

Clock skew between the 80286 and the 82786 should be kept to a minimum so the chips should be placed as close together as possible.

When the 82786 bus is free, the circuit in Figure 19 permits CPU slave accesses using 2 wait-states for writes and 3 wait-states for reads. Using DRAMs/VRAMs with slightly faster access times, the circuit in Figure 20 permits both read and write slave accesses using 2 wait-states. The 82C284 SRDY input is used instead of ARDY. The 82786 SEN timing is such that a minimum of 2-wait states are always generated for writes but a minimum of 2 or 3 wait-states are used for reads depending on the use of SRDY or ARDY. Notice that with 2 wait-state reads, the SEN signal must be qualified with $\overline{\text{CS}}$ so that SEN does not extend into the cycle following the slave write. The most critical relationship to be satisfied in order for 2 wait-state writes is:

$$\text{Tcac} < \text{Tc} + \text{Tch} - 15 - 45$$

For a 10 MHz 82786 the DRAM/VRAM column access time must be:

$$\text{Tcac} < 50 + 25 - 45 = 30 \text{ ns}$$

Note that x 1 DRAMs have two transceiver delays.

The critical timing calculations for slave mode are calculated as follows. The actual numbers calculated are for an 80286/82786 system running at 8 MHz.

chip-select-logic	=	path from 80286 address to 82786 $\overline{\text{CS}}$ pin				
	<	2 × clock period	—	address valid	—	setup
	<	2 × 286.T1	—	286.T13	—	82786.Ts1
	<	2 × 50 ns	—	35 ns	—	5 ns
	<	60 ns				
ready-logic	=	path from 82786 SEN to 82C284 SRDY pin				
	<	clock period	—	SEN active	—	ARDY setup
(if ARDY is used	<	286.T1	—	82786.S18	—	82C284.T13
as in Figure 19)	<	50 ns	—	25 ns	—	0 ns
	<	25 ns				
ready-logic	=	path from 82786 SEN to 82C284 ARDY pin				
	<	clock period	—	SEN active	—	SRDY setup
(if SRDY is used	<	286.T1	—	82786.S18	—	82C284.T11
as in Figure 20)	<	50 ns	—	25 ns	—	15 ns
	<	10 ns				

from SEN active to read data valid

read data valid ≥ 82786.Ts22 + transceiver delay

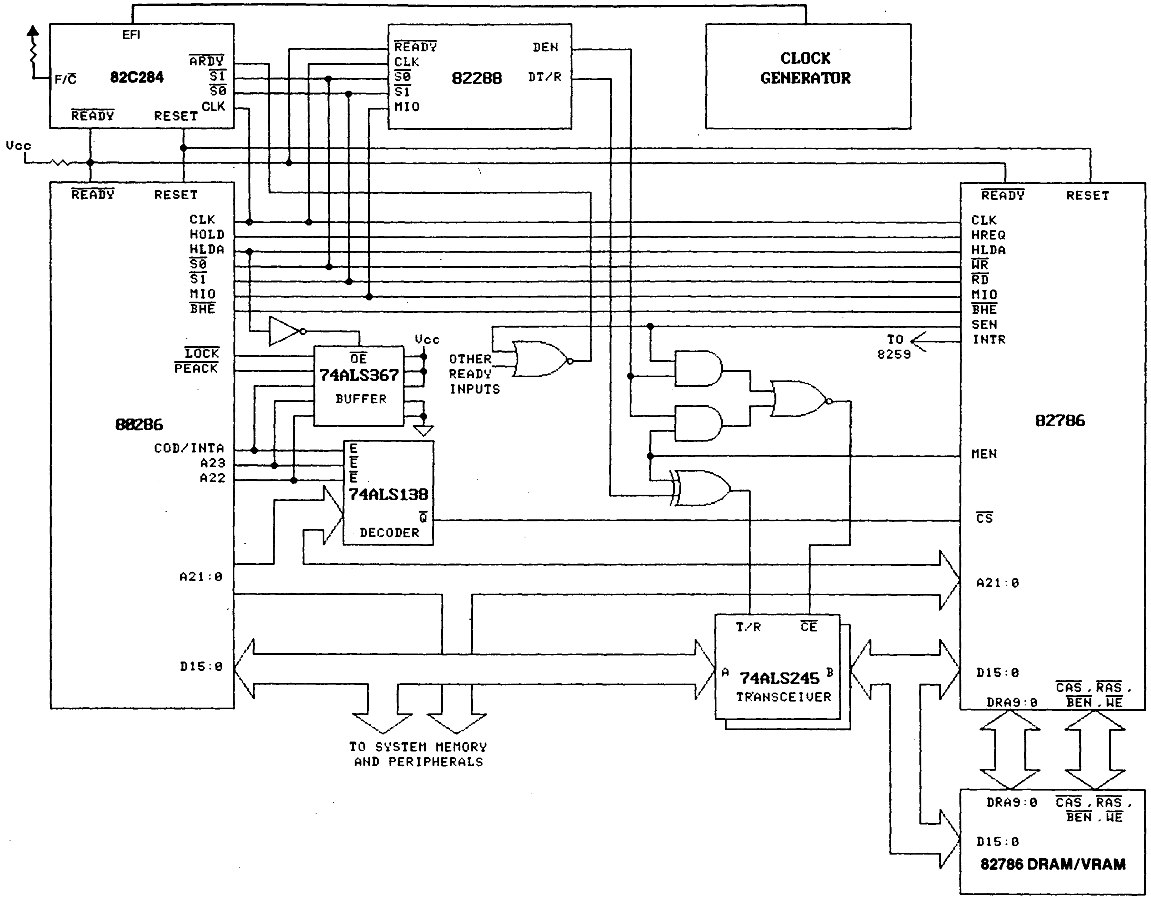
from SEN active to write data valid

writc data valid ≥ 82786.Ts20

The master mode signals generated by the 82786 are all within the specification range guaranteed by the 80286. In other words, if the system memory is designed to function with the 80286, it will also be able to function with the 82786. The only signals that may not be within the range of the 80286 specifications are the data bus signals due to the transceiver delays. Care must be taken to ensure that the memory subsystems that the 82786 is to be able to access in master mode can meet these more stringent requirements:

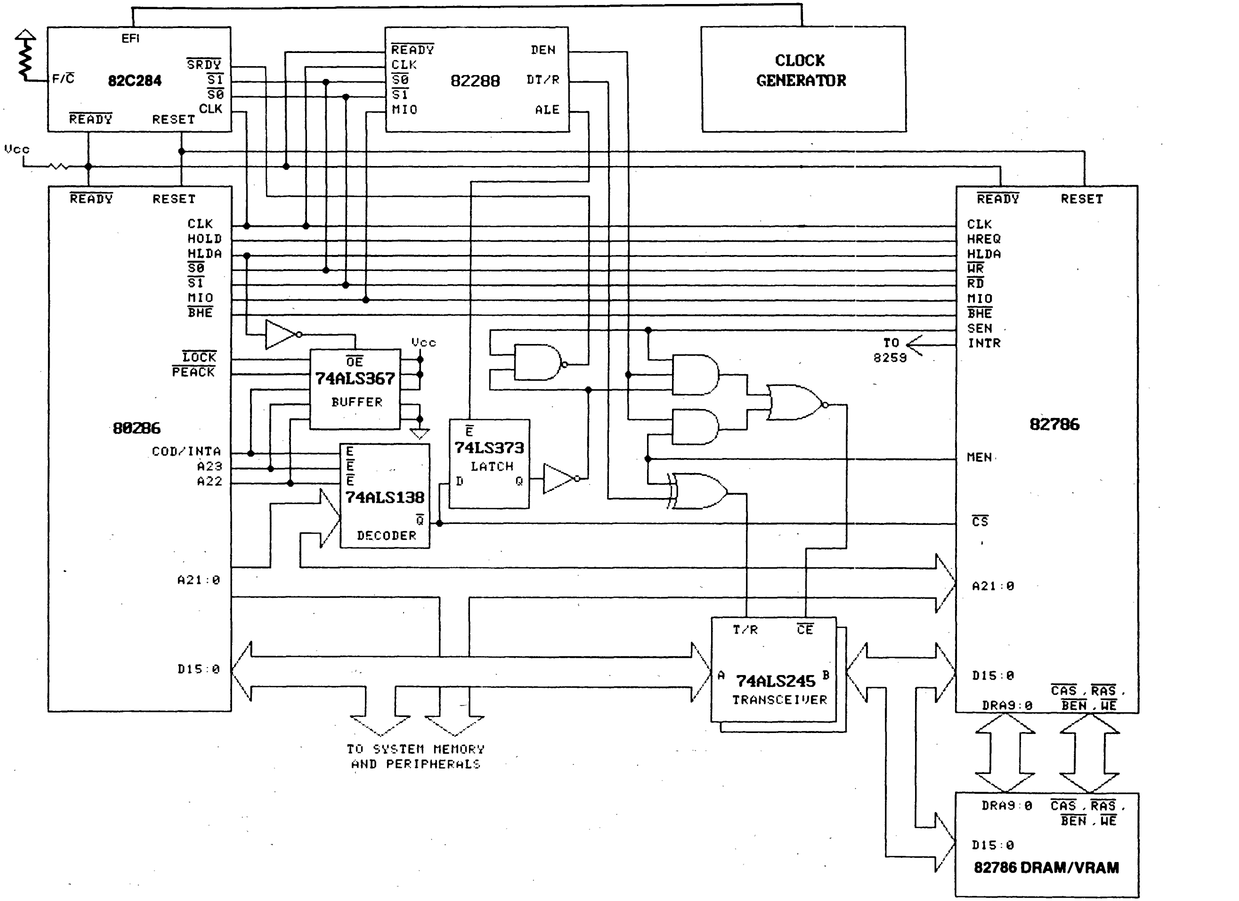
		data valid to falling clock after Tc phase 2		
read data setup	>	82786 read data setup	+	transceiver-delay
	>	82786.T8	+	data in to data out
	>	5 ns	+	Tprop
		data valid delay from falling clock after Ts phase 1		
write data valid	<	82786 write data valid	—	transceiver delay
	<	82786.T14	—	data in to data out
	<	40 ns	—	Tprop

The clock skew between the 80286 and the 82786 must be considered in all these calculations.



292007-21

Figure 19. 286/82786 Synchronous Master/Slave Interface Permits Minimum of 2 Wait-State Write, 3 Wait-State Read



292007-22

Figure 20. 286/82786 Synchronous Master/Slave Interface Fast DRAMs Permit Minimum of 2 Wait-State Read/Write

4.4 80186 Synchronous Interface

The 82786 supports a synchronous status interface to the 80186. The 82786 bus clock and the 80186 x 1 Crystal Input must be driven with the same external clock (EFI). The Reset inputs to the 82786 must be generated from the $\overline{\text{RES}}$ for the 80186 by delaying it by one clock (input). This guarantees that the 82786 Clock phase 1 is coincident with 80186 CLKOUT low. A synchronous 80186 interface is selected if BHE is high and MIO is high prior to falling 82786 RESET.

Generally this configuration will be used with a minimum of 3 wait states for the 82786 slave read and write accesses. Therefore the WT bit in the 82786 BIU Control Register should be set. The 82786 slave accesses will then only be initiated when the 82786 $\overline{\text{CS}}$ is actually activated.

There is, however, a way to allow this interface to use a minimum of 2 wait states (set WT=0). Rather than wait for $\overline{\text{CS}}$ to go active, the 82786 can be allowed to request a slave access as soon as the 80186 status lines go active. If the 82786 is not in the midst of another bus cycle and the CPU request is the highest priority, the bus will immediately be granted to the CPU and a bus cycle started. If the $\overline{\text{CS}}$ then goes active the 82786 can complete the access within 2 wait-states. If $\overline{\text{CS}}$ does not go active (because the 80186 is not accessing the 82786 but rather its own memory or I/O) then the 82786 will abort the bus cycle by running a dummy 82786 bus cycle.

If there is other RAM or ROM in the system besides the 82786 graphics DRAM/VRAM that the 80186 often accesses, then this 2 wait-state will probably hinder rather than help performance. Every time the 80186 fetches from its own system memory (such as an opcode fetch or operand access), and the 82786 bus is idle, the 82786 will waste time running a dummy cycle. Fortunately, the busier the 82786 bus is, the less likely it will be free when the 80186 initiates a bus cycle, and therefore the less likely the 82786 will waste time running a dummy cycle.

4.5 Asynchronous Interface

An asynchronous interface can be used to interface the 82786 with nearly any CPU. The CPU clock and the 82786 clock are independent and may run at different speeds. If the 80286 is connected asynchronously with the 82786 and both processors are run at approximately the same clock frequency, then the minimum possible wait-states is one more than for the corresponding synchronous mode.

Figure 21 shows a slave-only 10 MHz 82786 interface to an 8 MHz 80186. At 10 MHz, the 82786 requires that the address becomes valid S17 = 80 ns after $\overline{\text{RD}}$ or $\overline{\text{WR}}$ falls and remains valid for S16 = 130 ns. Because the 80186 address disappears the same cycle $\overline{\text{RD}}$ and $\overline{\text{WR}}$ fall, the address must be latched. This latched address can be shared by the other components on the 80186 bus.

Due to the indeterminate phase relationship between the CPU and 82786 clocks, care must be taken to ensure the read/write data timings have enough slack. When the read data is sampled, and when the write data is removed is determined by the CPU's ARDY input. The 82786 SEN signal is used to generate the ready signal which ensures that the data is indeed available. D-flip-flops can be used to delay the SEN signal to delay the CPU Ready signal. For a 10 MHz 82786:

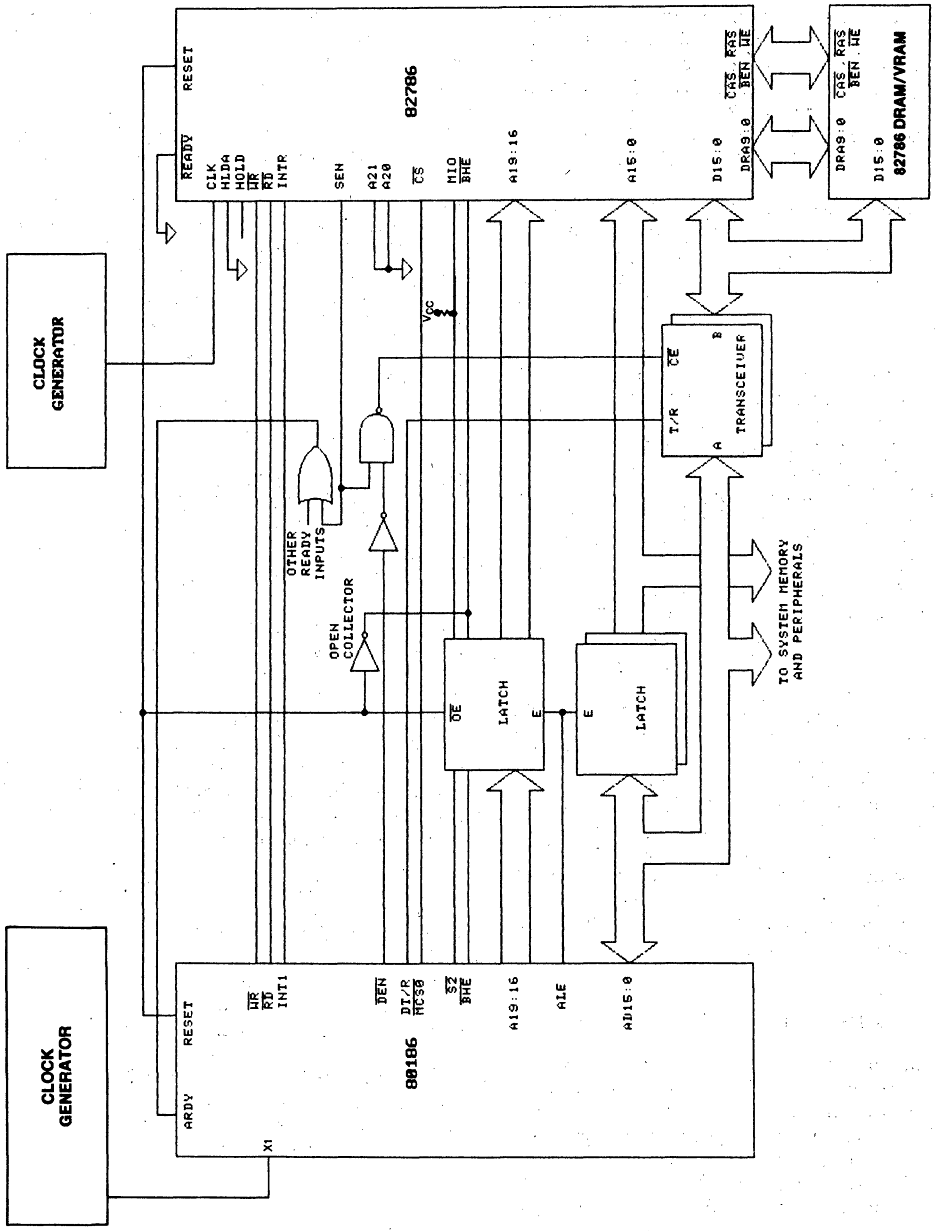
from SEN active to read data valid
read data valid ≥ 82786.Ts22 + Tprop

from SEN active to write data valid
write data valid ≥ 82786.Ts20

To initially place the 82786 into the asynchronous interface mode, the 82786 $\overline{\text{BHE}}$ pin must be low during the falling edge of RESET. To ensure this, the 74LS373 latch for $\overline{\text{BHE}}$ is tristated and an open-collector inverter pulls down $\overline{\text{BHE}}$ during RESET.

The 80386 processor can be interfaced to the 82786 either synchronously or asynchronously. For a synchronous interface, standard logic can be used around the 80386 to emulate a 80286 style bus for use with the interface described in Section 4.3. In this configuration the 82786 bus would run at half the clock rate of the 80386 (a 16 MHz 80386 would run with an 8 MHz 82786 bus). For an asynchronous interface, the standard local bus controller logic used by the 80386 to interface most peripherals can be used (Figure 22).

Although the actual bus transfers of a synchronous bus are faster than for an asynchronous bus, there are cases where an asynchronous interface provides the highest performance. For example, for a given display resolution, the Display Processor overhead of a 10 MHz 82786 is a lower percentage of the total bus throughput than for an 8 MHz 82786. If the 82786 is used with a 16 MHz 80386, then an asynchronous 10 MHz 82786 would have more bandwidth for the CPU and Graphics Processor than a synchronous 8 MHz 82786 and therefore CPU accesses, generally, will be completed faster with the asynchronous interface.



292007-25

Figure 21. 80186/82786 Asynchronous Slave-Only Interface

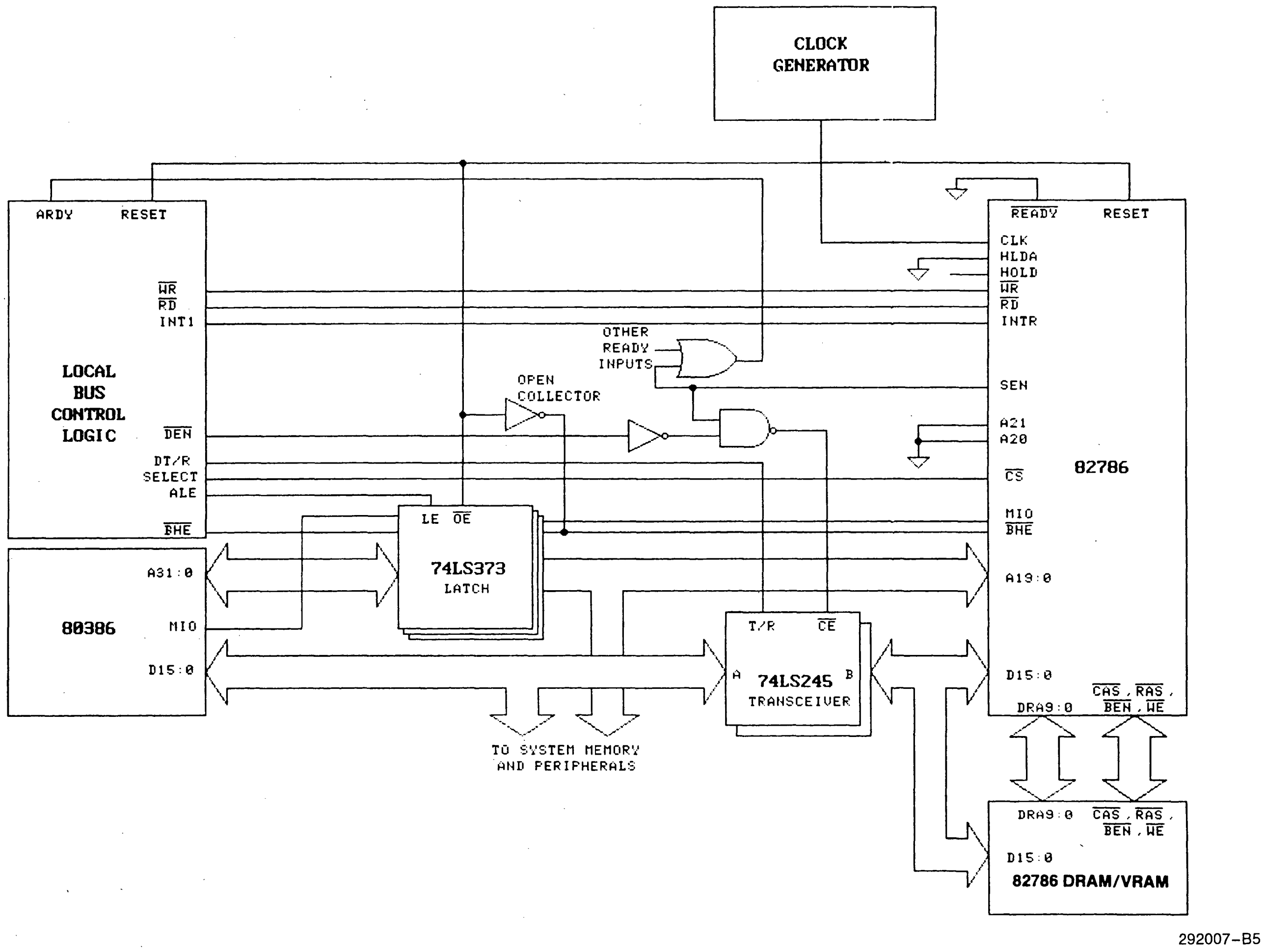


Figure 22. 80386/82786 Asynchronous Slave-Only Interface

4.6 Multiple 82786 Interface

For higher performance, it is possible to use several 82786 chips in the same system. Any of the above CPU/82786 interfaces can be used to attach multiple 82786s to one CPU in the system. Each 82786 will require its own separate DRAM/VRAM array.

The driving software for these multiple CPUs would most likely be sending nearly the same commands to all of the 82786s. Rather than forcing the software to write commands to each 82786 individually, it is possible to allow write commands to go to several or all the 82786s. One method of determining which 82786s should receive the write command would be to first write to an I/O port in which each bit corresponded to a different 82786. In Figure 23, the port bits set to 0 enable the corresponding 82786 for CPU writes. When a write to 82786 address-space occurs, all of the selected 82786s are chip-selected. The CPU will then wait for READY from all the selected 82786s before completing the bus cycle. In this manner, one, all, or any combination of 82786s can be written into at once.

Because it is impossible to read from several 82786s at once, a priority scheme is used on the I/O port to allow a read from only one of the selected 82786s. The circuit in Figure 23 only allows slave-accesses, the 82786s may not enter master-mode.

If master-mode operation of the multiple 82786s is desired, each 82786 must access the bus separately. A priority scheme is used to determine which 82786 is awarded the bus when the CPU issues HLDA. With only two possible 82786 masters, the random circuitry to hold one 82786 off the bus while the other is using it is straight-forward (Figure 24). With more 82786 masters, it is more feasible to use a state-machine (possibly implemented in PALs) to perform the arbiting.

5.0 VIDEO INTERFACE

The video interface connects the 82786 to the video display. The 82786 is optimized to drive CRT monitors but may also be used to drive other types of displays. Because CRTs provide an inexpensive method of generating moderate and high resolution, monochrome and color displays, this application note will concentrate on CRT interfaces. Section 5.10 briefly describes other display interfaces.

The video interface for a CRT is very dependent on the CRT requirements and the resolution and depth (bits/pixel) of the image desired. The 82786 can be programmed to directly generate all the CRT signals for up to 8 bits/pixel (256 color) displays at video rates up to 25 MHz. In addition, external hardware can be added to allow a color look-up table or to trade-off the number of bits/pixel for higher display resolutions, or to use VRAMs.

Some of the possible display configurations are shown below. The calculations assume a 60 Hz refresh rate. High resolution CRTs are often run at a slower rate, which permits the 82786 to generate significantly higher resolutions than those in the following table. All cases assume a CRT horizontal retrace time of 7 μs, except the 512 × 512 × 8 (10 μs) and 640 × 400 × 8 (13 μs) cases.

With Standard DRAMs

	Non-Interlaced	Interlaced
8 Bits/Pixel (256 colors)	512 × 512 640×400 640×480	900×675
4 Bits/Pixel (16 colors)	870×650	1290×968
2 Bits/Pixel (4 colors)	1144×860	1740×1302
1 Bit/Pixel (monochrome)	11472×1104	2288×1716

Multiple 82786s can be used together to generate even higher resolutions with more colors. For example, two 82786s allow a non-interlaced 1144 × 860 sixteen color display.

With Video DRAMs*

	Non-Interlaced
8 Bits/Pixel (256 colors)	1024 × 1024
4 Bits/Pixel (16 colors)	2048 × 1024
2 Bits/Pixel (4 colors)	2048 × 2048
1 Bit/Pixel (monochrome)	4096 × 2048

*For 64K by 4 - with 256K by 4 higher resolutions are supported

5.1 Various CRT Interfaces

CRT monitors use a wide variety of interfaces. Some use TTL-levels on all inputs, others require analog inputs. Some use separate color inputs (red, green and blue) and separate horizontal and vertical sync while others require that some or all of these signals be combined into composite signals. This application note will concentrate on the generation of separate color and horizontal and vertical sync signals. Standard techniques can be used to convert these separate signals into composite signals to meet the requirements of other displays.

The video clock (VCLK) required by the 82786 may be generated by a simple oscillator with TTL-outputs. Alternatively, the VCLK can be tied to the bus clock (CLK) (or any other available clock) if they are to run at the same speed.

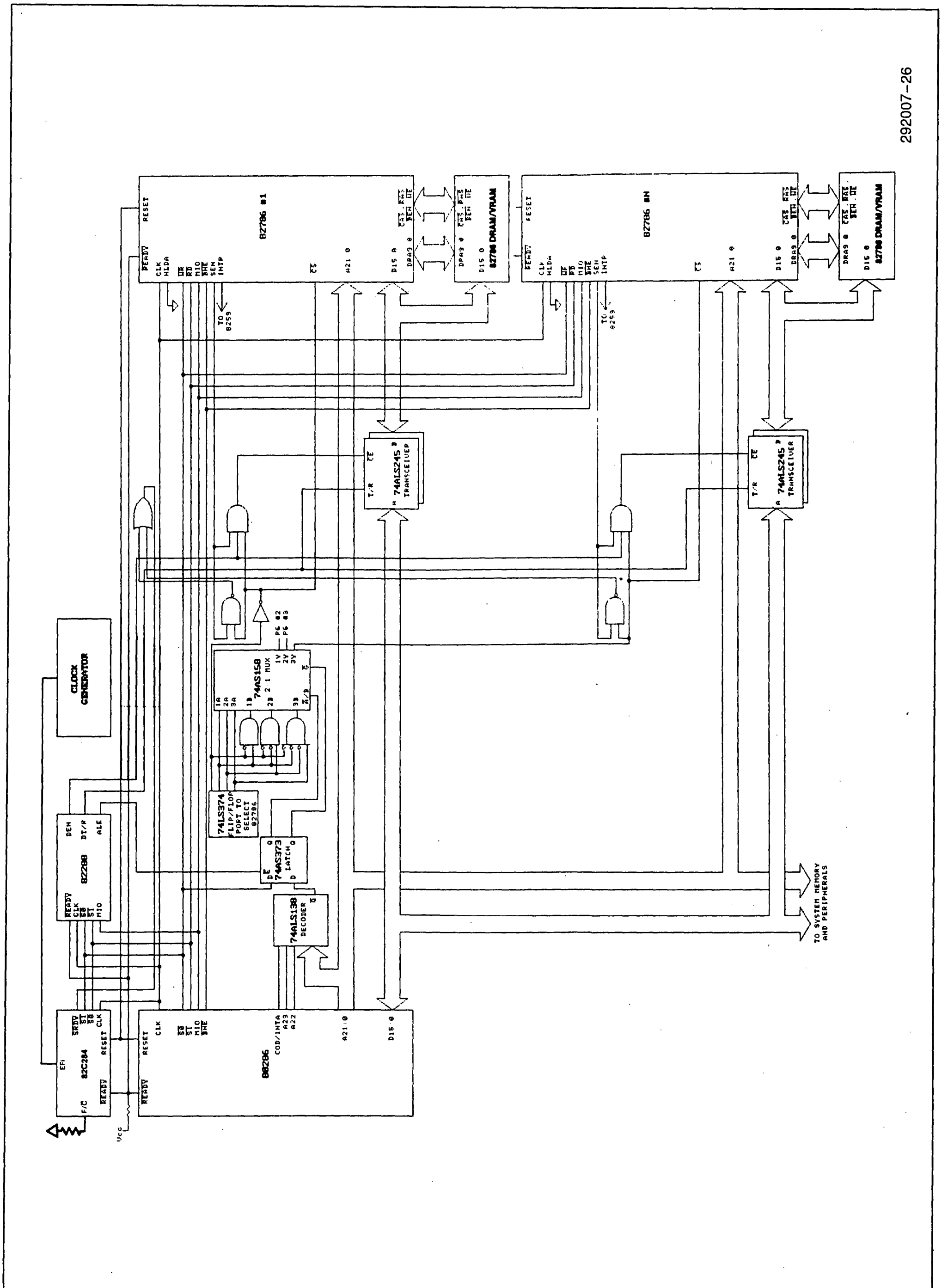


Figure 23. This Configuration Allows Several 82786s to be Written by 80286 Simultaneously—Only Slave Accesses are Supported

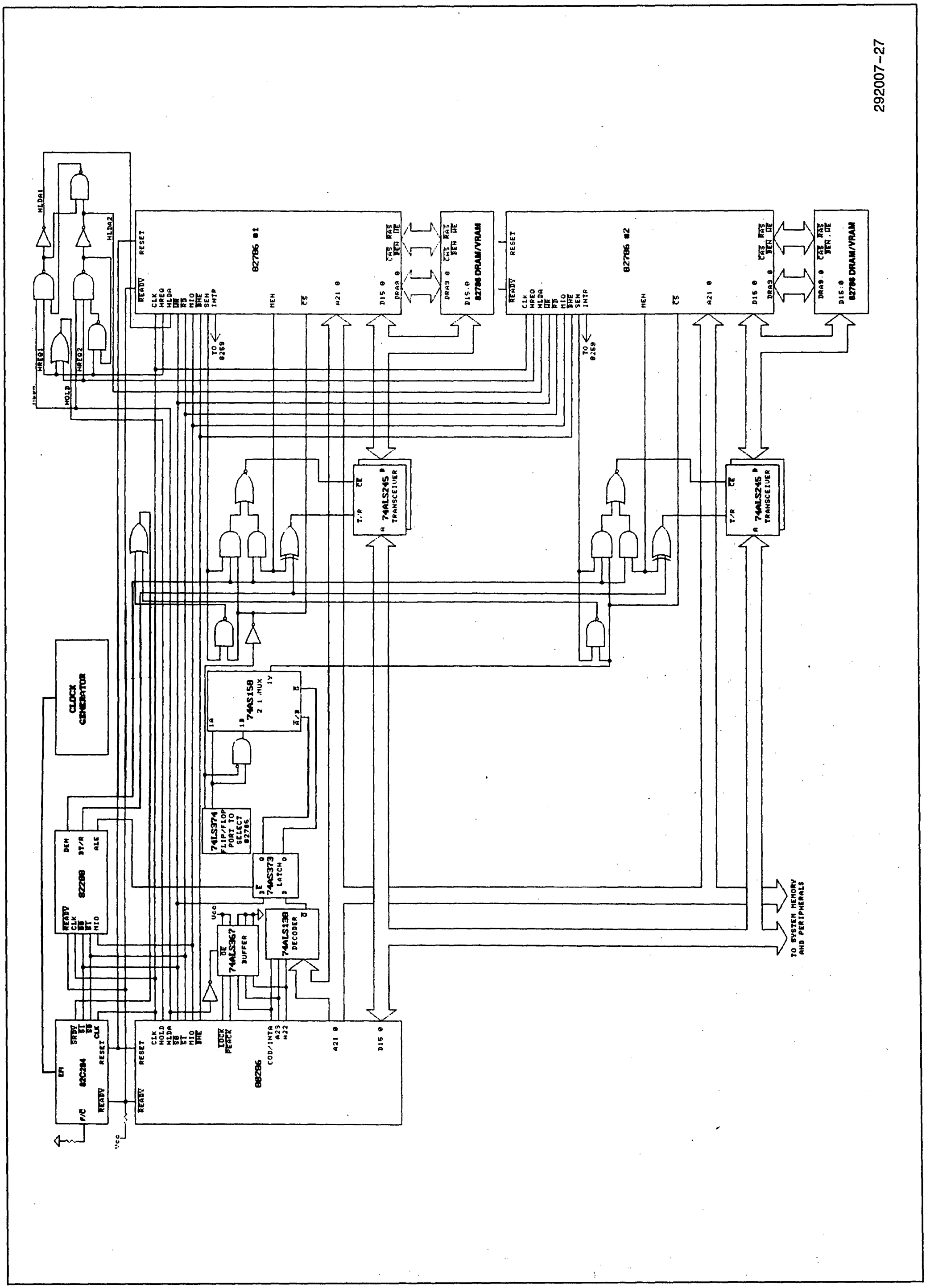


Figure 24. Two 82786s Connected to 80286, Permits Slave and Master Accesses

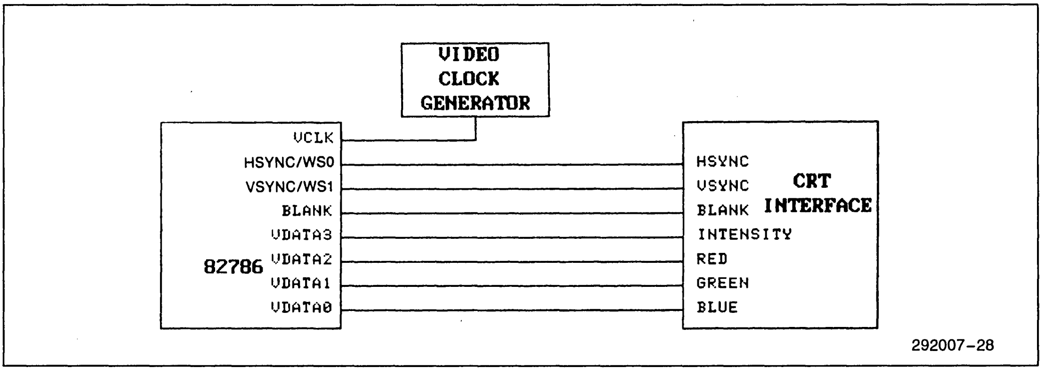


Figure 25. 82786 Can Directly Drive TTL-Input CRT Interface

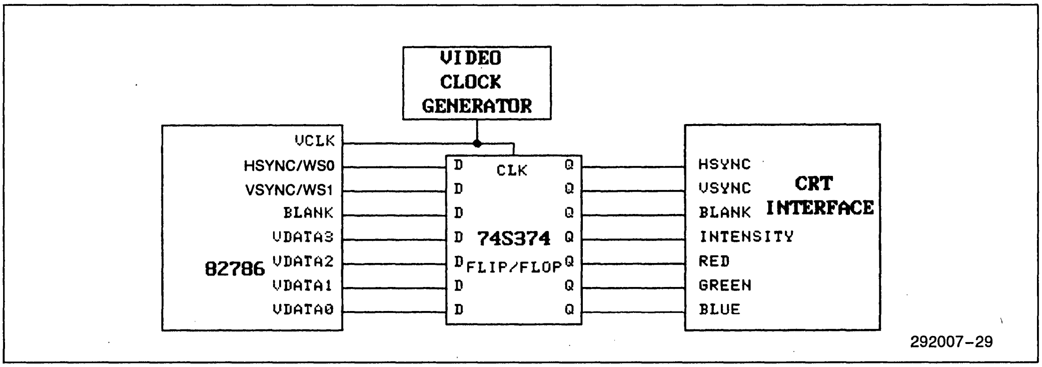


Figure 26. Buffer Used to Drive TTL-Input CRT Interface

5.2 CRTs with TTL-level Inputs

The simplest interface is to CRTs that use TTL-level inputs. The 82786 can generate these signals directly (Figure 25). However, the drive requirements of the CRT and cabling may make it necessary to buffer the signals (Figure 26). The example monitor in both of these cases happens to use a CRT that uses four-bits of color information per pixel. This means that 16 different colors are available and the CRT can use the 82786 1, 2, and 4 bits/pixel modes but can not take advantage of the 8 bit/pixel (256 color) mode. A monochrome monitor with only one TTL-level input could be connected directly to VDATA0 and use the 82786 1 bit/pixel mode but it then can not take advantage of any of the higher bit/pixel modes.

5.3 CRTs with Analog Inputs

Taking advantage of the 8 bit/pixel mode of the 82786 usually requires using a CRT with analog inputs. Signals for color CRTs with three separate analog video inputs, (red, green, and blue) can be generated using three digital-to-analog converters (Figure 27). Often these digital-to-analog converters can be constructed using simple resistor ladders (Figure 28). With 8 bits/pixel, usually three bits are used to select red, three for green and two for blue. This is because our eyes are much more sensitive to variations of red and green than of blue. These configurations can take advantage of all the 82786 modes; 1, 2, 4, and 8 bits/pixel.

The VDATA pins may be assigned to the three colors in any manner desired. In Figure 29 they are assigned so that a variety of colors are available for each mode (1, 2, 4, and 8 bits/pixel).

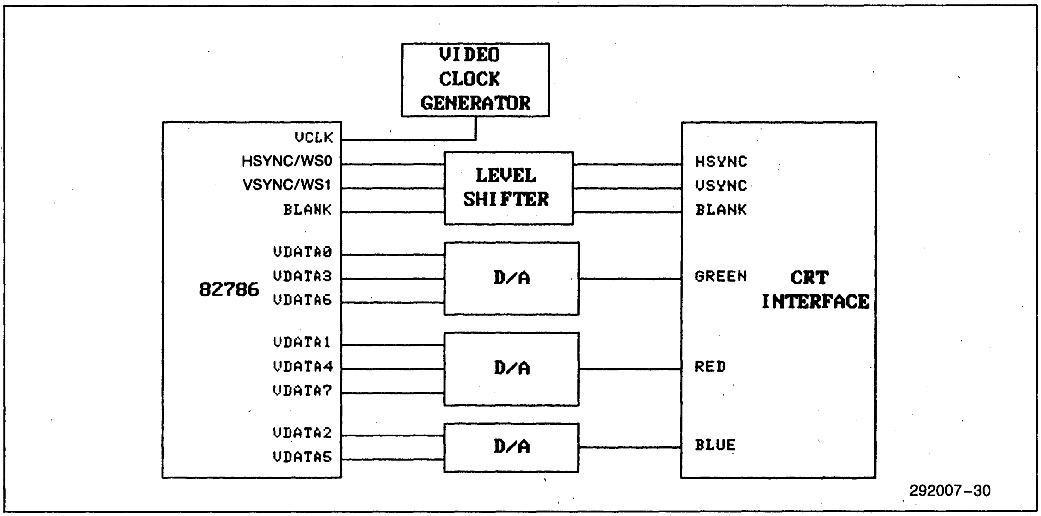


Figure 27. Analog CRT Interface Allows 256 Colors

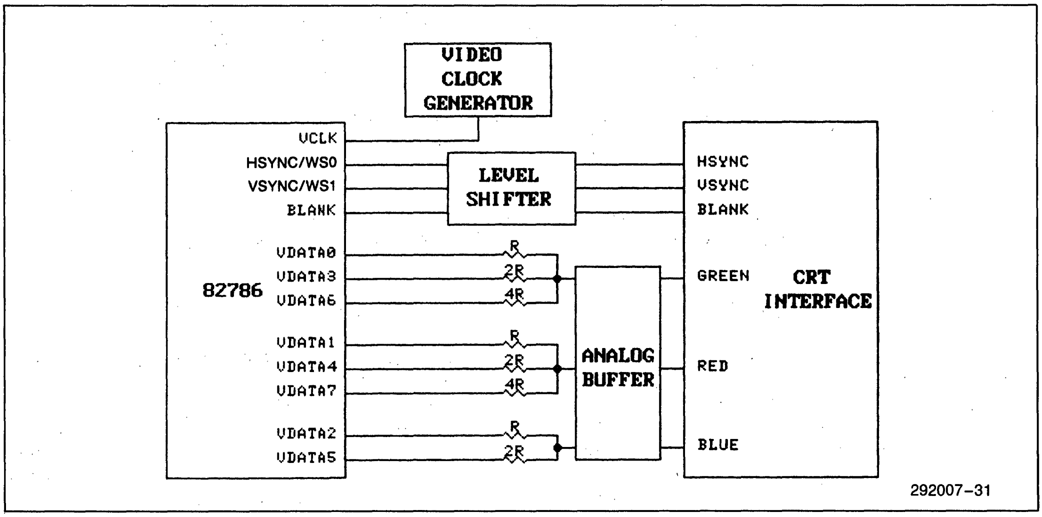


Figure 28. Resistor-Ladder Used for D/A

VDATA7	VDATA6	VDATA5	VDATA4	VDATA3	VDATA2	VDATA1	VDATA0
Red bit 0	Green bit 0	Blue bit 0	Red bit 1	Green bit 1	Blue bit 1	Red bit 2	Green bit 2

Figure 29. VDATA Pin Assignments

- The most-significant Green bit is connected to VDATA0 so that in the one bit/pixel mode this bit is controlled while the other bits are set to a constant level by the padding register internal to the Display Processor. If, for example, the padding bits are all set to zero, then a green and black image is shown in one bit/pixel windows.
- With two bits/pixel the most significant Green and Red bits are controlled while the rest are padded to a constant value. If, for example, the padding bits are set to zero then the colors black, green, red, and yellow are available in two bits/pixel windows.
- Four bit/pixel windows contain two Green bits and the most significant Red and Blue bits making 16 colors available.
- Eight bit/pixel windows allow control of all eight bits to make all 256 colors available.

5.4 Using a Color Look-Up Table

Color Look-up Tables, also known as Video Palette RAMs, allow more colors to be available with a minimal of actual bits/pixel and thus a minimal amount of display memory is required for the bitmap. For example, in a system using 16 bits of color information, 65536 different colors are possible. In such a system it is rarely necessary to display all 65536 colors on the screen simultaneously. It may be feasible to support a maximum of 256 colors simultaneously, providing that these 256 selections can be any combination of the 65536 available colors. Color look-up tables permit such a cost-effective system.

A block diagram of such a system is shown in Figure 30 and Figures 30a and 30b show actual circuits. The color look-up table can be loaded with up to 256 16-bit colors. In this way an 8-bit/pixel bit-map can be used to control the 16-bit colors.

The host CPU is responsible for loading the 16-bit colors into the look-up table. To load a color into a specific location in the look-up table, the 82786 Display Processor can be programmed to output the 8-bit address on the 8 VDATA pins during the horizontal and vertical blank times or on RESET by setting the Default Video Register. Then the CPU can load the color value into the 16-bit latch.

The circuitry in Figure 30 will then automatically write the 16-bit value into the look-up table during the next horizontal sync time. The CPU should generate the 74AS373 latch enable input so that the latch can be mapped into memory or I/O space and loaded by a CPU write. The register between the 82786 and the Palette RAM is used to allow the use of a RAM with a slower access time. This register is not necessary if a faster RAM is used.

The CPU program should wait until the color is loaded into the look-up table before loading the next color. One way to ensure this is to route the LookupLoading signal through a port which the CPU may poll. Sample assembly language code for this configuration follows this section. Another way is for the CPU program to delay a sufficient amount of time to ensure that HSync has occurred before writing the next value.

Hybrid circuits can be used which combine the functions of the look-up table, analog-to-digital conversions, and voltage shift for composite sync signals into one package. Figure 30b shows such a configuration. This particular hybrid circuit internally contains a 16 × 12-bit look-up table, 4 bits for each red, green, and blue.

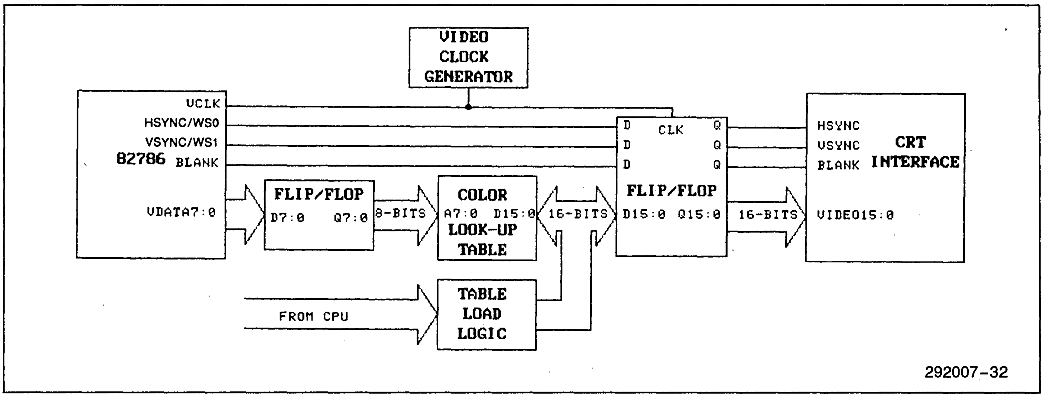
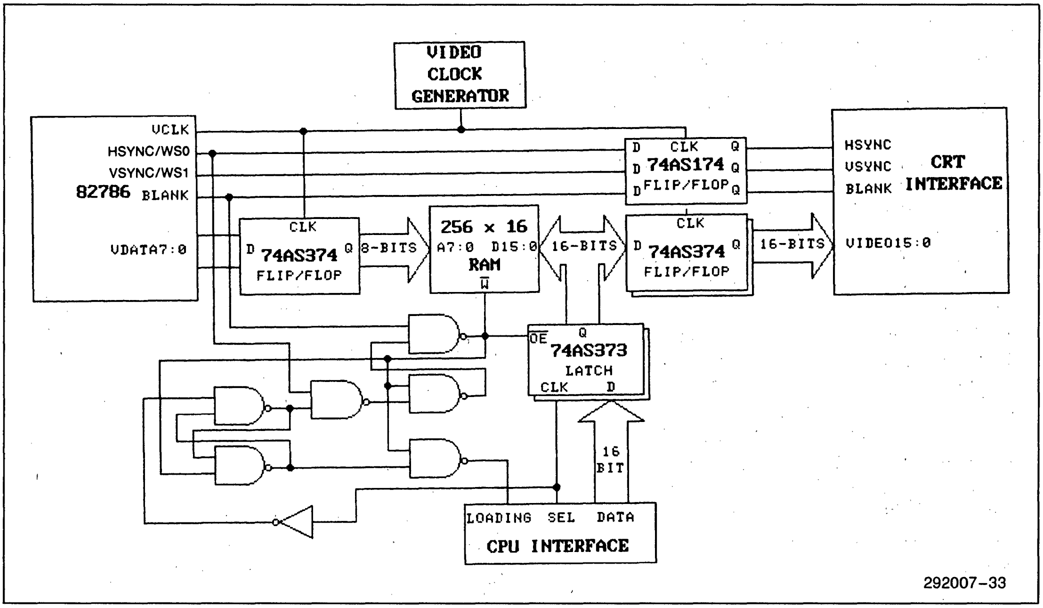
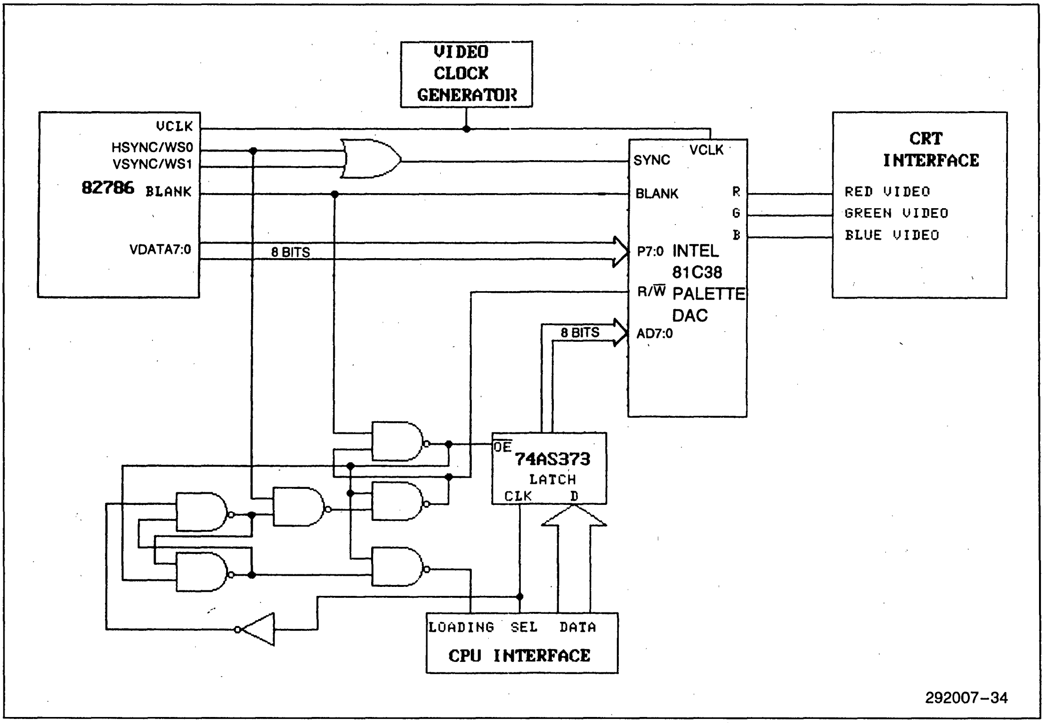


Figure 30. Block Diagram of Color Look-Up Table Used to Generate 16 Video Bits From 8



292007-33

Figure 30a. Circuit for Color Look-Up Table Used to Generate 16 Video Bits From 8



292007-34

Figure 30b. Hybrid Color Look-Up Table and DAC Simplifies Interface

```
Wait:   in      al,StatusPort         ;read port
        test    al,LookupLoadingBit   ;test LookupLoading bit
        jnz     Wait                  ;wait til last load completed
        mov     ax,EightBitAddr       ;get 8-bit value to load
        mov     DefaultVDATA,ax       ;make 82786 output during BLANK
        mov     ax,SixteenBitColor    ;get 16-bit color
        out     LookupLatch,ax        ;write color into latch
```

The look-up table is loaded by first writing the location into the 82786 DefaultVDATA register. Then a 4-bit color value is loaded into the latch along with color-select information. Therefore, in one load it is possible to place this 4-bit color value into any combination of the red, green, and/or blue tables.

5.5 Using the Window Status Signals

A graphics system design may require that the video data bits for different windows be interpreted in different ways. For example, the attributes controlled by various video data bits may need to be changed between windows for different tasks or number of bits/pixel. For these reasons, two Window Status bits are available externally which reflect a value which may be individually programmed for each window. These two pins always reflect the window which the display is currently scanning. The software is responsible for placing the two bit values for each window in the Tile Descriptor list.

In addition, the cursor can be programmed with a value for the window status bits which can be programmed to override the status bits from the windows for the portion of the display where the cursor resides.

The Window Status bits are multiplexed onto the HSync and VSync pins. Since they are only applicable during the visible display time, and since HSync and VSync are only applicable during the non-visible display time, Blank can be used to de-multiplex these pins (Figure 31).

A mode bit (bit 4 of CRTMode) in the Display Processor enables the Window Status bits so they become multiplexed onto the HSync and VSync signals. This bit must be set when the Window Status signals are used. In systems where the Window Status bits are not needed, this bit can be reset so that the HSync and VSync pins remain low during the visible display. This allows simpler systems to use HSync and VSync directly eliminating the need to AND these pins with Blank.

As an example, suppose the interpretation of the video data bits by a color look-up table was to be different for different windows. Possibly four different look-up tables are required for four different types of 8 bit/pixel windows. A large look-up table (1024 words) could be divided into four areas, one for each of the window interpretations. Then the Window Status bits could be used to select the area of the look-up table to be used for each specific window. Essentially four look-up tables would be available, one for each of four different types of windows. Figure 32 illustrates such a system.

The system also requires circuitry to load the look-up table such as that in the previous section. Note that the look-up table's Window Status inputs must be generated directly from the CPU when the RAM is to be loaded since they can not be programmed in specific states during the blank time as the VDATA pins can.

Another use of the Window Status bits is to allow 1, 2, 4, and 8 bit/pixel windows to each use a different look-up table along with a fifth look-up table for the cursor. A 1024 word look-up table above could be split up into four areas as above. Two of the areas can be used for two separate 8 bit/pixel look-up table and the other two shared by the 1, 2, and 4 bit/pixel windows for two separate look-up table for each of 1, 2, and 4 bits/pixel (Figure 33). The padding bits can be used to sub-divide this second area into separate tables for 1, 2 and 4 bit/pixel windows. Finally, this same table could also be used for twelve look-up tables, four each for 1, 2, and 4 bit/pixel windows.

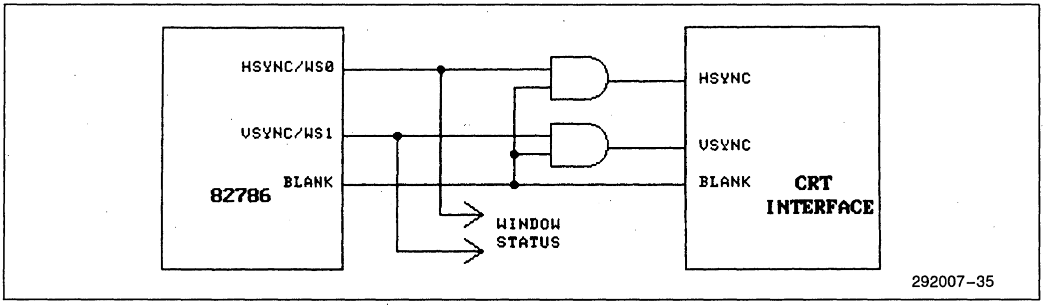
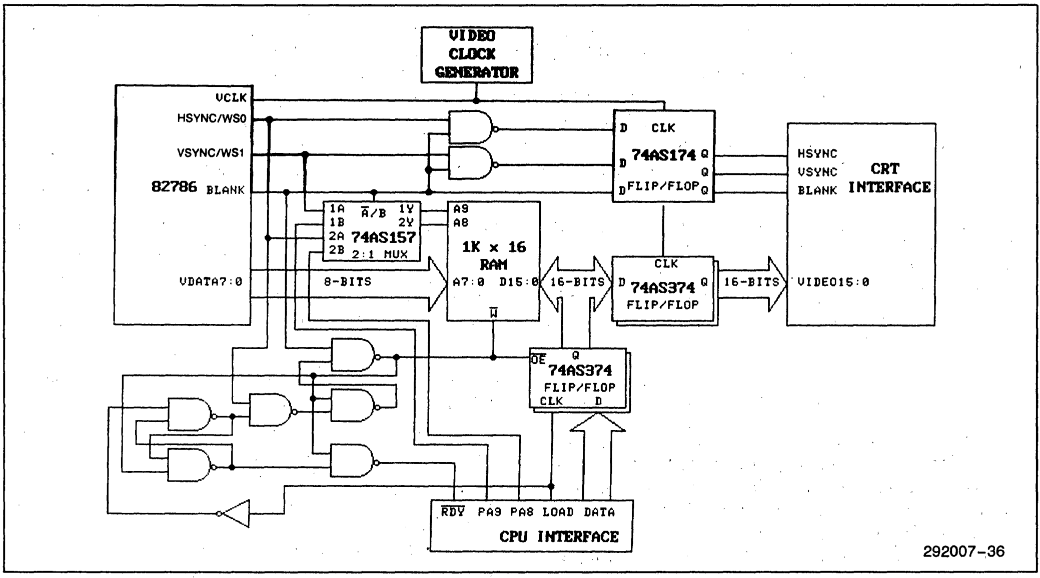
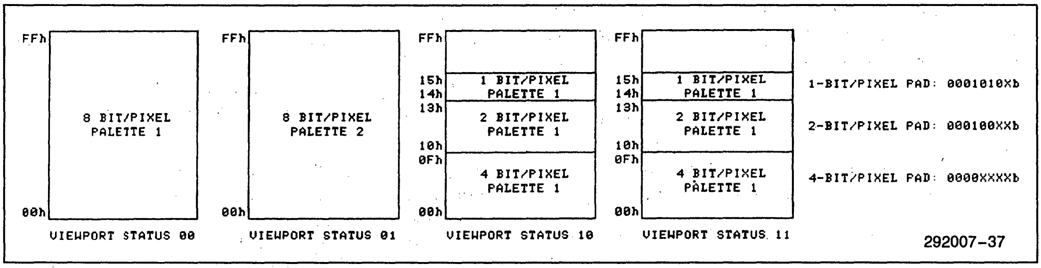


Figure 31. Using Blank to De-Multiplex Window Status



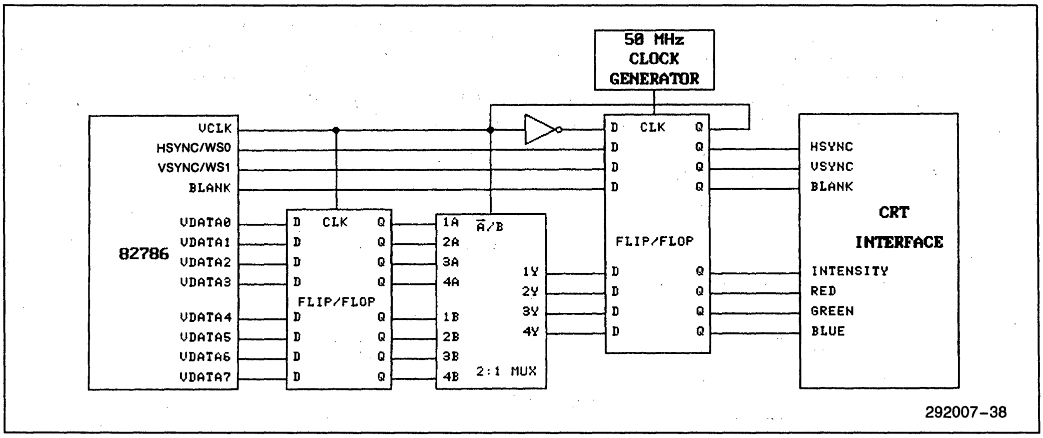
292007-36

Figure 32. Four Color Look-Up Tables—Selectable by Window Status Outputs



292007-37

Figure 33. Window Status and Padding Bits Allow Two Separate Look-Up Tables for Each of 1, 2, 4, and 8 Bit/Pixels



292007-38

Figure 34. External Multiplexer Allows Up to 50 MHz Video with 4 Bits/Pixel

5.6 Higher Resolutions with Standard DRAMs

The Video Clock rate on the 82786 can be a maximum of 25 MHz. For a non-interlaced display refreshed 60 times per second this limits the resolution to 512 × 512 or 640 × 400 or equivalent displays. 640 × 480 can also be achieved using a CRT with fast horizontal retrace. Still, some graphics system designs may require more detailed displays and therefore more resolution. It may very well be cost-effective to trade-off the number of bits/pixel for higher resolution. This is especially true in the case of monochrome displays where 256 grey-shades are not required but high resolution is.

The 82786 allows this trade-off to be made very effectively. Figure 34 shows how a video data rate of up to 50 MHz may be obtained with 4 bits/pixel (16 colors). The 82786 is used to output 8 bits of video data at a 25 MHz rate. The external multiplexer switches between the low 4 bits and the high 4 bits at a rate of 50 MHz. The register before the multiplexer is used to ensure that enough set-up time is provided for the multiplexer. The register after the multiplexer ensures that the video data out has smooth transitions. The circuit uses an inverter and one register stage to divide the 50 MHz clock by 2 to create the 25 MHz video clock for the 82786. Instead of a multiplexer such as the 74S157, a 74AS298 chip could be used which contains the multiplexer and the register in the same package.

The software has a minimum number of changes. The Graphics Processor is programmed identically and manipulates the bitmaps in the conventional manner (although it does not make sense to use 8 bits/pixel bitmaps since they cannot be displayed). The display processor programming is slightly different. The Accelerated Video control bits (CRTMode bits 1,0) are set for High Speed video (01). The HSynStp, HFldStrt, HFldStp, and LineLen registers are programmed for half the number of dot clocks (because the 82786 VCLK is half the speed of the pixel dot clock).

The Strip and Tile Descriptors list also change only slightly. Windows are programmed for same number of bits/pixel and FetchCount as they would be for non-accelerated modes. However, windows may only be positioned horizontally to start on even pixel boundaries. That is, they may only start at every-other pixel, not at any pixel as permitted with non-accelerated modes. This is because both an even and odd pixels are output on the VData pins simultaneously and it is not possible to mix windows during a single VCLK. The only valid values for the start/stop bits are listed in the following table. Notice that the Accelerated Modes do not permit all possible bitmap depths because fewer than 8 bits/pixel are available to the display.

Vertically, the windows may still be positioned at any pixel. The programming of the one pixel horizontal and vertical borders also does not change.

High-Speed video mode also requires that the Field windows are programmed with half the number of actual pixels for the pixel count (BPP/Start/Stopbit) register which again restricts horizontal positioning to a two pixel resolution.

The horizontal cursor position is programmed as half the actual value so the positioning is also restricted to a two pixel resolution. Vertically, the cursor is programmed as normal. Since the cursor is only a 1 bit/pixel region, every other horizontal pixel reflects only the cursor padding value so although simple cursor patterns are possible, arbitrary shapes are not possible with the box cursor. For this reason, the programmer may wish to create the cursor in software when using these high-resolution modes rather than use the 82786 hard-

Bitmap Depth	None (25 MHz)		High-Speed (50 MHz)		Very-High-Speed (100 MHz)		Super-High-Speed (200 MHz)	
	Start Bit	Stop Bit	Start Bit	Stop Bit	Start Bit	Stop Bit	Start Bit	Stop Bit
1 bit/pixel	0-15	0-15	odd	even	15,4,7,3	12,8,4,0	15,7	8,0
2 bit/pixel	odd	even	15,11,7,3	12,8,4,0	15,7	8,0	—	—
4 bit/pixel	15,11,7,3	12,8,4,0	15,7	8,0	—	—	—	—
8 bit/pixel	15,7	8,0	—	—	—	—	—	—

ware cursor. The crosshair cursor works well in Accelerated Mode, although the horizontal and vertical lines become two pixels wide and horizontal positioning is also limited to two pixels.

It is also possible to use external hardware to create the cursor. One method is to program the cursor as invisible (transparent and all background) and use the cursor's window status signals to activate the external hardware.

The horizontal zoom capability is also affected. Rather than replicating each individual pixel, pairs of pixels are replicated. Vertical zoom works as normal.

Figure 35 shows a configuration for video data rates of up to 100 MHz with 2 bits/pixel. A shift-register is used to multiplex the 8 video bits from the 82786 into 2-bit streams. A 74AS74 flip/flop is used to divide the 100 MHz clock by four. Every fourth clock the 82786 VCLK is raised and the shift registers are loaded with the previous 82786 VDATA. The video data is delayed two cycles by this circuit while the Sync and Blank are delayed only one. This should not be a problem if the 82786 is programmed to generate the correct Sync. The 82786 is limited to positioning the sync transitions at multiples of four pixels. If more accurate positioning is required, extra flip/flops can be used to delay sync for more cycles.

The timing in Figure 35 is very tight and the circuit may not operate at 100 MHz over all operating temperatures. The limiting speed path is the 74F195 shift-register parallel-load time (delay from clock to outputs valid) which must meet the set-up time of the 74AS374.

Figure 36 shows a configuration for video data rates of up to 200 MHz with 1 bit/pixel. Unfortunately, there is no TTL-logic available today which will run at the speeds required for 200 MHz. Therefore ECL or some other high-speed logic must be used to generate video at these high rates. Figure 36 converts the video data signals from the 82786 from TTL to ECL levels and then uses ECL shift-registers to generate the 200 MHz signal.

The software for the configurations in Figures 35 and 36 requires changes similar to the Figure 34 case. The window StartBits and StopBits are programmed restricted as shown in the preceding table. The pixel count for Field regions is also one-fourth or one-eighth the actual size. Horizontal positioning is also restricted to four and eight pixels for the 100 MHz and 200 MHz rates respectively. The Accelerated Video control bits must also be programmed for these configurations.

After the video signals are accelerated to these higher speeds, color look-up tables and analog-to-digital converters may be used. The circuits in the previous sections must be adapted for these higher speeds.

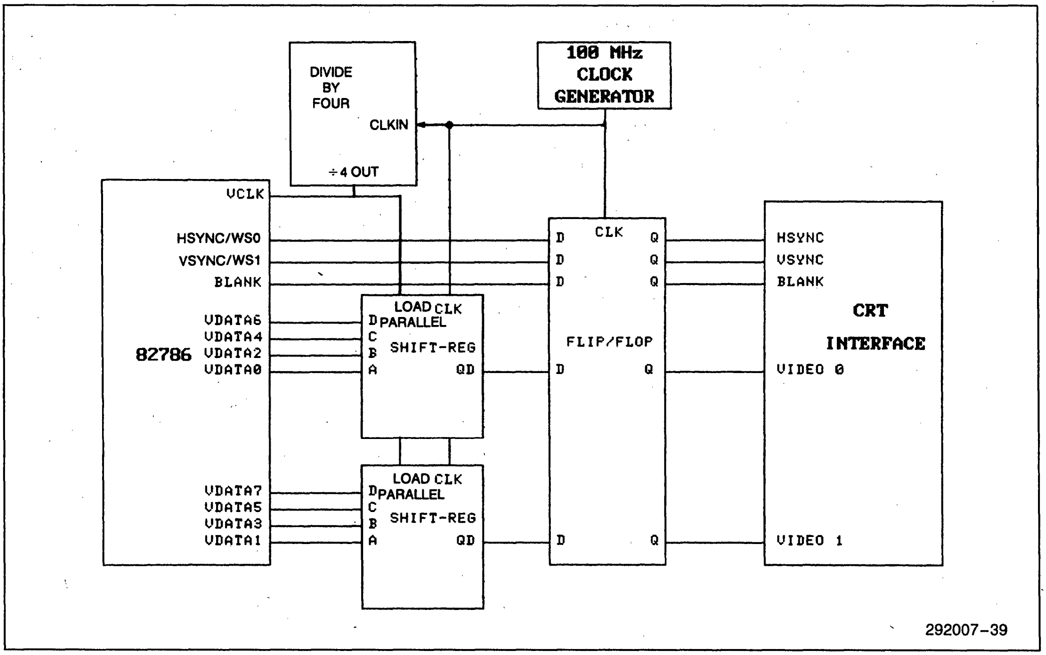


Figure 35. External Shift-Register Allows Up to 100 MHz Video with 2 Bits/Pixel

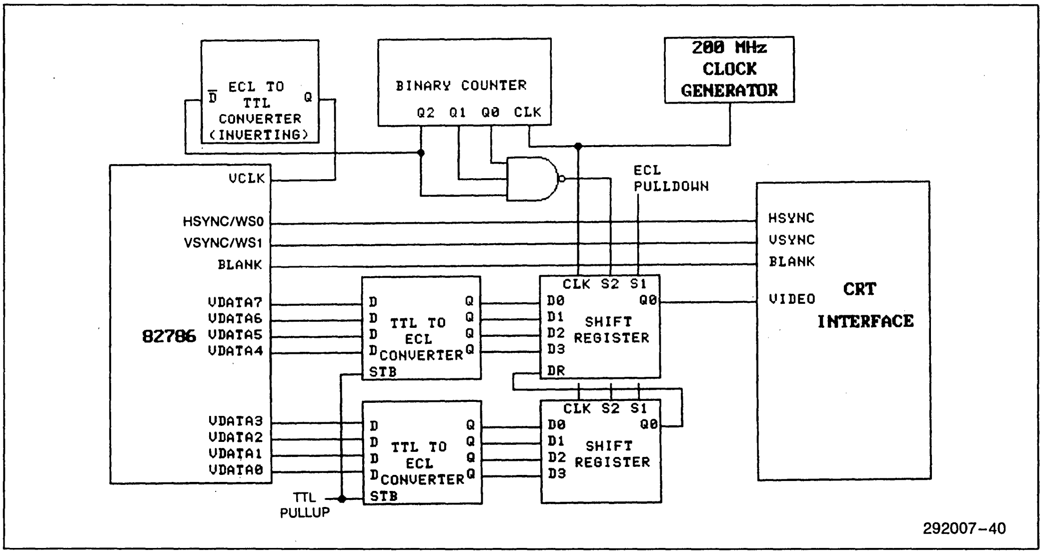


Figure 36. External ECL Shift-Register Allows Up to 200 MHz Video with 1 Bit/Pixel

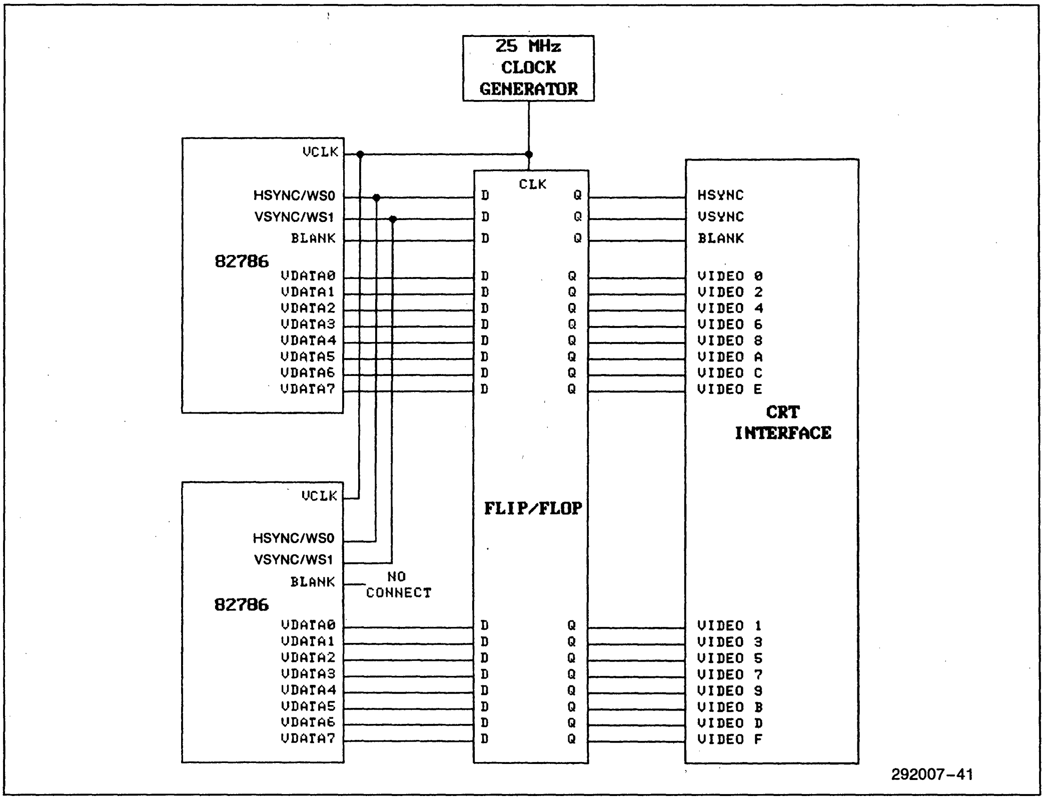


Figure 37. Two 82786s Can Generate 25 MHz Video with 16 Bits/Pixel

5.7 Multiple 82786s

If more colors or resolution are required than possible with one 82786 at a given resolution, several 82786s can be used together to generate the necessary bits/pixel. Figure 37 shows two 82786s used together to generate 16 bits/pixel at a 25 MHz video rate. This configuration would allow a 640 × 480 display with 65536 colors.

Both 82786s' video must be kept in sync. To allow this, one 82786 is programmed as normal to generate the master video horizontal and vertical sync. The second 82786 is programmed for slave video sync with the Slave CRT control bit in the CRTMode Register (Display Processor register 5-bit 3 set). The HSync and VSync lines of the slave 82786 then become inputs and are driven by the HSync and VSync output lines of the master 82786. If the window status signals are used, the master's HSync and VSync signals should be qualified with the Blank signal (similar to Figure 31) to correctly drive the slave 82786 HSync and VSync inputs. Window Status signals are only available from the master 82786 since the slave uses these pins as inputs.

Both 82786s should have six of their eight video timing registers (HSyncStp, HFldStrt, HFldStp, LineLen, VSyncStp, VFldStp, VFldStrt, FrameLen) programmed identically; HFldStrt and HFLdStp should be programmed to be 2 less than the master. (These parameters are calculated in Section 5.11.)

The slave 82786 will then automatically sync itself up to the master 82786 by waiting for its HSync input to fall before each scan line and waiting for its VSync input to fall before beginning a new display field. If a non-interlaced display is used, the two 82786s will always be in sync.

If an interlaced display is used, care must be taken to ensure both 82786s start on the same field. The easiest way to ensure they lock in sync correctly is to ensure they start scanning the display simultaneously. First set up the slave 82786 CRTMode and video timing registers with a LD__ALL command. The slave 82786 will then be ready to begin scanning the display but will wait until HSync and VSync fall. HSync and VSync will be floating because they are tristated by all the 82786s. Then the master 82786 can be set up with a LD__ALL command to program its CRTMode and video timing registers. Once the master starts scanning, the HSync and VSync signals will be driven by the master and all 82786s will begin on the even interlaced field.

To create a 16 bit/pixel bitmap, both 82786 Graphics Processors should be programmed for 8-bit/pixel bitmaps of identical size. To draw in both bitmaps, a graphics command block (GCMB) can be created for both 82786s. These GCMBs are generally identical for both 82786s except for the color values for the Def__Color and the mask value for the Def__Logical__Op commands. To display 16 bit/pixel bitmaps, both 82786s should be given an identical Strip Descriptor list for each to display 8 bits of each 16 bit pixel.

Similarly, 8-bit/pixel bitmaps could be created by splitting the bitmaps between both 82786s having each 82786 responsible for 4 of the 8 bits/pixel. This would split the work between the two 82786s so that the BitBlt and Scan__Line fill graphic commands will execute twice as fast. Also, because the Display Processor bus overhead is split between the 82786s, there will be less bus contention so all other drawing commands will be faster.

Alternatively, 8-bit/pixel bitmaps could be generated by only one of the 82786s. This would minimize the overhead between the host CPU and the 82786 since the CPU must communicate with only one 82786.

The method in which the 16 video data bits are mapped into colors for the display interface will determine which of the two above methods will be used for bitmaps of 8 bits/pixel or less. If the mapping is flexible enough, it may be feasible to create any bitmap depth. For example, 9 bits/pixel bitmaps could potentially be created using one 82786 for 8 bits and the other for only 1 bit of each pixel.

The displays discussed in the previous section obtained high resolutions at the expense of bits/pixel. Several 82786s can be combined to provide more bits/pixel at these high resolutions.

Figure 38 shows a configuration that uses two 82786s to create a 4-bit/pixel display at a video rate of 100 MHz. This configuration would support a 1144 × 860 sixteen-color non-interlaced 60 Hz display. Each 82786 is required to generate 2 bits of each 4-bit/pixel. Therefore, both 82786s draw and maintain half of the bitmap in their own graphics memory, 4-bit/pixel windows are divided into two 2-bit/pixel bitmaps, one generated by each 82786. The Graphics Processors are programmed as normal for 2-bit/pixel bitmaps. The Display Processors are programmed the way mentioned in the previous section. Each window is programmed with one-fourth the horizontal positioning resolution.

5.8 Video RAM Interface

The 82786 can use dual-port video DRAMs (VRAMs) to generate the video data stream. The VR bit in the BIU Control Register must be set to 1 to enable the mode. In this mode the first tile in each strip generates VRAM cycles; the second tile and any subsequent tiles in the strip generate DRAM cycles. In VRAM Mode, a minimum of two tiles must exist. The first tile is programmed for the VRAM. The second tile must be programmed to be a field tile detailed by the F bit in the Tile Descriptor if no hardware overlays are required. There is no limit on the number of strips. The pixel data for every scan line in the entire display must be contained in a single row in memory (256 words for non-interleaved memory and 512 words for interleaved memories). The Strip Descriptors for each VRAM tile are set up to indicate only 1 pixel. The address specified for this pixel corresponds to the first display pixel.

During the horizontal retrace period, the 82786 transfers the contents of the memory row containing the first pixel into the VRAM shift register. The VRAM shift clock is gated with a Blank signal. During the active display time, the shift clock is active and periodically clocks out the video data. External multiplexers must be used to convert the 16-bit (32 interleaved) data stream into a serial stream depending upon the bits per pixel needed (Figure 9).

In this mode, pixel depth is fixed by external hardware and all Display Processor registers referring to video data fetch should be programmed to zero.

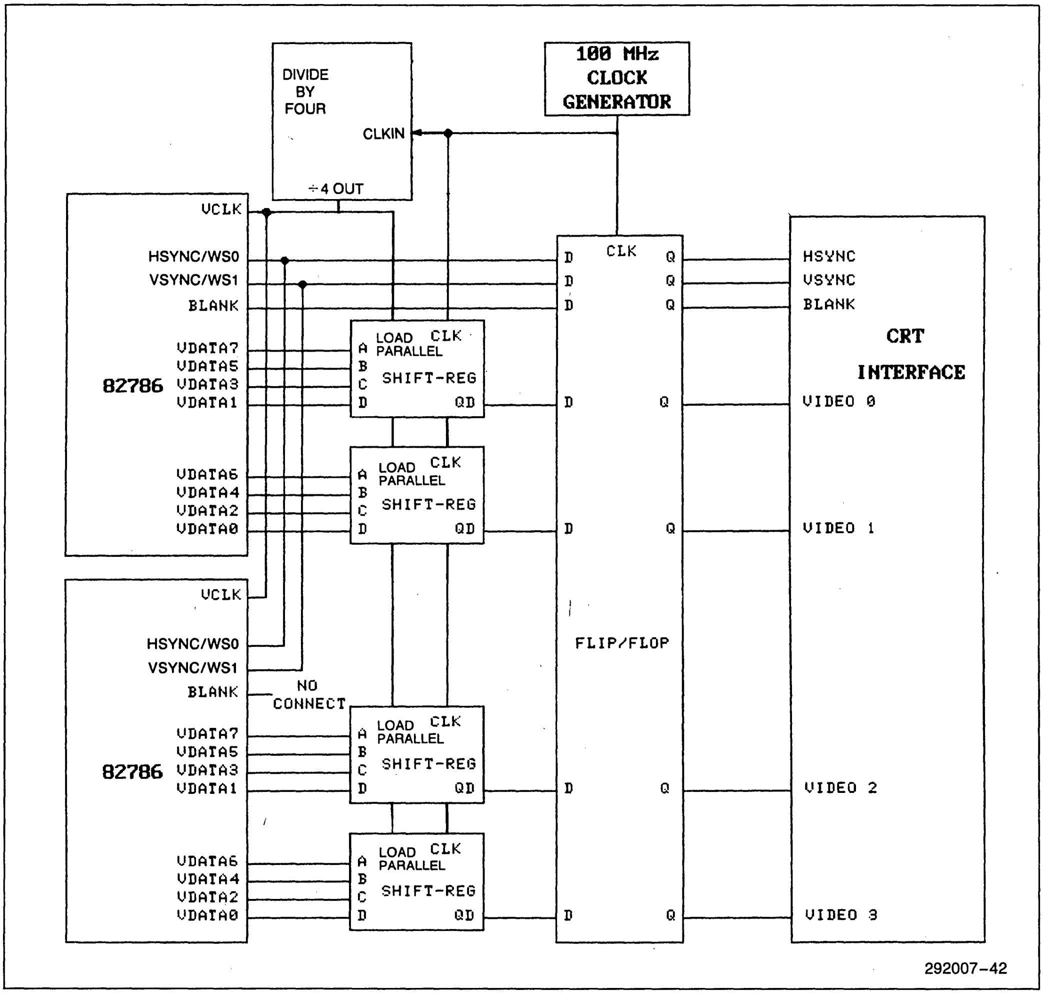


Figure 38. Two 82786s Can Generate 100 MHz Video with 4 Bits/Pixel

5.9 External Character ROM

Few 82786 applications will require, or even benefit from, the use of an external character ROM.

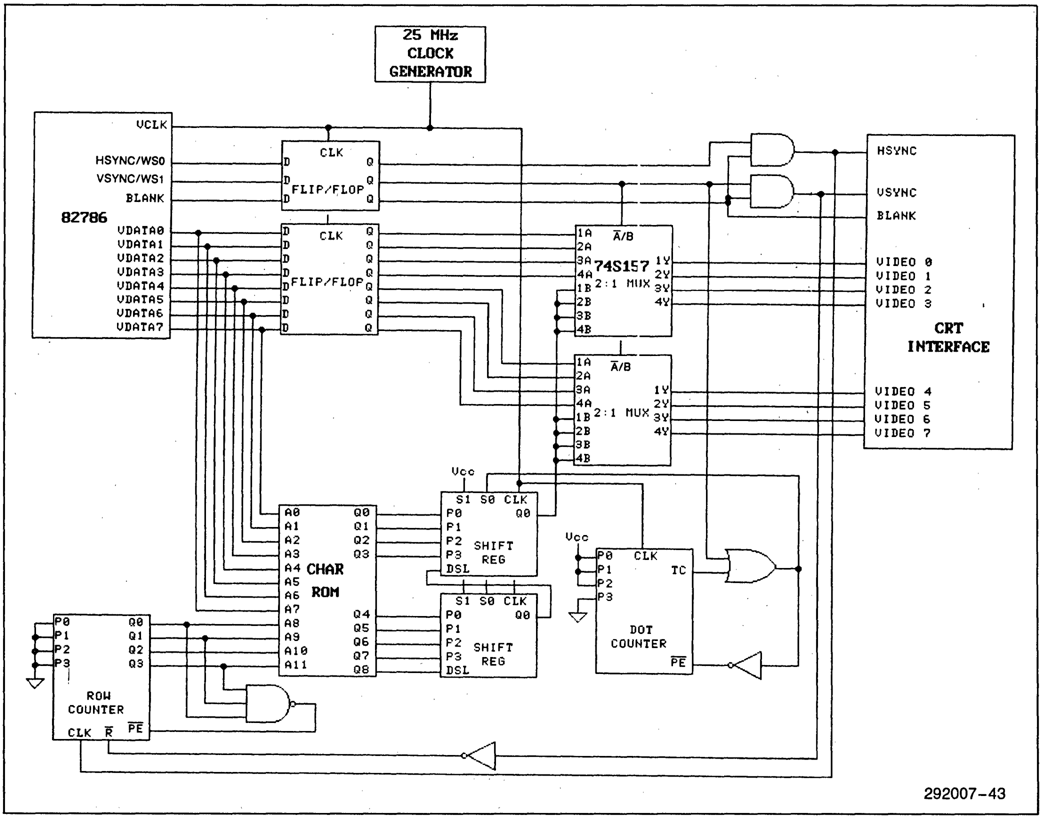
The 82786 Graphics Processor can very rapidly draw characters. It can fill an 80x25 character screen with highly detailed 16x16 characters in less than one tenth of a second.

The Graphics Processor is also very flexible in the way it draws characters. Characters may be:

- — formed from an unlimited number of character fonts
- — placed at any pixel on the screen
- — rotated in 4 directions with 4 paths
- — combined with graphics
- — drawn in any color
- — have transparent or opaque background

A character ROM display forces characters from a predefined font to be restricted to character-cell positions on the screen with few, if any, of the above flexibilities. For downward compatibility reasons, however, it may be necessary to provide the character ROM function in a 82786 system. Figure 39 illustrates a system capable of displaying both character ROM text and 82786 graphics. A multiplexer is used to switch between the character ROM output and the direct 82786 output. One of the window status bits is used to switch the multiplexer so both the character ROM and the 82786 graphics windows can be shown simultaneously on the same screen. It is important to delay the direct 82786 VDATA and window status signal the same number of clocks as the character-cell video so that all signals get to the multiplexer on the same clock. The extra D-flip/flops before the multiplexer are used to perform the needed delay.

The character ROM in Figure 39 is capable of displaying 256 characters using a 9x14 pixel character-cell. The characters are stored as an 8-bit pixels within a 82786 bitmap. To display the character, the window is programmed as an 8-bit/pixel bitmap with a horizontal zoom of 9 and a vertical zoom of 14. The 82786 will then place the 8-bit character code on its VDATA pins



292007-43

Figure 39. Support of Character-ROM and Bit-Mapped Graphics on Same Screen

during the scan lines when the character is to be displayed. The pixel counter is used to load the shift register every 9 pixels. This counter is synchronized to the beginning pixel of the window by starting when the window status pin falls. The row counter is used to supply row information to the character ROM. This counter is synchronized to the frame by starting from the end of the VSync pulse. Therefore, any character ROM window must begin at a multiple of 14 scan lines after VSync.

Another situation in which a character ROM display may be practical is if a very large character set is required. The Japanese Kanji characters are an example. The size of this character set is so large that it may be more practical to store the characters in a character ROM rather than load them from disk into the 82786 graphics memory. Figure 40 illustrates a configuration that can display up to 65536 characters from a very detailed (32x32) font. This circuit allows both text and graphics windows to be displayed on the screen simultaneously. One of the window status signals is used to select between text and graphics.

Such a character set requires a high resolution, generally monochrome display. The circuit in Figure 40 allows up to 200 MHz video (one bit/pixel) for very high resolution screens. The 82786 is programmed in super High-Speed Acceleration Mode as described in Section 5.6.

The character-codes to be displayed should be placed in one bit/pixel bit-maps with 16 consecutive bits for each character. The hardware combines the 8-bit VDATA values from two consecutive pixels to generate the 16-bit character-code for the Character-ROM. If less than 65536 characters are required, not all of the 16-bit character code addresses need be used for the character-ROM. Some of these bits may be used for attributes such as blinking and reverse video. The ROM contains a 32x32 character font, each character is split up into 32-lines of four 8-bit bytes. The "pane" counter selects one of the four 8-bit bytes at a time. The "row" counter determines the current row of the character.

Character cell windows should be zoomed by 2 horizontally and by 32 vertically. The window must be placed at a multiple of 4-pixels from HSync and a multiple of 32-lines from VSync. It is possible to place windows at non-multiples from HSync and VSync if the "pane" and "row" counter parallel inputs are tied to other than ground.

5.10 Combining the 82786 With Other Video Sources

It is possible to combine graphics output from the 82786 with output from other video sources such as broadcast TV, video recorders, and video laser disc players. The main requirement to perform such a feat is that both 82786 and the video source are locked in sync.

The 82786 has two independent Video Slave modes and HSync/VSync and Blank can be independently configured as outputs or inputs. When HSync/VSync are programmed as inputs, then they are still outputs during the active display period if the window status is enabled. External HSync/VSync reset the 82786 horizontal and vertical counters respectively.

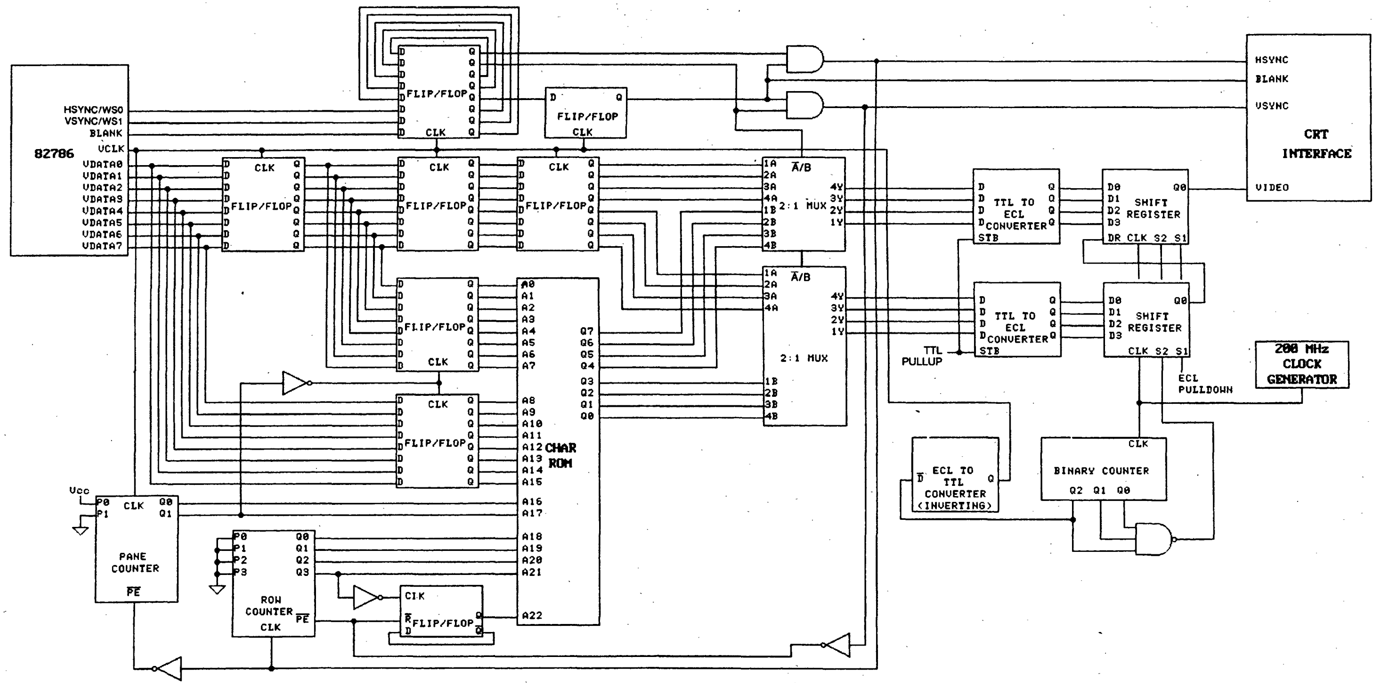
When Blank is configured as output, the active display period is determined by the programmed values of VFldStrt, VFldStp, HFldStrt, and HFldStp. When Blank is configured as an input, the external system determines the active display period. The internal video shift register generates video data only during the active display period.

HSync/VSync and Blank would normally be programmed as input/output as follows:

HSync/ VSync	Blank	Application
Output	Output	Normal display generated by 82786
Input	Output or	82786 generated display superimposed on externally-generated video or
Input	Input	Multiple 82786 systems

The 82786 sync timing registers should be programmed to be as close to the frequency of the video source as possible. The 82786 should also be programmed for slave video-sync. The sync signals from the video source must be converted into separate TTL-level horizontal and vertical sync and fed to the 82786 HSync and VSync pins. The 82786 will then automatically sync itself up to the video source by waiting for its HSync input to fall before each scan line and waiting for its VSync input to fall before beginning a new display field.

For many applications, the 82786 video clock can be derived directly from a crystal oscillator. Since the 82786 syncs up to the nearest pixel on every scan line, even video sources with imperfect timings, such as video recorders where speed variations are common, will produce an acceptable picture. The frame-to-frame deviation of the 82786 graphics information on the screen relative to the video source will never be more than one pixel.



292007-44

Figure 40. Support of Very Large Character-ROM and Bit-Map

For more demanding applications, the 82786 video clock can be synthesized directly from the video source timings using a phase-locked-loop circuit. The 82786 will still sync itself up every scan line, but now the relationship between HSync and the 82786 VCLK will remain constant. This implementation will create virtually no deviation between the 82786 graphics and the video source.

In the case of interlaced video, care must be taken to initially start the 82786 display just prior to the VSync before an even-field. The 82786 initialization software is responsible to guarantee that the first LD_ALL to start the 82786 display occurs sufficiently before the VSync during an odd-field so the first 82786 display field will match the video source even-field.

Once the 82786 is locked in sync with the video source, then the VDATA information from the 82786 can easily be combined with the video from the video source. Although the two video signals could be mixed on top of each other, probably the most common implementation is to switch between one or the other source. For example, the 82786 could create letters that are to be placed over the video picture. During the display scan, whenever a portion of a letter is to be displayed, the video from the 82786 can be switched in, otherwise the video source is switched in.

If the output of the video source is analog, the 82786 VDATA can be converted into an analog signal and an analog switch can be used. The state of the switch can be derived in a number of ways. If the switching is to be done on window boundaries, one window status pin can be used to control the switch. If the switching must be done within a window, a special graphics color code can be used to indicate that the 82786 video should be replaced with that from the video source. Possibly the color 11111111 could be placed on the VDATA pins and an 8-input NAND gate used to control the analog switch.

5.11 Other Types of Displays and Printers

The 82786 not only can be used with CRTs, but can also be used with other types of displays such as LCD, plasma, and intelligent printers. These devices have such a wide range of interface requirements that space does not permit each individual situation to be addressed in detail. Rather, some example requirements are discussed to illustrate how the 82786 can meet those needs.

PIXEL CLOCK RATE

The rate at which pixels are clocked into displays can vary immensely. The 82786 allows a very wide range of video clock frequencies from DC levels to 25 MHz to accommodate such devices. In addition, faster clock rates can be generated using the method described in Section 5.6

NO REFRESH

Printers and some displays are not required to be continuously refreshed. Needlessly running the 82786 Display Processor through refresh cycles steals bus bandwidth from the Graphics and other Processors. To eliminate this waste, the display can be turned off by resetting the DspOn bit (bit 0) in the Display Processor VStat Register (register number 0). When DspOn is reset, the Display Processor will continue to generate HSync, VSync, and Blank and place Default-VDATA on the VDATA lines, but no bus bandwidth will be required.

When a change to the display is required, the DspOn bit can be set using the LD_REG or LD_ALL command. Once the refresh starts, another LD_REG command to turn the display back off can be placed in the Display Processor Opcode Register. The Display Processor will then automatically execute it when the refresh is completed.

PARTIAL DISPLAY UPDATES

Some displays that do not require continuous refresh, do have a long update time. It may take several seconds to update every pixel on the display. For small changes to the display, such as adding each character as it is typed by the user, it may be much more feasible to update only the portion of the display which is affected.

Using the very flexible windowing capability of the 82786, it is possible to only scan through a specific portion of the display.

PIXEL ADDRESS GENERATION

Some displays, especially those which allow only partial display updates, require that pixel location addresses be generated along with the pixel data. Although external circuitry could be used to generate these addresses, the 82786 can be used to generate them directly. If a single 8-bit address is all that is required, the Default Video register can be programmed to a value that the VDATA pins will reflect during blank time. With proper programming of the sync timing registers, this value can be clocked into the display before each scan line using HSync.

More complex pixel addresses can be generated by using the 82786 windowing capability. By creating a thin window at the beginning of each scan line, one or more bytes of address information can be sequentially clocked out over the VDATA pins before each line.

ULTRA HIGH RESOLUTION

Because some displays use either slow refresh times or don't require refresh at all, it is possible to have very high resolutions. All of the display counters in the 82786 are 12 bits allowing up to a 4096 × 4096 display size (some of this resolution may not be available depending on the number of sync clock cycles required). Trading off bits/pixel for resolution, the Accelerated Modes can provide 2, 4, or 8 times this resolution horizontally, up to 32768 pixels.

Still, some applications, such as printers, may require even greater resolutions. This is possible with the 82786 using external counters to generate the HSync, VSync, and Blank inputs for the 82786. The 82786 should be programmed for slave video mode by setting the CRTMode Register (Display Processor Register 5, Blank Slave Mode and HSync/VSync Slave Mode, bits 2 and 3 should be set). Using all 16 horizontal windows, the horizontal resolution may be up to 4096 × 16 = 65536 pixels. Again, trading off bits/pixel for resolution, the Accelerated Modes can provide 2, 4, or 8 times this resolution, up to 524288 pixels. Vertically there is no limit to the resolution.

Such high resolutions require a lot of memory. For example, suppose a printer can generate 300 x 300 dots per square inch and is used on 8.5 x 11 inch paper. Assuming only one bit/pixel (no gray scale) the entire page would require:

$$\frac{300 \times 300 \times 8.5 \times 11}{8 \text{ bits}} = 1.05 \text{ Megabytes}$$

It may not be feasible to place this much memory into a printer. But it may be feasible to generate the display one strip at a time. Suppose that the first 300 lines are generated and printed. Once printed the next 300 lines can be generated and printed using the same memory. Now the memory required is only:

$$\frac{300 \times 300 \times 8.5}{8 \text{ bits}} = 96 \text{ Kbytes}$$

If the image to be printed can be described by a set of commands for the Graphics Processor, each strip can be very easily generated. The drawing bitmap and clipping rectangle are set for the first strip and the Graphics Processor then runs through the command list. Once completed the strip may be printed. Then the bitmap and clipping rectangle are set for the second strip and the Graphics Processor again traverses the same command list to generate the second strip.

If there is enough memory for two strips, double buffering can be used to pipeline the operation. While the Display Processor is busy printing one strip, the Graphics Processor can be generating the next strip. The same approach can be extended to multiple pages, even using more than one 82786.

5.12 Calculating the Video Parameters

The 82786 video Display Processor is programmable to afford a wide variety of display formats. To determine the display format(s) that one would like to generate, several parameters must be considered.

Application parameters: these are dependent on the needs of the specific application and must be chosen by the designer.

Hres—horizontal resolution — number of pixels per horizontal line. When using Accelerated Video Modes, Hres must be a multiple of Accel (following pages).

Vres—vertical resolution — number of vertical pixels (scan lines) per display

Vfreq—vertical frequency — rate at which CRT beam makes one pass from the top of the screen to the bottom. It is common to use 60 Hz but almost any other frequency can be generated by the 82786. US broadcast television standards use a 59.95 Hz rate. European video systems are based on a 50 Hz field rate. High resolution displays often use 40 Hz or lower. Slower rates reduce the speed requirements of the monitor and the 82786 video circuitry and also permit higher resolutions. However, slower rates also flicker more and may be intolerable to the operator. Generally, rates significantly under 60 Hz will tend to cause some perceptable flicker unless CRTs with long persistence phosphor are used.

ILC—interlacing — A non-interlaced display generates the entire display frame in one field scan. One method to double the resolution of a display is to use interlacing. Rather than use just one field to display all the information, two consecutive fields are used to create the entire display frame (Figure 41). Alternate scan lines are written during each alternate field. For TV-like pictures, where the image generally doesn't change drastically from one line to the next, interlacing allows a 30 Hz frame rate with a 60 Hz field rate without perceptable flicker. For detailed computer graphics, however, one line may change drastically from the next in color and/or intensity, in which case interlacing at such rates do cause perceptable flicker.

The 82786 supports both non-interlaced and interlaced displays. In addition, an interlaced-sync mode is available which generates sync signals in the manner used by interlaced displays, but generates the video signals in the manner used by non-interlaced displays (both fields identical). This permits interlaced screens with consecutive pairs of lines identical.

Monitor parameters: these are dependent on the specific requirements of the display monitor used.

Hblank—horizontal blanking time — the time required for the CRT beam to jump from the right side of the display back to the left and stabilize. This is also called horizontal retrace time and is the sum of the horizontal sync and front and back porch times (Figure 42). Monitors typically range from 5–12 μs.

Vblank—vertical blanking time — the time required for the CRT beam to jump from the bottom of the display back to the top and stabilize. This is also called vertical retrace time and is the sum of the vertical sync and front and back porch times (Figure 43). Monitors typically range from 600–1400 μs.

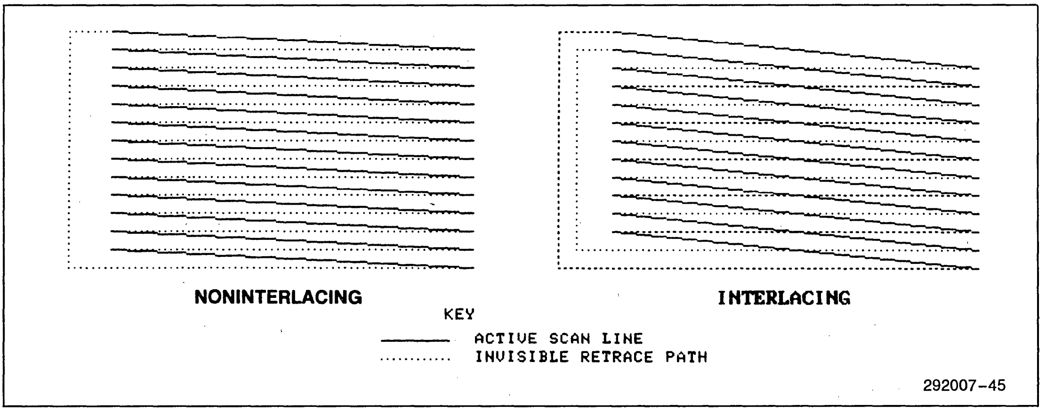


Figure 41. Non-Interlacing and Interlacing Displays

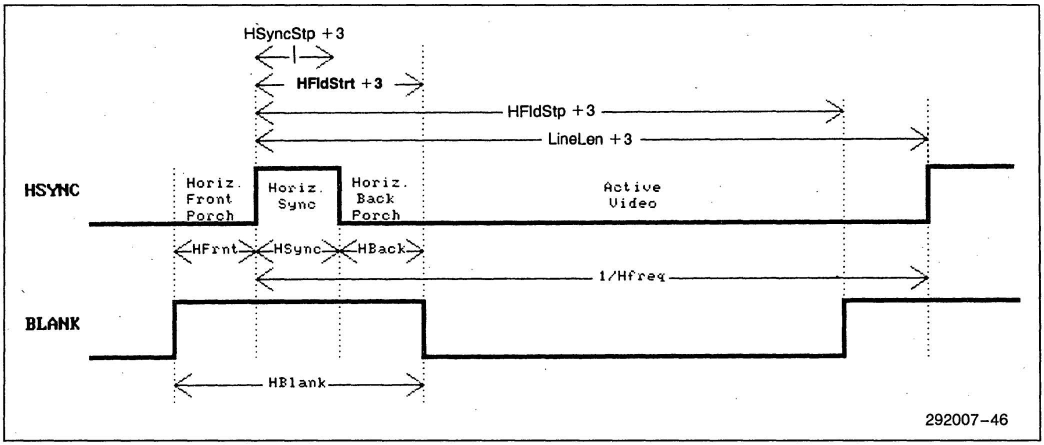


Figure 42. Horizontal Sync and Blank Timing Parameters

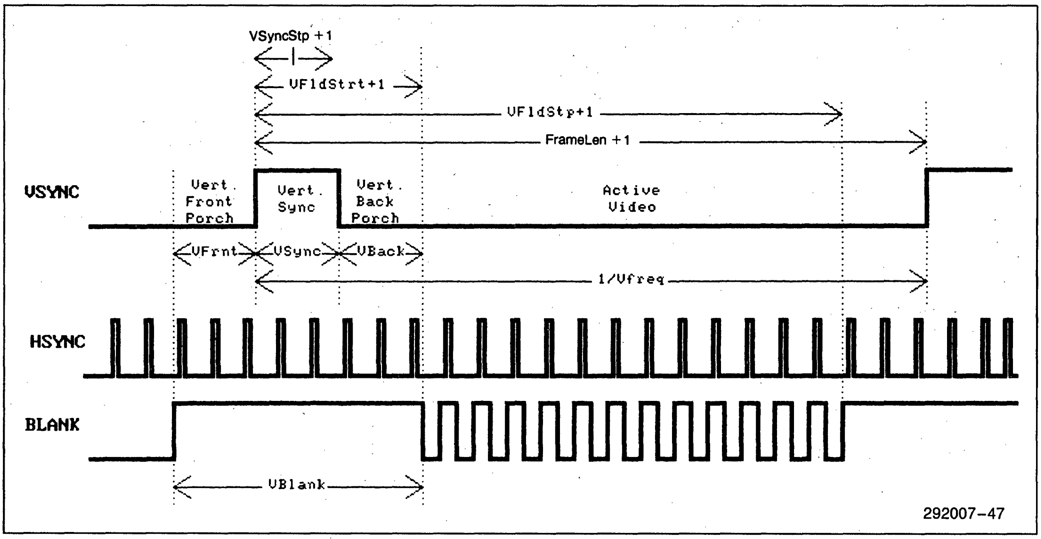


Figure 43. Vertical Sync and Blank Timing Parameters

Hfreq—horizontal frequency — the frequency at which horizontal lines are scanned. Monitors typically range from 15–36 kHz.

Vfreq—vertical frequency — the frequency at which the display field is scanned. Monitors typically range from 40–70 Hz.

BPP—bits per pixel — monitors with digital inputs restrict the number of usable bits/pixel. Monitors with analog inputs allow a virtually unlimited range of intensities with the use of Digital-to-Analog converters. This parameter is mainly dependent on the video interface hardware described in the previous sections.

Color monitors generally limit the perceivable horizontal and vertical resolution due to their shadow mask. See the specific monitor specifications for more details.

Video interface parameters: these are dependent on the 82786 component and the video interface logic.

VCLK—video clock frequency — the video input clock into the 82786. It has a maximum rate of 25 MHz and may be chosen so that the frequency evenly divides by both Hfreq and Vfreq.

Accel—82786 video acceleration — this parameter is determined by what mode the 82786 is used in. Normally Accel = 1. If the trade-offs mentioned in Section 5.6 are used to attain higher video rates at the expense of fewer bits/pixel, then the value for Accel should be 2, 4, or 8.

	Video Mode	Max DotClk	Programmed Accel Bits
Accel = 1	Normal	25 MHz	0 0
Accel = 2	High Speed	50 MHz	0 1
Accel = 4	Very High Speed	100 MHz	1 0
Accel = 8	Ultra High Speed	200 MHz	1 1

DotClk—pixel dot clock frequency — this is normally the same as VCLK. However, when accelerated video modes are used, this is either 2, 4, or 8 times VCLK.

$$\text{DotClk} = \text{VCLK} \times \text{Accel}$$

HSyncStp, HFldStrt, HFldStp, LineLen—these are values programmed into the 82786 Display Processor to determine the horizontal scan timing (Figure 42). They may be set to any value from 0 to 4095. Their values should also fit the formula:

$$\text{HSynStp} < \text{HFldStrt} < \text{HFldStp} < \text{LineLen}$$

VSyncStp, VFldStrt, VFldStp, FrameLen—these are values programmed into the 82786 Display processor to determine the vertical scan timing (Figure 43). They may be set to any value from 0 to 4095. Their values should also fit the formula:

$$\text{VSyncStp} < \text{VFldStrt} < \text{VFldStp} < \text{FrameLen}$$

Once the above parameters are evaluated, the video parameters can actually be calculated. The parameters interact quite heavily so that, for example, if a specific horizontal and vertical resolution at a specific field rate is required, the monitor frequencies and blank times may need to be altered. It may take several iterations to optimize all the parameters. The calculations can be performed by hand. However, a much easier way to manipulate these values is by using a spreadsheet program. A spreadsheet allows the parameters to be easily manipulated with their affects immediately displayed. A spreadsheet template for this purpose is given in Section 5.13.

The following formulas are used to determine the video parameters. Along with the formulas is an example calculation. For the example, let's generate a 640 × 400 × 8 bit/pixel (256 color) screen at 60 Hz non-interlaced. We will assume:

Hres	=	640 pixels
Vres	=	400 pixels
Vfreq%	=	60 Hz
Hblank%	=	12 μs
Vblank%	=	1300 μs
Accel	=	1 (no external acceleration)

Variables with a percent (%) after them represent desired values, the actual value will be calculated below.

ROUND(X) will be used to denote rounding off X to the nearest integer.

First, calculate the vertical resolution per field. Since our display is noninterlaced, the value is the same as the vertical resolution.

If interlaced then: VresFld = Vres/2
else: VresFld = Vres Vresfld
= 400 pixels

With interlaced screens, VresFld is half the vertical resolution. For example, with 525 lines, use 262.5 for VresFld.

Now, calculate the horizontal frequency required. Subtract the vertical Blank time from the vertical period and divide by the active vertical lines to obtain the horizontal period. Inverting all that gives the horizontal frequency.

$$\text{Hfreq\%} = \frac{1}{\text{Hperiod\%}} = \frac{\text{VresFld}}{(1/\text{Vfreq\%}) - \text{Vblank\%}}$$

$$= \frac{400}{(1/60) - 1300\ \mu\text{s}} = 26.03\ \text{kHz}$$

In a similar manner, calculate the pixel dot clock required.

$$\text{DotClk\%} = \frac{1}{\text{DotPeriod\%}} = \frac{\text{Hres}}{(1/\text{Hfreq\%}) - \text{Hblank\%}}$$

$$= \frac{640}{(1/26.03\ \text{kHz}) - 12\ \mu\text{s}} = 24.23\ \text{MHz}$$

And then calculate the actual 82786 VCLK. Since external acceleration circuits are not used in our example, it turns out to be the same as the DotClk.

$$\text{VCLK\%} = \text{DotClk\%} / \text{Accel} = 24.23\ \text{MHz} / 1 = 24.23\ \text{MHz}$$

Great, now all we need is a 24.23 MHz crystal is needed to generate VCLK. But since such a crystal is tough to find, try a 25 MHz crystal instead and see how it affects the rest of the parameters. First of all, the pixel dot clock changes.

$$\text{DotClk} = \text{VClk} \times \text{Accel} = 25.00\ \text{MHz} \times 1 = 25.00\ \text{MHz}$$

Now, see how many VCLK's are required for the horizontal blank time.

$$\text{HblankClks} = \text{ROUND}(\text{VCLK} \times \text{Hblank\%}) = \text{ROUND}(25\ \text{MHz} \times 12\ \mu\text{s}) = 300$$

Now, calculate the actual horizontal Blank time.

$$\text{Hblank} = \frac{\text{HblankClks}}{\text{VCLK}} = \frac{300}{25\ \text{MHz}} = 12\ \mu\text{s}$$

The actual horizontal period is then the time required to display one line of pixels plus the Blanking time.

$$\text{Hfreq} = \frac{1}{(\text{Hres} / \text{DotClk}) + \text{Hblank}}$$

$$= \frac{1}{(640 / 25\ \text{MHz}) + 12\ \mu\text{s}} = 26.60\ \text{kHz}$$

The number of horizontal lines per frame can now be calculated:

$$\text{VFrameLines} = \text{ROUND}(\text{Hfreq} / \text{Vfreq\%}) = \text{ROUND}(26.60\ \text{kHz} / 60\ \text{Hz}) = 443$$

If an interlaced display is used, VFrameLines should be rounded-off to the closest odd integer.

The number of scan lines determines the actual vertical frequency:

$$\text{Vfreq} = \text{Hfreq} / \text{VFrameLines} = 26.60\ \text{kHz} / 443 = 60.05\ \text{Hz}$$

Now that the major parameters are calculated and we are satisfied with them, we can break up the Blanking times into sync, front and back porch times. Typical monitor values are:

HSync = 2 μs VSync = 300 μs
HBack = 6 μs VBack = 800 μs

HSyncClks = ROUND (VCLK x HSYNC)
 = ROUND (25 MHz x 2 μs) = 50

HBackClks = ROUND (VCLK × HBack)
 = ROUND (25 MHz x 6 μs) = 150

VSyncClks = ROUND (Hfreq× VSYNC)
 = ROUND (26.6 kHz × 300 μs) = 8

VBackClks = ROUND (Hfreq × VBack)
 = ROUND (26.6 kHz × 800 μs) = 21

Now it's a simple matter to calculate the values for the eight 82786 Display Processor video timing registers.

HSyncStp = HSyncClks − 3
 = 50 − 3 = 47

HFldStrt = HSyncStp + HBackClks
 = 47 + 150 = 197

HFldStp = HFldStrt + (Hres / Accel)
 = 197 + (640 / 1) = 837

LineLen = HBlankClks + (Hres / Accel) − 3
 = 300 + (640 / 1) − 3 = 937

For noninterlaced displays:

VSyncStp = VSyncLines − 1
 = 8 − 1 = 7

VFldStrt = VSyncStp + VBackLines
 = 7 + 21 = 28

VFldStp = VFldStrt + Vres
 = 28 + 400 = 428

FrameLen = VFrameLines - 1
 = 443 − 1 = 442

For interlaced and interlace-sync displays:

VSyncStp = (VSyncLines × 2) − 1
VFldStrt = VSyncStp + (VBackLines × 2)
VFldStp = VRes Total
FrameLen = VFrameLines − 2

In the preceding formula, VResTotal equals the Vertical Resolution for Field1 plus the Vertical Resolution for Field2 as shown below.

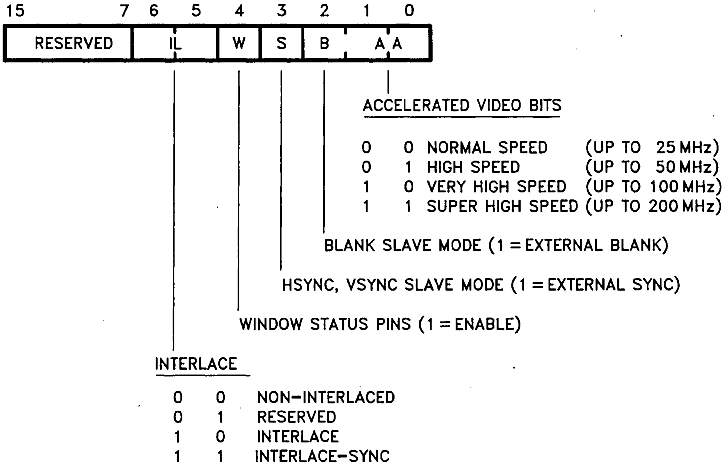
VResTotal = VResField1 + VResField2.

VFrameLines equals the total number of HSyncs in an entire frame.

Make sure LineLen > HFldStp and that FrameLen > VFrameLines. If not, your parameters are inconsistent and you should modify your requirements and re-calculate.

Finally, the bits for the CRTMode Register should be determined. For our example, non-interlaced mode is used and no accelerated video is required. Assuming the 82786 is used to generate the HSync, VSync and Blank signals and assuming the window Status pins are not used, the CRTMode registers should be loaded with all zeros.

CRTMode — Display Processor Register #5


15
7
6
5
4
3
2
1
0
RESERVED
IL
W
S
B
A A
ACCELERATED VIDEO BITS
0 0 NORMAL SPEED (UP TO 25 MHz)
0 1 HIGH SPEED (UP TO 50 MHz)
1 0 VERY HIGH SPEED (UP TO 100 MHz)
1 1 SUPER HIGH SPEED (UP TO 200 MHz)
BLANK SLAVE MODE (1 = EXTERNAL BLANK)
HSYNC, VSYNC SLAVE MODE (1 = EXTERNAL SYNC)
WINDOW STATUS PINS (1 = ENABLE)
INTERLACE
0 0 NON-INTERLACED
0 1 RESERVED
1 0 INTERLACE
1 1 INTERLACE-SYNC

292007-48

The host CPU software is required to load the values of the eight video timing registers and the CRTMode Register. Generally, this is done during system initialization. The registers should all be loaded simultaneously using the LD__ALL command rather than using individual LD__REG commands. This ensures that the video sync signals are never invalid while registers are being loaded.

Some CRTs can be permanently damaged by supplying the wrong sync frequencies to them. To prevent invalid video sync signals, the HSync, VSync, and Blank pins are tristated after RESET until the CRTMode Register has been written to.

5.13 A Spreadsheet for Calculating Video Parameters

As seen in the previous section, quite a number of calculations are required to determine the 82786 video parameter constants. Often several iterations through the calculations are required to optimize the display format. This process can be greatly simplified by using a spreadsheet.

An example of the output from such a spreadsheet is shown below. This example illustrates a 1290 x 968 x 4-bit/pixel (16 color) interlaced 60 Hz display. The user has supplied all of the values under the "DESIRED" column and the spreadsheet program has calculated the rest. The "ACTUAL" column shows the closest timings and parameters that the 82786 can actually supply. The "82786 DP REGISTER VALUES" shows the values that should be programmed into the Display Processor Registers to generate such a display.

The user can easily modify the "DESIRED" values until the "ACTUAL" values meet the application's needs. Care should be taken to ensure that all "ACTUAL" values are logically correct. If for example, any of the calculated parameters are negative, then the set of "DESIRED" parameters can not produce such a display, so some parameters must be adjusted.

```
              8 2 7 8 6    V I D E O    P A R A M E T E R S

Type under DESIRED column only: ACTUAL & REGISTER columns are calculated
------------------------------------------------------------------------

    PARAMETER               DESIRED  ACTUAL    82786 DP REGISTER VALUES
    ---------               -------  ------    ------------------------
Video Clock     VCLK (MHz):      25      25
Acceleration (1,2,4 or 8):        2       2
Interlacing (1 = no, 2 = yes):    2       2
Horiz Resolution (Pixels):     1290    1290
Vert. Resolution (Pixels):      968     968

Horiz Line Rate    (kHz):       --- 30.487    LineLen:        818
Horiz Sync Width   (µs):          2      2    HSyncStp:        48
Horiz Back Porch   (µs):          4      4    HFldStrt:       148
Horiz Front Porch  (µs):          1      1    HFldStp:        793

Vert. Frame Rate   (Hz):         60 59.956    FrameLen:      1015
Vert. Sync Width   (µs):        200  196.8    VSyncStp:        10
Vert. Back Porch   (µs):        400  393.6    VFldStrt:        34
Vert. Front Porch  (µs):        ---  213.2    VFldStp:       1002
```

The template follows. This template should be easily adaptable to nearly any spreadsheet program. This particular spreadsheet program uses @ROUND(X,0) to denote rounding to the nearest integer. If no rounding function is available in your spreadsheet program, you can substitute the integer function (which truncates the fractional portion to return the next lowest integer) for the round function:

substitute @INT(X+0.5) for @ROUND(X,0)

After entering the template into your favorite spreadsheet, you may wish to verify that it is working correctly by entering the "DESIRED" values of the above example and checking that the "ACTUAL" and "REGISTER" results match.

```
    ------------A------------ ---B--- ------------C------------- ---D--- -----------------E------------------
 1:            8 2 7 8 6   V I D E O   P A R A M E T E R S
 2: Type under DESIRED column only: ACTUAL & REGISTER columns are calculated
 3: ----------------------------------------------------------------------------------------------------------
 4:     PARAMETER              DESIRED  ACTUAL                   82786 DP REGISTER VALUES
 5:     ---------              -------  ------                   ------------------------
 6: Video Clock    VCLK (MHz):         +B6
 7: Acceleration (1,2,4 or 8):         +B7
 8: Interlacing (1=no, 2=yes):         +B8
 9: Horiz Resolution (Pixels):         @ROUND(B9/C7,0)*C7
10: Vert. Resolution (Pixels):         @ROUND(B10,0)
11:
12: Horiz Line Rate   (kHz):    ---    (C6*1000)/(E12+2)         LineLen:  @ROUND(C6*B15,0)+E15
13: Horiz Sync Width   (μs):           (E13+2)/C6                HSyncStp:  @ROUND(C6*B13,0)-3
14: Horiz Back Porch   (μs):           (E14-E13)/C6              HFldStrt: @ROUND(C6*B14,0)+E13
15: Horiz Front Porch  (μs):           (E12-E15)/C6              HFldStp:  +E14+(C9/C7)
16:
17: Vert. Frame Rate   (Hz):           (C8*C12*1000)/(E17+C8)    FrameLen:  @ROUND((C12*1000)/B17-(C8-1)/2,0)*C8-1
18: Vert. Sync Width   (μs):           ((E18+C8)*1000)/(C12*C8)  VSyncStp:  (@ROUND((C12*B18)/1000,0)-1)*C8
19: Vert. Back Porch   (μs):           ((E19-E18)*1000)/(C12*C8) VFldStrt: @ROUND((C12*B19)/1000,0)*C8+E18
20: Vert. Front Porch  (μs):    ---    (E17-E20)*1000/(C12*C8)   VFldStp:  +E19+C10
```

APPENDIX A SAMPLE INITIALIZATION CODE

Many registers within the 82786 must be initialized to configure the 82786 for the particular hardware environment it resides in. This appendix contains assembly language code to initialize the 82786 for one particular configuration:

— synchronous 10 MHz 80286 interface (Sections 4.2 and 4.3, Figure 18)

— one row of two interleaved banks of 51C256 Fast Page Mode DRAM (Section 3.3, Figure 9)

— 640 x 300 x 8-bit/pixel non-interlaced 60 Hz display, 25 MHz VCLK (Section 5.11, Figure 27)

All of the parameters to be initialized for this configuration are calculated under their corresponding sections in the body of this application note. To calculate the parameters for other configurations, refer to these sections.

This example of initialization code can be used to initially test many of the hardware functions. The code should create a stable display on the CRT. The display will consist of a black field which covers the entire screen (a 640 x 400 black rectangle). In the center of the rectangle is a 16 x 16 pixel arrow-shaped red and yellow cursor.

```
name Initialization82786

Memory82786 segment at 0C000h
                  ;segment located at start of CPU-mapped 82786 memory

          ;define locations of 82786 internal registers

                        org 0

Internalrelocation      dw      ?       ;BIU registers
Reserved                dw      ?
BIUControl              dw      ?
RefreshControl          dw      ?
DRAMControl             dw      ?
DisplayPriority         dw      ?
GraphicsPriority        dw      ?
ExternalPriority        dw      ?

                        org 20h

GPOpcode                dw      ?       ;Graphics Processor registers
GPLinkAddressLower      dw      ?
GPLinkAddressUpper      dw      ?
GPStatus                dw      ?
GPInstructionPtrLower   dw      ?
GPInstructionPtrUpper   dw      ?

                        org 40h

DPOpcode                dw      ?       ;Display Processor registers
DPParameter1            dw      ?
DPParameter2            dw      ?
DPParameter3            dw      ?
DPStatus                dw      ?
DefaultVDATA            dw      ?
```

```
          ;location of values for Display Processor LD_ALL instruction

                        org 80h

DPLdAllRegs label word

    dw    3        ;VStat:    turn on display and cursor
    dw    0FFh     ;IntMask:  mask all interrupts
    dw    20       ;TripPt:   trip point = 20 FIFO dwords
    dw    0        ;Frint:    cause interrupt every frame (interrupt is masked)
    dw    0        ;          reserved
    dw    0        ;CRTMode:  non-interlaced, non-accelerated, master sync&blank
    dw    47       ;HSyncStp: horizontal sync stop    :
    dw    197      ;HFldStrt: horizontal field start  :
    dw    837      ;HFldStp:  horizontal field stop   : 8 video timing registers
    dw    937      ;LineLen:  horizontal line length  : are programmed for
    dw    7        ;VSyncStp: vertical sync stop      : 640 x 400 at 60 Hz
    dw    28       ;VFldStrt: vertical field start    : with 25 MHz VCLK
    dw    428      ;VFldStp:  vertical field stop     :
    dw    442      ;FrameLen: vertical frame length   :
    dw offset WinDescl  ;DescAddrL:descriptor address pointer lower
    dw    0             ;DescAddrU:descriptor address pointer upper
    dw    0        ;(Reserved)
    db    0        ;ZoomY:    no vertical zoom
    db    0        ;ZoomX:    no horizontal zoom
    dw    0        ;FldColor: black field color
    dw    0FFh     ;BdrColor: white border color
    dw    0        ;Pad1BPP:  pad with zeros for 1 bit/pixel
    dw    0        ;Pad2BPP:  pad with zeros for 2 bits/pixel
    dw    0        ;Pad4BPP:  pad with zeros for 4 bits/pixel
    db    2        ;CursorPad:pad with red for cursor (yellow cursor in red box)
    db    80h      ;CsrStyle: opaque 16x16 block cursor, no window status
    dw    510      ;CsrPosX:  put cursor in middle of screen (horizontally)
    dw    220      ;CsrPosY:  put cursor in middle of screen (vertically)

    dw    0000000110000000b       ;CsrPat0:    create arrow-shaped cursor pattern
    dw    0000001111000000b       ;CsrPat1:
    dw    0000011111100000b       ;CsrPat2:
    dw    0000111111110000b       ;CsrPat3:
    dw    0001111111111000b       ;CsrPat4:
    dw    0011111111111100b       ;CsrPat5:
    dw    0111111111111110b       ;CsrPat6:
    dw    1111111111111111b       ;CsrPat7:
    dw    0000011111100000b       ;CsrPat8:
    dw    0000011111100000b       ;CsrPat9:
    dw    0000011111100000b       ;CsrPatA:
    dw    0000011111100000b       ;CsrPatB:
    dw    0000011111100000b       ;CsrPatC:
    dw    0000011111100000b       ;CsrPatD:
    dw    0000011111100000b       ;CsrPatE:
    dw    0000011111100000b       ;CsrPatF:

          ;location of strip descriptor list

WinDescL label word ;strip descriptor list
                        ;header of strip descriptor
        dw      399             ;lines in strip (400 covers entire screen)
        dw      0               ;lower link to next strip descr (there is none)
        dw      0               ;upper link to next strip descr (there is none)
        dw      0               ;number of tiles in strip (only one)
```

```
                    ;first (and only) tile descriptor
        dw      0               ;bitmap width (not applicable, this is field)
        dw      0               ;memory start lower addr (not applicable)
        dw      0               ;memory start upper addr (not applicable)
        dw      639             ;field width (640 covers entire screen)
        dw      0               ;fetch count (not applicable, this is field)
        dw      00001h          ;set field bit,use top,bottom,left,right borders

Memory82786 ends

Initialize82786 segment         ;code to initialize 82786

        mov     ax,seg BIUControl

        mov     ds,ax                           ;put 82786 register segment in ds

    assume cs:Initialize82786, ds:Memory82786

        mov byte ptr BIUControl, 30h            ;convert 82786 to 16-bit bus...
        mov byte ptr BIUControl+1, 0            ;...must use two 8-bit transfers

        mov     InternalRelocation, 01h         ;locate reg's at 82786 mem addr 0h

        mov     DRAMControl, 1Dh                ;1 row, interleaved 51C256 DRAM
        mov     RefreshControl, 18              ;request refresh every 15.2 uS

        mov     DisplayPriority,  110110b       ;set Display  FPL, SPL = 6

        mov     GraphicsPriority, 010010b       ;set Graphics FPL, SPL = 2
        mov     ExternalPriority, 100000b       ;set External FPL      = 4

        mov     DPParameter1, offset DPLdAllRegs ;address for LD_All command
        mov     DPParameter2, 0CH
        mov     DPOpcode, 5                     ;let DP perform LD_All command

        ret                                     ;end of initialization subrtn

Initialize82786 ends
```

If the constants in the CPU-mapped 82786 memory for the LD_ALL command and the Strip Descriptor list (in Memory82786 segment) cannot be loaded into 82786 memory by the system's program loader, they will have to be loaded by the initialization code. One method is to have the loader load them into CPU system memory and use a repeat-move-string command in the initialization code to move these constants into the 82786 graphics memory. Alternatively, it is possible to place these constants in the 82786-mapped CPU memory and allow the 82786 to fetch them using master-mode. This method, however, is not as efficient because the 82786 must re-fetch the Strip Descriptor list for every display frame.

The Graphics Processor is not used in this initialization code. To fully initialize the Graphics Processor, the following commands are required:

Def_Bit_Map	for all drawing and BitBlt commands
Def_Logical_Op	for all drawing and BitBlt commands
Def_Colors	if line/character drawing used
Def_Texture	if line drawing used
Def_Char_Set	if character drawing used
Def_Char_Orient	if character drawing used
Def_Char_Space	if character drawing used
Load_Reg	initialize stack pointer if macros used
Load_Reg	set poll-on-exception mask if used
Load_Reg	set interrupt mask if interrupts used

APPLICATION NOTE

AP-408

October 1987

An Introduction to Programming the 82786 Graphics Coprocessor

RAY TORRES
APPLICATIONS ENGINEER

Order Number: 240048-001

RELATED DOCUMENTATION

This software applications note should be used with the 82786 User's Manual (Order Number: 231933-002).

Other documentation available for the 82786 includes: Hardware Configuration Applications Note (Order Number: 292007-003), The 82786 Architectural Overview (Order Number: 122711-003), The 82786 Data Sheet (Order Number 231676-003), and 82786 Design Example—Interfacing to the IBM PC/AT (Order Number: 240049-001).

CHAPTER 1 INTRODUCTION

1.0 INTRODUCTION

This application note shows, by example, how to program the 82786. These software interface examples are written for an Intel 82786-based graphics board as described in the Application Note: 82786 Design Example-Interfacing to the IBM PC/AT. However, the concepts presented in these examples can be applied to any system using the 82786. With the appropriate modifications, these programs will run on other 82786 systems. Contact your nearest Intel Sales Office for more information about availability of 82786 graphics boards and availability of machine-readable copies of the software presented in this Application Note.

Chapter 2 presents an overview of the programmers model of the 82786.

Chapter 3 presents an 80286 Assembly Language example. The **objectives** of this example program are:
1) Initialize the 82786 registers,
2) Program the Display Processor (DP) for one full-screen window,
3) Draw a simple graphics image using the Graphics Processor (GP).

Chapter 3 also suggest several modifications to the Example Program as exercises for the reader. Solutions to the exercises are provided in the appendix. By working through these exercises, the reader gains an understanding of the concepts of programming the 82786.

Chapter 4 provides a Quick Reference Section, containing information frequently used by 82786 programmers.

1.1 Hardware System Requirements

Hardware system requirements to run the programming examples:

(1) An 82786 graphics board as described in the Application Note: 82786 Design Example-Interfacing to the IBM PC/AT.

(2) 6 MHz or 8 MHz—IBM AT computer.

NOTE:

(3) The Intel Evaluation Board cannot be used in a computer in which the EGA Graphics Adapter is installed. (For your text display, use the Monochrome Adapter or CGA adapter.)

Any other peripheral device that uses the A-segment of CPU address space or CPU addresses C4400–C4474 cannot be used with the 82786 Evaluation Board.

(4) NEC Multisync monitor (Model No: JC-1401P3A) or
SONY Multiscan monitor (Model no: CPD-1302)

You may need to adjust the monitor controls for vertical and horizontal hold, size, position, etc.

Settings for the NEC Monitor:

Set the switches on the rear of the NEC Multisync monitor as follows:
(1) Set the "MANUAL" switch to "ON".
(2) Set the TTL-ANALOG switch to "TTL".
(3) Set DIP switch 5 to "ON".
Set DIP switch 6 to "OFF".

Settings for the SONY Monitor:

Set the Digital-Analog switch to "DIGITAL".

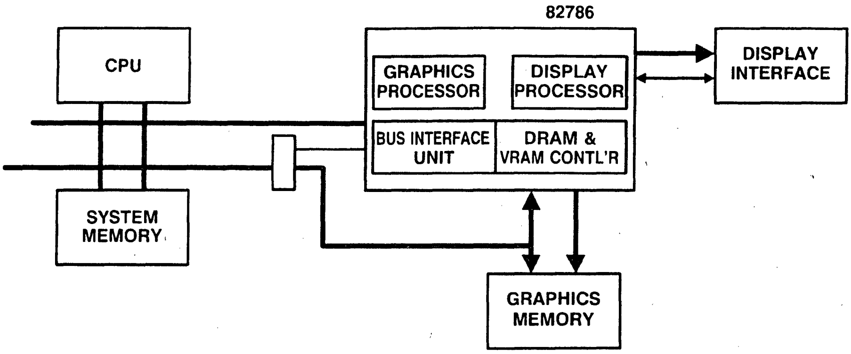
CHAPTER 2 PROGRAMMER'S MODEL OF THE 82786

2.0 INTRODUCTION

This Chapter presents an explanation of the programmer's model of the 82786. There are 5 sections in this chapter:
2.1) Overview
2.2) Graphics Processor
2.3) Display Processor
2.4) Bus Interface Unit
2.5) Summary

2.1 Overview

PROGRAMMING MODEL



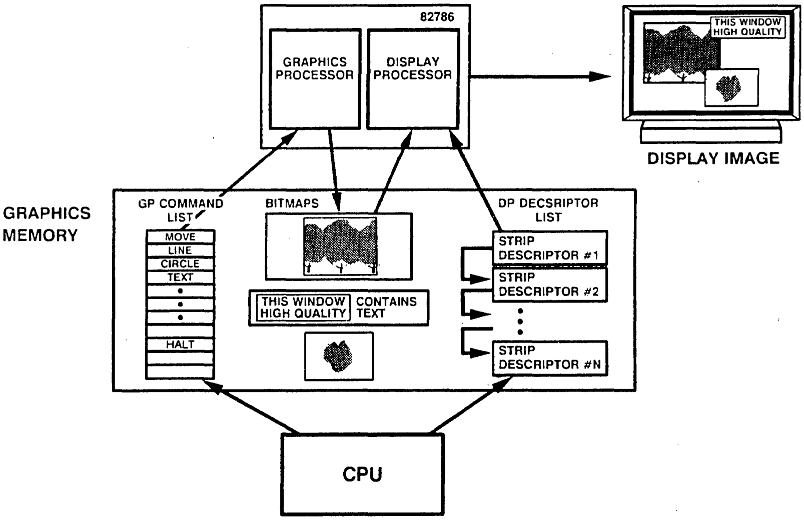
- ➡ OVERVIEW
- GRAPHICS PROCESSOR (GP)
- DISPLAY PROCESSOR (DP)
- BUS INTERFACE UNIT (BIU)

240048-1

Here is a block diagram of a typical 82786 system. The Display Processor and Graphics Processor are programmed independently. The Bus Interface Unit has programmable priority levels to control bus arbitration between the DP, GP, Host CPU, and DRAM refresh.

The Host CPU can write directly to the 82786 registers and directly into graphics memory.

82786 PROGRAMMING MODEL

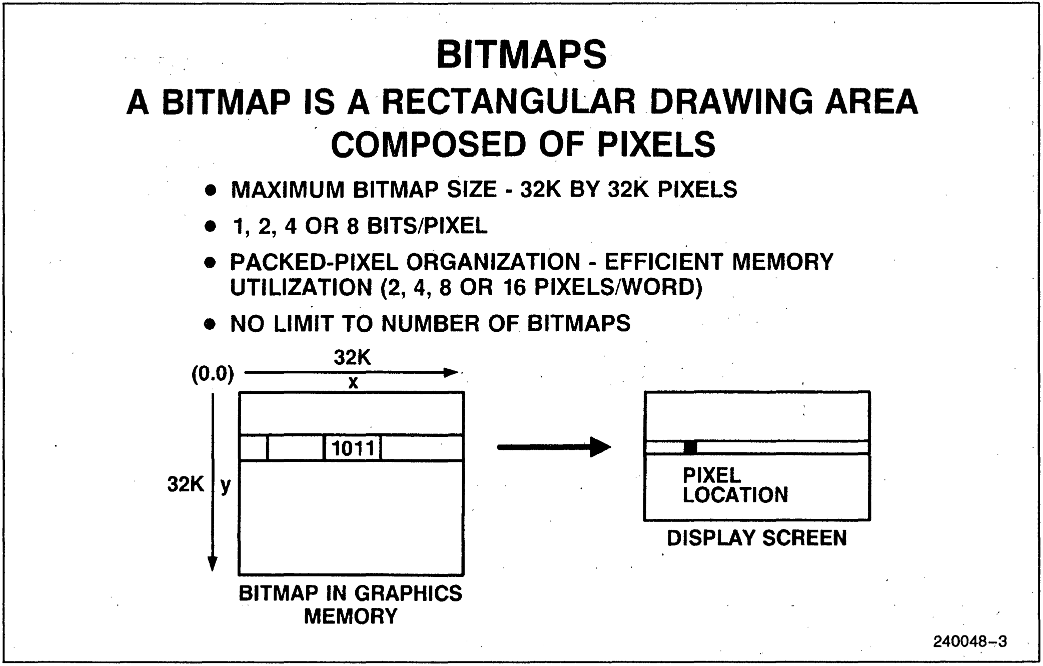


240048-2

To program the Graphics Processor, the host CPU writes a GP command list into graphics memory. Then, the GP executes the command list, drawing geometric shapes and text into the bitmaps in graphics memory.

To program the Display Processor, the host CPU writes a Screen Descriptor List into graphics memory. The DP reads the Descriptor List and sends graphics data, in the desired format, from the bitmaps to the display device. The DP can simultaneously display data from many different bitmaps. This is called Hardware Windows. Hardware Windows provides window movement, scrolling, and spanning and allows instantaneous changes in window content and screen format.

2.2 Graphics Processor Programming



BITMAPS

A BITMAP IS A RECTANGULAR DRAWING AREA COMPOSED OF PIXELS

- MAXIMUM BITMAP SIZE - 32K BY 32K PIXELS
- 1, 2, 4 OR 8 BITS/PIXEL
- PACKED-PIXEL ORGANIZATION - EFFICIENT MEMORY UTILIZATION (2, 4, 8 OR 16 PIXELS/WORD)
- NO LIMIT TO NUMBER OF BITMAPS

240048-3

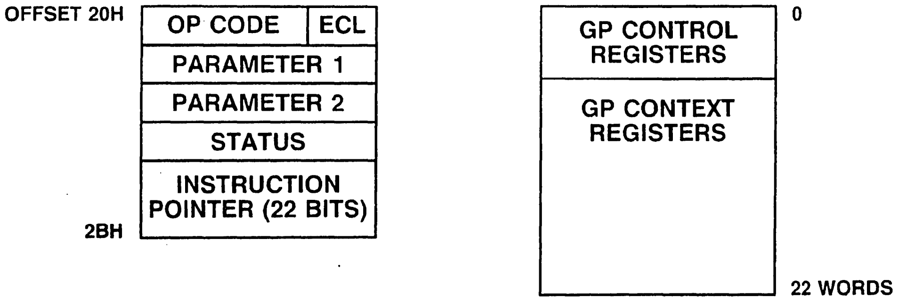
A **bitmap** can be thought of as a rectangular drawing area composed of pixels. Bitmaps are located in graphics memory.

The 82786 supports:
— VERY LARGE bitmaps, up to 32K x 32K.
— Flexible color capacity: 1,2, 4, or 8 bits/pixel providing 2, 4, 16, or 256 colors
— Packed pixel organization allows for efficient memory utilization
— Unlimited number of bitmaps, limited only by amount of available graphics memory.

GRAPHIC PROCESSOR REGISTERS

DIRECT ACCESS REGISTERS

INDIRECT ACCESS REGISTERS



OFFSET 20H

OP CODE | ECL

PARAMETER 1

PARAMETER 2

STATUS

INSTRUCTION POINTER (22 BITS)

2BH

GP CONTROL REGISTERS

GP CONTEXT REGISTERS

0

22 WORDS

- GP INTERNAL REGISTERS

- GP CONTROL REGISTERS
- GP CONTEXT REGISTERS
- DUMP__REG, LOAD__REG

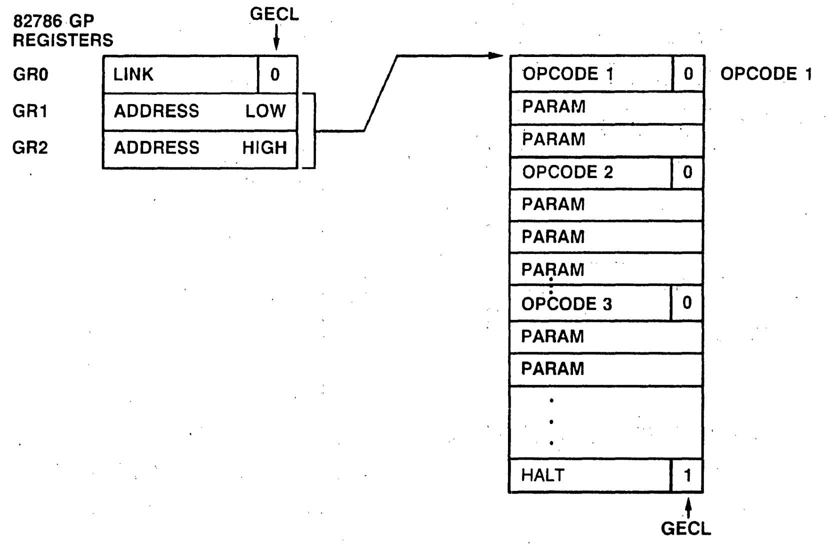
240048–4

Overview of Graphics Processor Registers

The Graphics Processor has 2 sets of registers: directly accessible and indirectly accessible.

The directly accessible registers include:
An Opcode register, two parameter registers, a Status Register, and an Instruction Pointer.

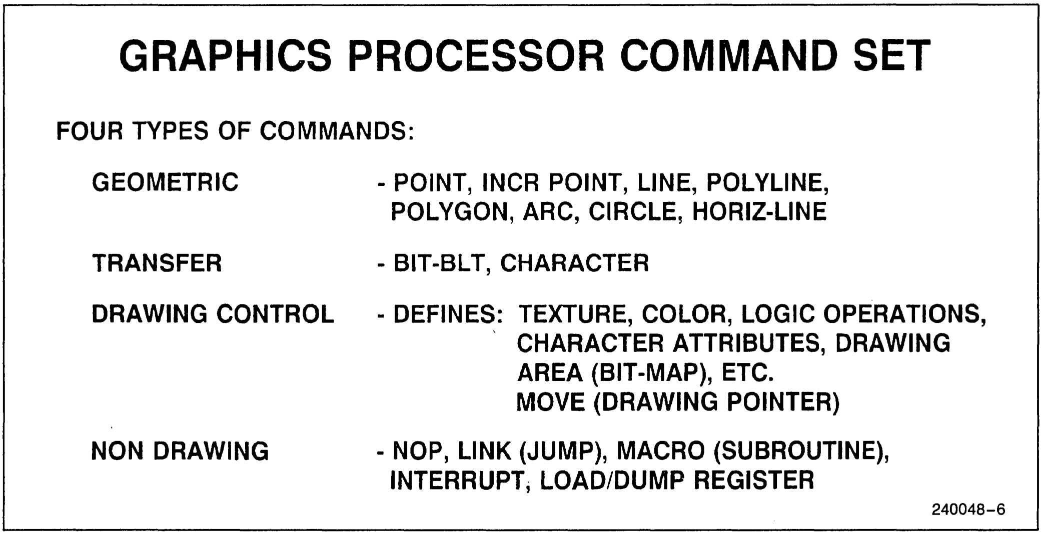
The indirectly accessible registers include the GP Control registers and the Context Switching registers used in multi-tasking systems. The indirectly accessible registers are loaded with the LOAD__REG command and read with the DUMP__REG command.



GRAPHICS PROCESSOR COMMAND LIST

240048-5

The graphics processor command list is composed of a sequence of Graphics opcodes and parameters. This command list is written into graphics memory by the host CPU. The GP begins execution of the command list when the host CPU writes a LINK instruction and the address of the command list into the GP registers. The GP halts execution when it reaches the HALT instruction.



GRAPHICS PROCESSOR COMMAND SET

FOUR TYPES OF COMMANDS:

GEOMETRIC	- POINT, INCR POINT, LINE, POLYLINE, POLYGON, ARC, CIRCLE, HORIZ-LINE
TRANSFER	- BIT-BLT, CHARACTER
DRAWING CONTROL	- DEFINES: TEXTURE, COLOR, LOGIC OPERATIONS, CHARACTER ATTRIBUTES, DRAWING AREA (BIT-MAP), ETC. MOVE (DRAWING POINTER)
NON DRAWING	- NOP, LINK (JUMP), MACRO (SUBROUTINE), INTERRUPT, LOAD/DUMP REGISTER

240048-6

Overview of Graphics Processor commands.

The Graphics Processor has 4 types of commands:
— Geometric drawing commands
— Transfer commands
— Drawing Control
— Non-drawing commands.

The GP commands provide a CGI-like graphics interface. These graphics primitives are extremely fast, since they are implemented in hardware. The Geometric commands provide primitives for POINT, LINE, ARC, and CIRCLE. The INCREMENTAL__POINT, POLYLINE, POLYGON, and HORIZONTAL__LINE (SCAN__LINES) commands can draw many points or lines with only one GP command for maximum efficiency. The SCAN__LINES command is used for Area Fill.

The GP Transfer commands provide high-speed BLOCK DATA TRANSFER and Text CHARACTER support.

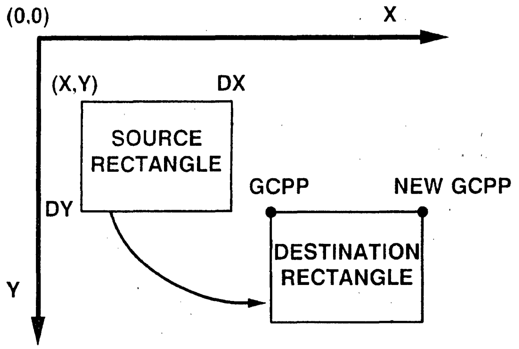
The Drawing Control commands provide settings for COLOR, TEXTURE, LOGICAL OPERATOR, DEFINING BITMAPS, CLIPPING RECTANGLE, AND CHARACTER ATTRIBUTES.

The Non-drawing commands provide LINK, MACRO (SUBROUTINE) CALL and RETURN commands, as well as an INTERRUPT and LOAD/DUMP REGISTER commands.

The LINE and CIRCLE commands are implemented by Breshenham's Algorithm (in a state machine).

BIT BLOCK TRANSFER COMMAND

⋮	
BIT-BLT	0
SOURCE X	
SOURCE Y	
DX	
DY	
⋮	
HALT	1

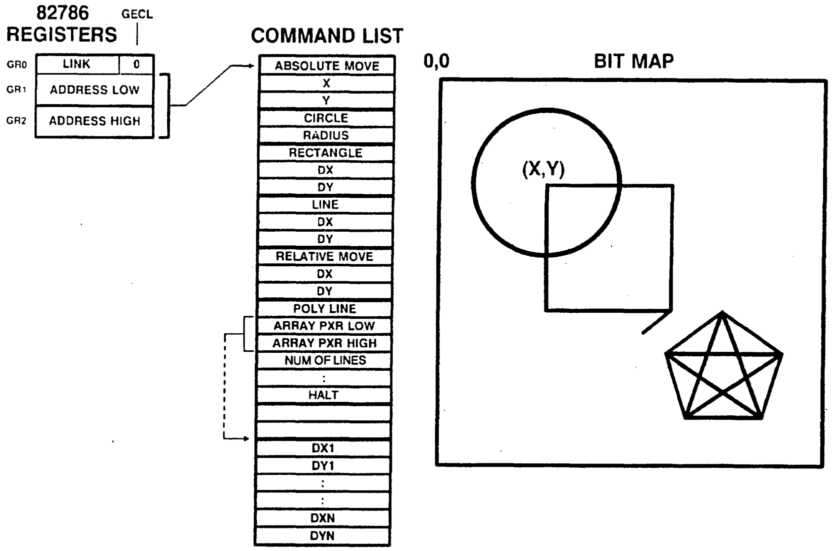


BIT BLOCK TRANSFER

240048-7

Here is an example of a Graphics Processor command, showing the format of the Bit_Blit (Bit Block Transfer) command. The opcode comes first followed immediately by its associated parameters, the Source X and Y co-ordinates and the width (dx) and height (dy). This command copies a block of data to the destination indicated by the Graphics Current Position Pointer (GCPP).

GRAPHICS PROCESSOR DRAWING EXAMPLE



240048-8

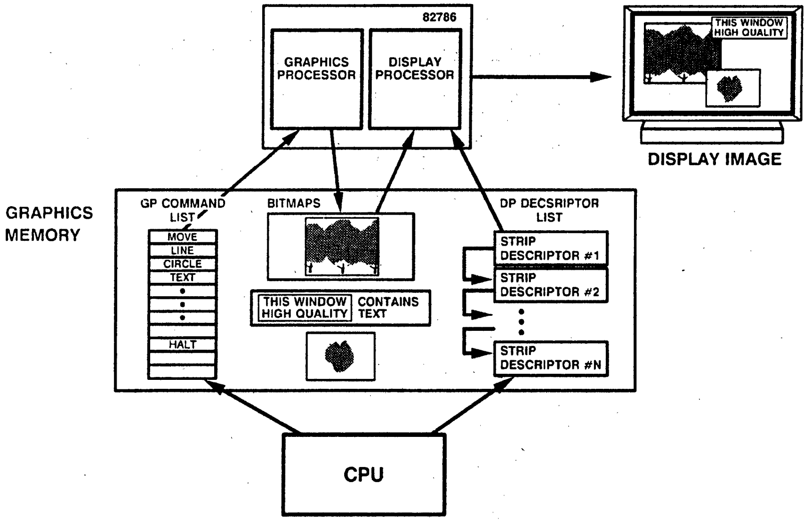
This Figure shows a specific example of a GP command list and its resultant drawing in the bit map. This demonstrates the capability of the Graphics Processor and how easy it is to create a drawing using the built-in graphics commands.

This GP command list example shows the ABSOLUTE__MOVE, CIRCLE, RECTANGLE, LINE, and POLYLINE commands.

First, the CPU writes the command list into graphics memory. The GP command list is executed when its address and the LINK instruction is written into the GP opcode registers.

The ABSOLUTE__MOVE instruction moves the Position Pointer to the given (x, y) coordinate, the CIRCLE command draws the circle with the given radius, the RECTANGLE command draws a rectangle with the given width and height, the LINE command draws a line with the given offset for the endpoint. The POLYLINE command draws a series of lines with only one GP command. The parameter for a POLYLINE command is a pointer to an array of endpoints for several lines.

82786 PROGRAMMING MODEL



240048-2

In **summary,** to program the 82786 GP: first the CPU writes a command list into graphics memory, as shown here. We have seen the details of the command list structure and details of some of the GP commands.

The CPU instructs the GP to execute a command list by first writing the address of the command list into the GP registers GR1 and GR2, and then writing the LINK opcode into the GP Opcode Register, GR0.

The graphics processor then executes the command list and draws the images into the bitmaps.

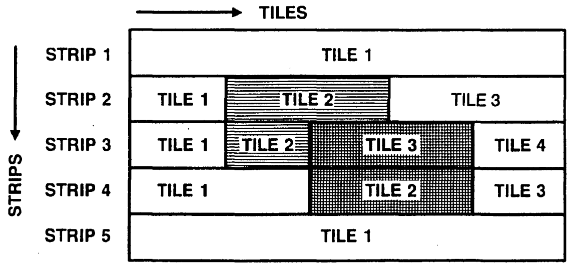
2.3 Display Processor Programming

This section describes general concepts of programming the Display Processor. As mentioned earlier, the DP reads a Screen Descriptor List that was written in graphics memory by the host CPU. This descriptor list determines how graphics data contained in the bitmaps is displayed on the screen in windows.

82786 SCREEN CONFIGURATION

BASIS OF HARDWARE WINDOW

- SCREEN IS DIVIDED INTO STRIPS
- EACH STRIP HAS SEVERAL TILES (MAX 16/STRIP)
- EACH TILE CAN DISPLAY DATA FROM DIFFERENT BIT MAP
- EACH TILE CAN HAVE DIFFERENT DEPTH (BITS/PIXEL)
- ANY TILE CAN BE ZOOMED (PIXEL REPLICATION)

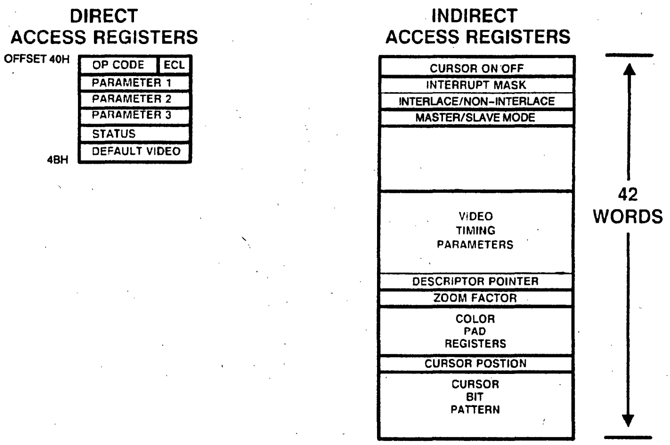


240048-10

Explanation of Display Processor Screen Descriptor List.

The 82786 uses a flexible and powerful method for describing the screen composition. The screen is described in terms of Strips, each strip is composed of Tiles. Each tile can display data from a different bitmap of a different depth (bits/pixel). Each tile may be zoomed independently. The screen format can be completely changed every frame refresh cycle. (This is typically every 1/60 second.)

DISPLAY PROCESSOR REGISTERS



- DP INTERNAL REGISTERS

- DISPLAY CONTROL BLOCK REGISTERS
- ACCESS BY LOAD, DUMP REGISTER COMMANDS

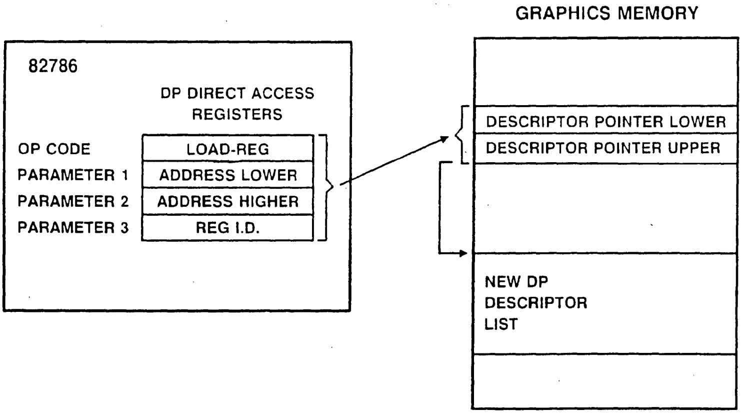
240048-11

The Display Processor has 2 sets of registers: directly accessible and indirectly accessible.

The Directly accessible registers include: An Opcode register, three parameter registers, and a Status Register.

The Indirectly accessible registers are also known as the DP Control Block Registers. These registers contain parameters for controlling DP operations such as the Video Timing Signals, location of the Descriptor list, cursor position, cursor pattern, etc. The indirectly accessible registers are loaded with the DP LOAD and DUMP commands.

HOW TO CHANGE SCREEN FORMAT



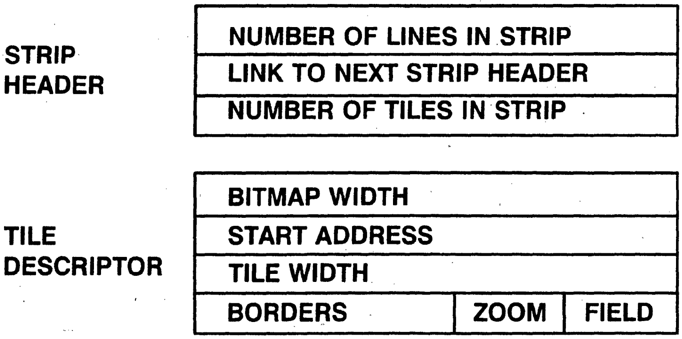
240048-12

The screen format is changed by writing a new Descriptor list into graphics memory or modifying a copy of the current descriptor list. Next, a pointer to the new descriptor list is written into graphics memory. Lastly, the address of the pointer, the LOAD__REG command, and register ID (0E Hex for Descriptor Address Pointer) are written into the DP parameter and opcode registers.

This is all that is necessary to change the screen format. The new screen appears during the next screen frame.

HOW TO DEFINE THE STRIP

DISPLAY PROCESSOR DESCRIPTOR LIST



STRIP HEADER

NUMBER OF LINES IN STRIP

LINK TO NEXT STRIP HEADER

NUMBER OF TILES IN STRIP

TILE DESCRIPTOR

BITMAP WIDTH

START ADDRESS

TILE WIDTH

BORDERS | ZOOM | FIELD

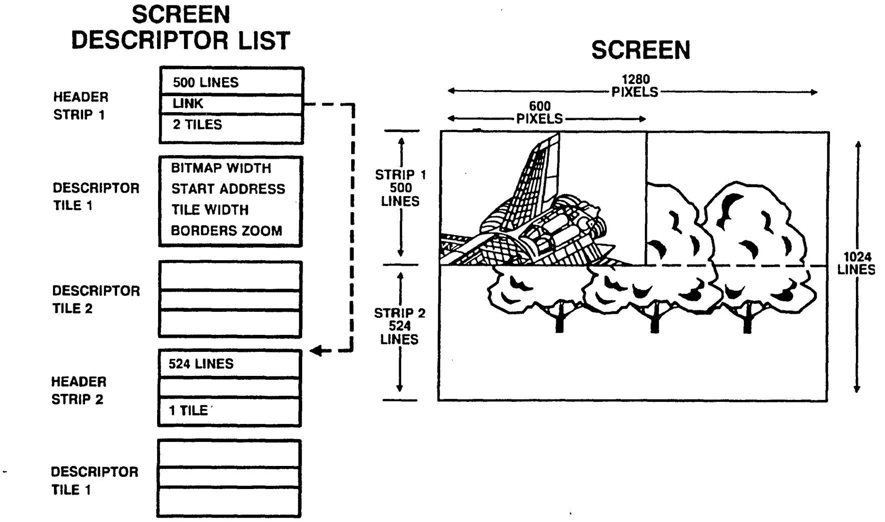
240048–13

As mentioned earlier, a Screen Descriptor List is composed of Strip and Tile descriptors. Here, we see an overview of a strip and tile descriptor. See Figure 3.2 for a more detailed diagram of a strip and tile descriptor.

The Strip descriptor contains the number of lines in the strip, link to the next strip descriptor, and the number of tiles in the strip.

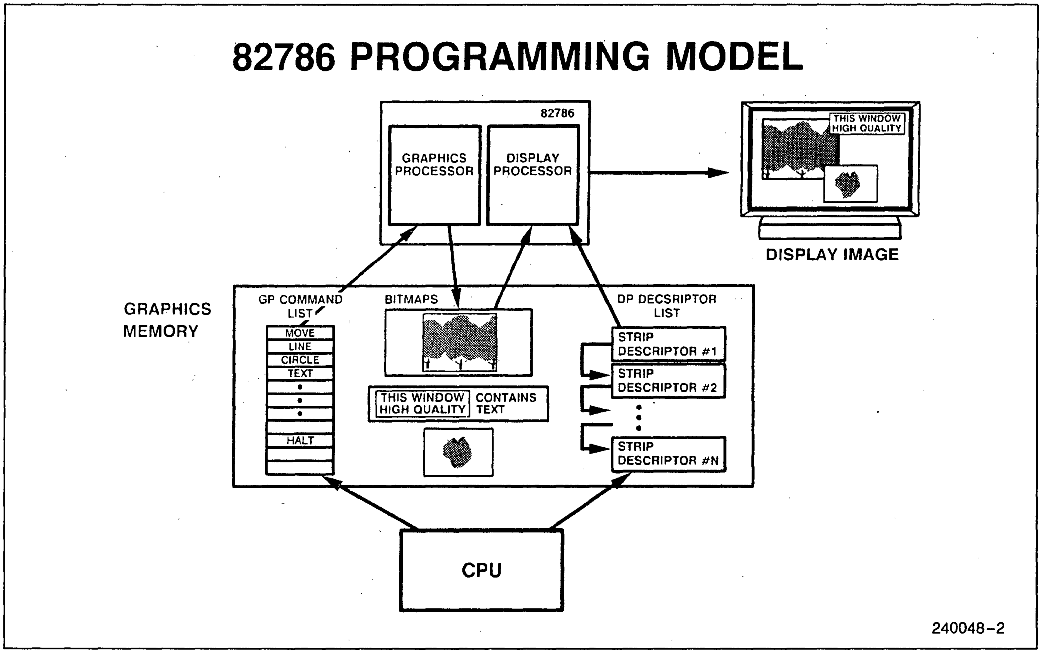
The Tile Descriptor contains the width of the source bitmap, the starting address of graphics data to be displayed, the tile width, and settings for turning borders, zoom and field color on or off. Each tile has its own tile descriptor.

SCREEN CONFIGURATION EXAMPLE



240048-16

Here, we see an example of a Descriptor List and its resultant display screen. The first strip, containing 500 lines vertically, is composed of two tiles. The second strip, containing 524 lines vertically, is composed of one tile.



In summary, to program the 82786 DP: first the CPU writes a DP descriptor list into graphics memory, as shown here. We have seen the details of the descriptor list. The display processor reads this descriptor list to determine how graphics data contained in the bitmaps is displayed on the screen in windows.

The screen format may be changed by simply writing a new descriptor list into graphics memory and changing the Descriptor Pointer to point to the new Descriptor List.

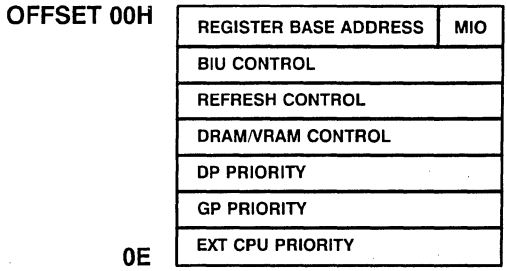
2.4 Bus Interface Unit

This section describes an overview of programming the Bus Interface Unit.

The Bus Interface Unit is programmable and controls the following functions:

- The base address for access of the 82786 registers
- The Graphics Memory Configuration
— VRAM/DRAM type
— Memory Access Mode
— Bank Configuration
— DRAM Refresh frequency.
- Memory Access Priority
— Sophisticated Bus Access Arbitration
— 8 Priority Levels

BUS INTERFACE UNIT REGISTERS



OFFSET 00H
REGISTER BASE ADDRESS | MIO
BIU CONTROL
REFRESH CONTROL
DRAM/VRAM CONTROL
DP PRIORITY
GP PRIORITY
0E EXT CPU PRIORITY

- **SYSTEM CPU/MEMORY INTERFACE PROGRAMMING**
- **GRAPHICS MEMORY CONFIGURATION**
- **MEMORY ACCESS PRIORITY**

240048-18

Programming the BIU is simple and straightforward. The programmer must simply write the correct values into each of the seven BIU registers. After these registers have been set, they do not need to be changed unless the chip is reset. The GP, DP and CPU priorities may be changed at any time, if desired.

2.5 Summary

This concludes the overview of the 82786 programming model. We have seen an overview of the powerful Graphics commands and how these commands are used.

We also talked about the concepts of programming the Display Processor and how to use the powerful hardware windowing capabilities of the 82786.

Chapter 3 provides a specific programming example and more specific programming details.

CHAPTER 3 EXAMPLE PROGRAM

3.0 INTRODUCTION

This Chapter presents an example 82786 program written in 80286 Assembly Language. The objectives of the Example program are:
1) Initialize the 82786 registers
2) Program the Display Processor (DP) for one full-screen window
3) Draw a simple graphics image using the Graphics Processor (GP).

Section 3.1 presents an overview of the program. Section 3.2 presents a detailed explanation of the program, section by section. Section 3.3 presents the complete source-code listing.

3.1 Overview Of Example Program

3.1.0 PROGRAM OUTLINE

Constant Definitions

— Special Addresses
— DP Opcodes
— GP Opcodes

Register Segment

— Define 82786 Internal Register Block Addresses

Data Segment

— Define DP Control Block Register Values
— Define DP Descriptor List
— Define GP Command List

Code Segment

— BIU Initialization—Load BIU Registers
— Clear Page 0 of Graphics Memory
— Copy DP Control Block Registers from CPU Memory to Graphics Memory
— Copy DP Descriptor List from CPU Memory to Graphics Memory
— Copy GP Command List from CPU Memory to Graphics Memory
— Start DP by Loading DP Control Block Registers
— Execute GP Command List to Draw Image

Program Outline

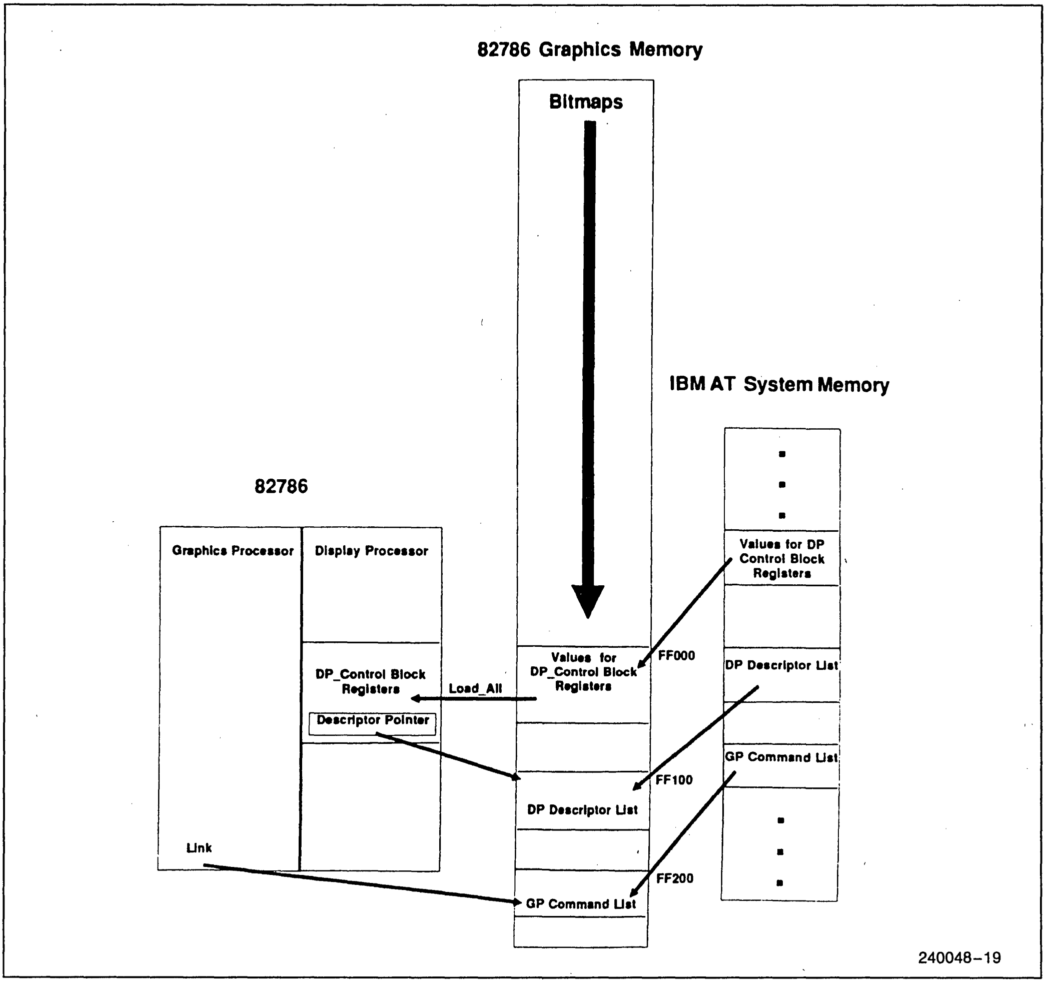


Figure 3.1

3.1.1 OVERVIEW OF PROGRAM

Figure 3.1 shows a graphical description of the Example program.

First, the correct values are written into the 82786 BIU Registers.

Section 3.2 explains how the values for these registers have been determined.

Next, Page 0 of graphics memory is cleared (used for bitmaps).

Next, values for the DP Control Block Registers, DP Descriptor List and GP Command List are copied from CPU memory space to Graphics memory space.

The values for the DP Control Block Registers are loaded from graphics memory into the 82786 registers by the DP LOAD_ALL command. Two of the DP Control Block Registers, the Descriptor Pointer Upper and Descriptor Pointer Lower, are combined to give a 22-bit address. This is the address of a valid DP Descriptor List located at location FF100 in graphics memory. The DP Descriptor List instructs the DP to fetch bitmap data starting at location 0 in graphics memory.

Now the Graphics Command list is executed by writing its address into the GP Parameter Registers and then writing a LINK command into the GP Opcode Register. The GP now draws the image into the graphics memory bitmap area.

3.2 A Detailed Description of the Example Program

3.2.0 TECHNICAL FACTS

This section provides technical facts used in the Example program.

Graphics Memory Addressing:

The graphics board uses 64 Kbytes of CPU address space from A0000 to AFFFF. The page selection register chooses one of 16 pages. This allows addressing of a total of 1 Megabyte of graphics memory.

The page selection register on the graphics board is set by outputting the page number (using the 80286 OUT instruction) to port location 0x00300. This technique will vary on other hardware systems using the 82786.

Bitmap Location:

The example program stores bitmaps in Graphics memory starting at location 0.

Graphics Command Buffer Location:

Starts at Graphics memory location FF200

Display Processor Descriptor List Location:

DP Descriptor list FF100 (base address in graphics memory)

82786 Register Access:

The 82786 Internal Register Block is accessed by memory access to CPU memory locations C4400 through C447F. The Graphics board decodes these addresses and issues an I/O access.

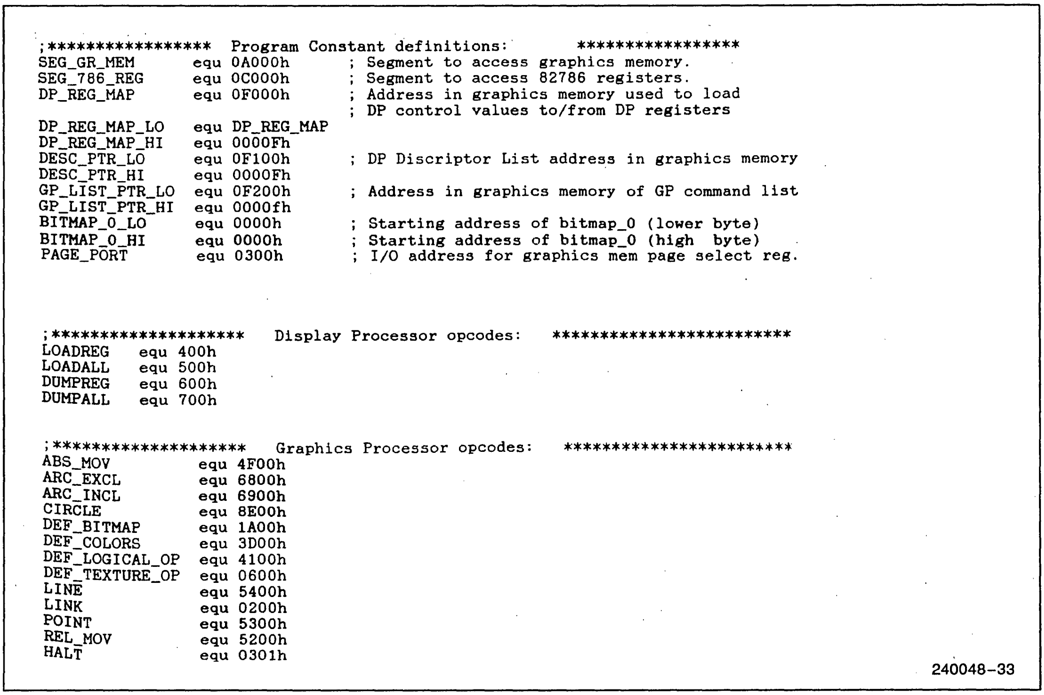
Video Timing Parameters:

The initialization values for the video timing parameters assume an 18 MHz VCLOCK.

Assembler:

The example programs were assembled with the Microsoft Macro Assembler Version 4.0

3.2.1 CONSTANT DEFINITIONS

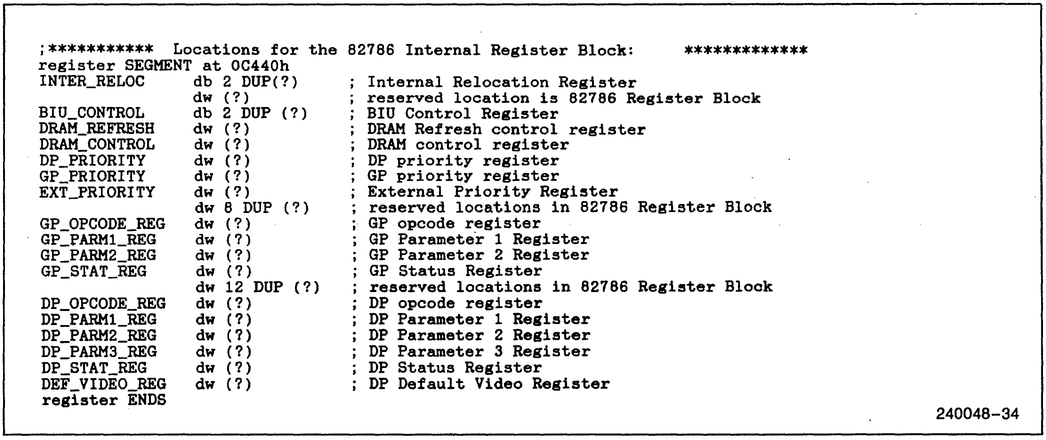


```
;*****************  Program Constant definitions:      *****************
SEG_GR_MEM        equ 0A000h       ; Segment to access graphics memory.
SEG_786_REG       equ 0C000h       ; Segment to access 82786 registers.
DP_REG_MAP        equ 0F000h       ; Address in graphics memory used to load
                                   ; DP control values to/from DP registers
DP_REG_MAP_LO     equ DP_REG_MAP
DP_REG_MAP_HI     equ 0000Fh
DESC_PTR_LO       equ 0F100h       ; DP Discriptor List address in graphics memory
DESC_PTR_HI       equ 0000Fh
GP_LIST_PTR_LO    equ 0F200h       ; Address in graphics memory of GP command list
GP_LIST_PTR_HI    equ 0000fh
BITMAP_0_LO       equ 0000h        ; Starting address of bitmap_0 (lower byte)
BITMAP_0_HI       equ 0000h        ; Starting address of bitmap_0 (high  byte)
PAGE_PORT         equ 0300h        ; I/O address for graphics mem page select reg.



;********************   Display Processor opcodes:   *************************
LOADREG    equ 400h
LOADALL    equ 500h
DUMPREG    equ 600h
DUMPALL    equ 700h

;********************   Graphics Processor opcodes:   ************************
ABS_MOV           equ 4F00h
ARC_EXCL          equ 6800h
ARC_INCL          equ 6900h
CIRCLE            equ 8E00h
DEF_BITMAP        equ 1A00h
DEF_COLORS        equ 3D00h
DEF_LOGICAL_OP    equ 4100h
DEF_TEXTURE_OP    equ 0600h
LINE              equ 5400h
LINK              equ 0200h
POINT             equ 5300h
REL_MOV           equ 5200h
HALT              equ 0301h
```

240048-33

3.2.2 LOCATIONS FOR THE 82786 INTERNAL REGISTER BLOCK

The REGISTER SEGMENT defines a template of locations for access to the 82786 Internal Register Block. The register segment is set to begin at memory location 0C440 (hex). As mentioned above, the Intel board issues an I/O access when the CPU accesses memory at addresses C4400–C447F.



```
;***********  Locations for the 82786 Internal Register Block:     *************
register SEGMENT at 0C440h
INTER_RELOC      db 2 DUP(?)      ; Internal Relocation Register
                 dw (?)           ; reserved location is 82786 Register Block
BIU_CONTROL      db 2 DUP (?)     ; BIU Control Register
DRAM_REFRESH     dw (?)           ; DRAM Refresh control register
DRAM_CONTROL     dw (?)           ; DRAM control register
DP_PRIORITY      dw (?)           ; DP priority register
GP_PRIORITY      dw (?)           ; GP priority register
EXT_PRIORITY     dw (?)           ; External Priority Register
                 dw 8 DUP (?)     ; reserved locations in 82786 Register Block
GP_OPCODE_REG    dw (?)           ; GP opcode register
GP_PARM1_REG     dw (?)           ; GP Parameter 1 Register
GP_PARM2_REG     dw (?)           ; GP Parameter 2 Register
GP_STAT_REG      dw (?)           ; GP Status Register
                 dw 12 DUP (?)    ; reserved locations in 82786 Register Block
DP_OPCODE_REG    dw (?)           ; DP opcode register
DP_PARM1_REG     dw (?)           ; DP Parameter 1 Register
DP_PARM2_REG     dw (?)           ; DP Parameter 2 Register
DP_PARM3_REG     dw (?)           ; DP Parameter 3 Register
DP_STAT_REG      dw (?)           ; DP Status Register
DEF_VIDEO_REG    dw (?)           ; DP Default Video Register
register ENDS
```

240048-34

3.2.3 VALUES FOR DP CONTROL BLOCK

The following program segment defines values for the Display Processor Control Block. Refer to the 82786 User's Manual for an explanation of each register. The comments in the program explain this setting used in our example.

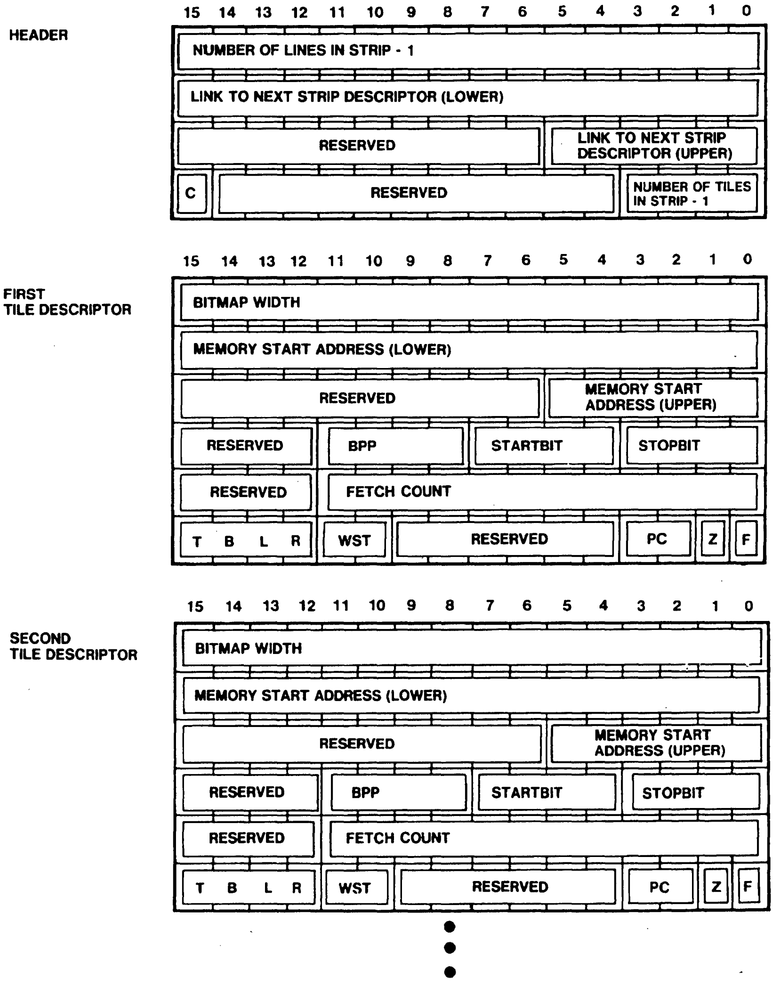
```
data SEGMENT

;*************   Values for the Display Processor Control Block:   *************
beg_dp_ctrl_blk LABEL word
               ; REGISTER NAME   :          SETTING
               ;---------------  ------------------------------------
    dw 3       ; Video Status    : cursor ON, and display ON
    dw 1111h   ; Interrupt Mask  : all interrupts disabled
    dw 00010h  ; Trip Point      : controls when DP fifo is loaded
    dw 00000h  ; Frame Interrupt : no interrupts on frame count
    dw 00000h  ; Reserved
    dw 00000h  ; CRT Mode        : non-interlaced, window status off,
               ;                 : DP master mode Blank master mode,
               ;                 : acceleration mode off

; The following 8 registers contain the video timing parameters for a screen
; resolution of 640 X 381 pixels.  These values assume  VCLOCK = 18MHz.
; These values achieve a screen refresh of 60 Hz.
    dw    86  ; Hsyncstp
    dw    95  ; Hfldstrt
    dw   735  ; Hfldstp
    dw   753  ; Linelength
    dw    11  ; Vsynstp
    dw    15  ; Vfldstrt
    dw   396  ; Vfldstp
    dw   398  ; Framelength

    dw DESC_PTR_LO ; DP descr ptr low
    dw DESC_PTR_HI ; DP descr ptr high
    dw 00000h ; Reserved
    dw 00101h ; Zoom factor     : X-zoom = 2, Y-zoom = 2
    dw 00006h ; Field color
    dw 00003h ; Border color
    dw 00000h ; 1 BPP pad
    dw 00000h ; 2 BPP pad
    dw 00000h ; 4 BPP pad
    dw 0A0FFh ; Cursor Style   : Size = 16 X 16, transparent, cursor pad
    dw 500  ; Cursor X-position
    dw 180  ; Cursor Y-position

  ; The following 16 registers define the cursor bit pattern (an upward arrow):
    dw 0000000100000000b
    dw 0000001110000000b
    dw 0000011111000000b
    dw 0000111111100000b
    dw 0001111111110000b
    dw 0011111111111000b
    dw 0111011111011100b
    dw 1100001110000110b
    dw 0000001110000000b
    dw 0000001110000000b
    dw 0000001110000000b
    dw 0000001110000000b
    dw 0000001110000000b
    dw 0000001110000000b
    dw 0000001110000000b
    dw 0000001110000000b
 end_DP_ctrl_blk LABEL word
```

240048-35

3.2.4 DISPLAY PROCESSOR DESCRIPTOR LIST

The following program segment defines a Display Processor Descriptor List. The DP reads the Descriptor List every frame, starting over at the beginning of the Descriptor List during vertical retrace. The Descriptor List determines the graphics memory addresses from which display data is fetched.

A Screen Descriptor List is composed of a header for each strip and a Tile Descriptor for each tile in a strip. (See Figure 3.2)



240048-20

Figure 3.2

3.2.4.1 Strip Header

The Strip Header defines the number of scan lines in the strip, the address of the next Strip Descriptor (link), and the number of tiles in the strip. The descriptor list in our example defines one strip, 381 lines long, composed of one tile, 80 bytes wise (640 pixels).

3.2.4.2 Tile Descriptor

The Tile Descriptor defines the Bitmap Width, Memory Start Address, BPP, StartBit, StopBit, Fetch Count, Border Control bits, Window Status (Window ID number), Zoom Control, and Field Tile Control.

The Bitmap Width gives the width of the source bitmap as defined by the GP DEF__BITMAP command when the bitmap was drawn.

NOTE:
The Bitmap Width value is not related to the tile width.

The Memory Start Address determines the beginning location in graphics memory where data is to be fetched for a given tile. This address is not necessarily the beginning address of the bitmap. If the Memory Start Address is higher than the beginning address of the bitmap, the tile will contain an image beginning at the corresponding location in the bitmap.

The width of a tile is determined by the **FETCH COUNT** value of the tile descriptor. The FETCH COUNT determines the amount of data to be fetched from graphics memory and displayed in a given tile. The value to be programmed into FETCH COUNT is the (Number of bytes − 2).

Fetch Count can be determined as follows:

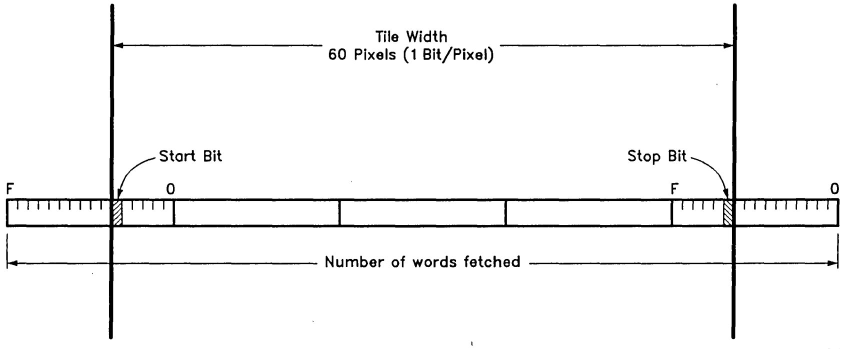
FETCH COUNT = (Desired tile width in pixels * bpp/8) − 2

The bitmap in our example is 1 bpp and the desired tile width is 640 pixels; therefore:

FETCH COUNT = (640 * 1/8) − 2 = 78.

BPP is a 4-bit field containing the bpp of the source bitmap as defined by the DEF__BITMAP when the bitmap was drawn.

The **STARTBIT** and **STOPBIT** fields are both 4-bits wide. Although FETCH COUNT is specified as a number of bytes, STARTBIT and STOPBIT are specified as a bit location within a word (0–F). These fields give pixel resolution to the beginning and ending of a tile. In our example, the STARTBIT is F and the STOPBIT is 0. It is the responsibility of the programmer to ensure that the STARTBIT and STOPBIT settings result in a valid number of bits for the given bitmap depth (bpp). For example, when the bitmap is 4 bpp, the total number of bits fetched must be a multiple of 4. See Figure 3.3.



240048-21

FETCH COUNT = (numbers of words fetched * 2) − 2
= (number of bytes fetched) − 2

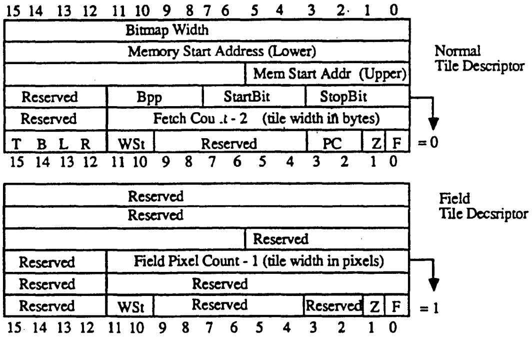
Figure 3.3. STARTBIT and STOPBIT

Valid STARTBIT and STOPBIT Values

Bits/Pixel	Valid STARTBIT	Valid STOPBIT
1	F,E,D,C,B,A,9,8,7,6,5,4,3,2,1,0	F,E,D,C,B,A,9,8,7,6,5,4,3,2,1,0
2	F,D,B,9,7,5,3,1	E,C,A,8,6,4,2,0
4	F,B,7,3	C,8,4,0
8	F,7	8,0

Field Tiles

When the field bit (bit zero of the last word) in a tile descriptor is set to one, the tile is filled with the color programmed in the FIELD COLOR register. When the field bit is set, the STARTBIT, STOPBIT, and BPP parameters become one 12-bit parameter that specifies the tile width in pixels. All other bits except WINDOW STATUS and Zoom should be programmed to zero. Although field tiles are not used in our examples, they are useful for filling a tile with a solid color. See Figure 3.4.

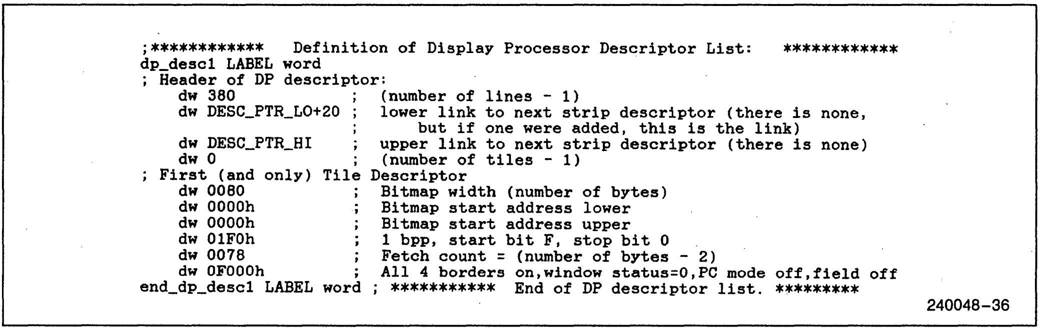


240048-22

NOTE:
Reserved fields must be programmed to zero for future compatibility.

Figure 3.4

Refer to the Intel 82786 User's Manual for more information on the Display Processor Descriptor List.



```
;************   Definition of Display Processor Descriptor List:    ************
dp_desc1 LABEL word
; Header of DP descriptor:
   dw 380          ;  (number of lines - 1)
   dw DESC_PTR_LO+20 ;  lower link to next strip descriptor (there is none,
                   ;      but if one were added, this is the link)
   dw DESC_PTR_HI  ;  upper link to next strip descriptor (there is none)
   dw 0            ;  (number of tiles - 1)
; First (and only) Tile Descriptor
   dw 0080         ;  Bitmap width (number of bytes)
   dw 0000h        ;  Bitmap start address lower
   dw 0000h        ;  Bitmap start address upper
   dw 01F0h        ;  1 bpp, start bit F, stop bit 0
   dw 0078         ;  Fetch count = (number of bytes - 2)
   dw 0F000h       ;  All 4 borders on,window status=0,PC mode off,field off
end_dp_desc1 LABEL word ; ***********  End of DP descriptor list. *********
```

240048-36

3.2.5 GRAPHICS PROCESSOR COMMAND LIST

The following program segment defines a Graphics Processor Command List.

A GP command list consists of a series of GP opcodes and parameters. The Graphics Processor reads and executes the command list until a halt instruction is encountered.

The first command (DEF__BITMAP) sets the beginning address in graphics memory of the bitmap to be modified. This command also sets the bitmap dimensions and the number of bits per pixel (bpp) of the bitmap. All subsequent drawing commands will affect this bitmap until a new DEF__BITMAP command is issued. It is the resonsibility of the programmer to ensure the BPP in the tile descriptor is the same as the BPP used by the GP when drawing the picture.

Bitmaps must begin at a word (even byte) address. Also, a bitmap must be an integral number of words wide. The value for xmax must satisfy the following equation:

$$[(xmax + 1) * bpp] \text{ MOD } 16 = 0$$

Next, the DEF__TEXTURE, DEF__COLORS, and DEF__LOGICAL__OP commands are issued. These settings stay in effect for all subsequent drawing commands. They can be reset whenever necessary.

Next, an ABS__MOVE command is issued to move the Graphic Current Position Pointer (GCPP) to the beginning location of the drawing. The remainder of the GP Command List in our Example is composed of REL__MOV, LINE, and ARC__INCL commands.

Figure 3.5 illustrates the result of the command list in the example program.

The HALT command at the end of the GP command list is very important. The GP continues execution until it encounters a HALT instruction (a NOP with the ECL bit set). If the HALT instruction is not present, the GP will continue fetching and trying to execute instructions until it reaches a "command" with the low bit set.

Several different GP command lists may be kept in graphics memory at the same time. Each command list may be executed by writing the appropriate address into the GP parameter registers and then writing a LINK command into the GP Opcode Register.

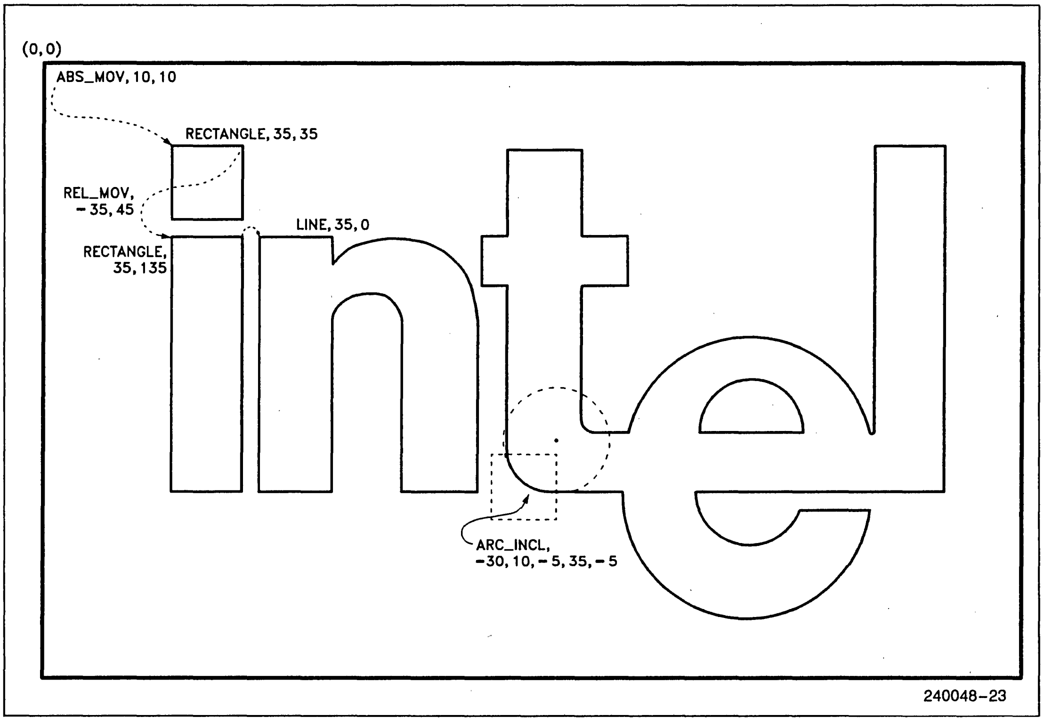
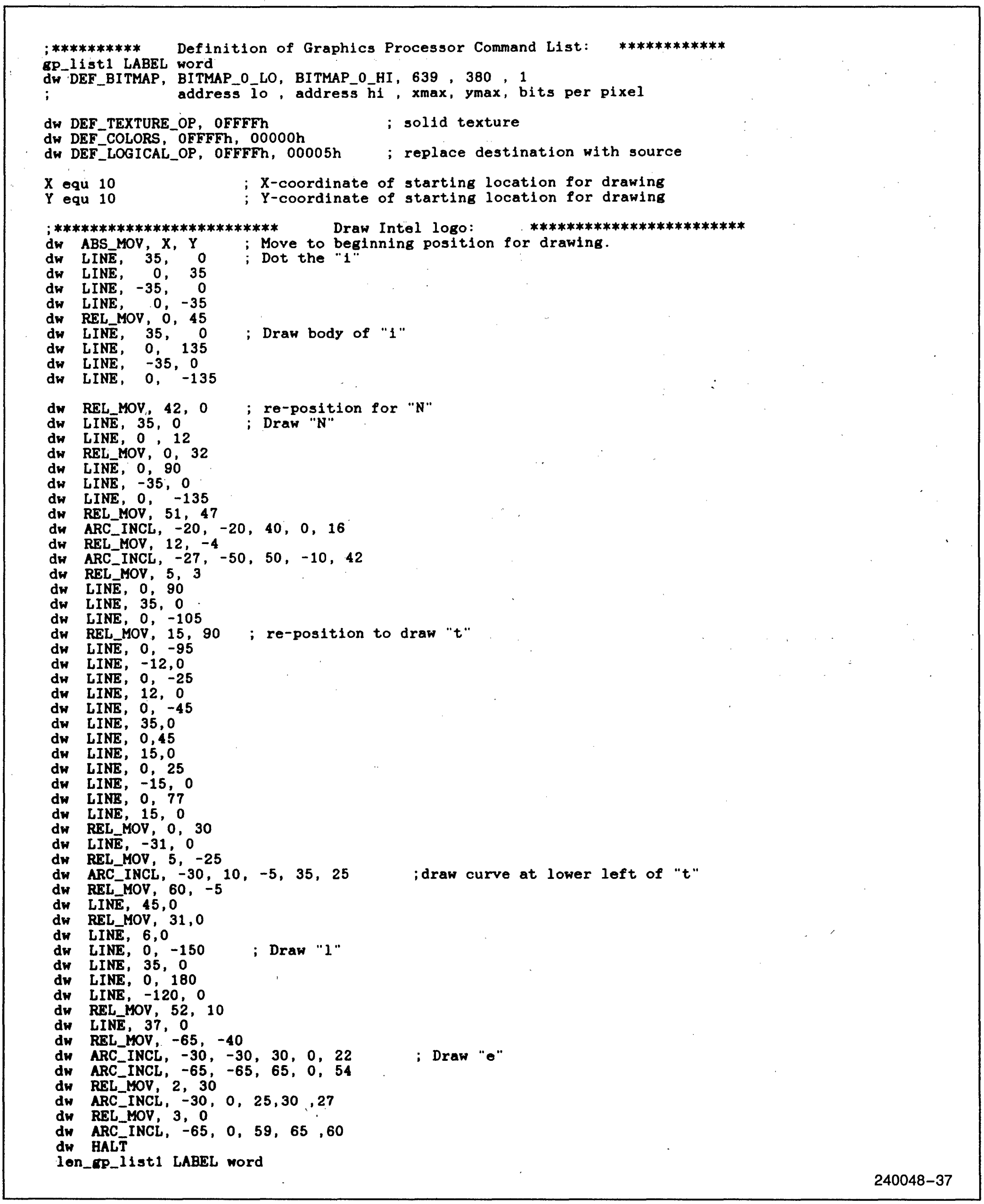

(0,0)
ABS_MOV, 10, 10
RECTANGLE, 35, 35
REL_MOV, −35, 45
RECTANGLE, 35, 135
LINE, 35, 0
ARC_INCL, −30, 10, −5, 35, −5
240048–23

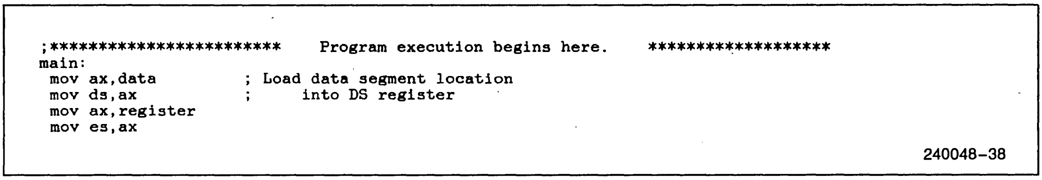
Figure 3.5



```
;**********    Definition of Graphics Processor Command List:   ************
gp_list1 LABEL word
dw DEF_BITMAP, BITMAP_0_LO, BITMAP_0_HI, 639 , 380 , 1
;              address lo , address hi , xmax, ymax, bits per pixel

dw DEF_TEXTURE_OP, 0FFFFh            ; solid texture
dw DEF_COLORS, 0FFFFh, 00000h
dw DEF_LOGICAL_OP, 0FFFFh, 00005h    ; replace destination with source

X equ 10              ; X-coordinate of starting location for drawing
Y equ 10              ; Y-coordinate of starting location for drawing

;**************************        Draw Intel logo:      ************************
dw  ABS_MOV, X, Y     ; Move to beginning position for drawing.
dw  LINE,  35,   0    ; Dot the "i"
dw  LINE,   0,  35
dw  LINE, -35,   0
dw  LINE,   0, -35
dw  REL_MOV, 0, 45
dw  LINE,  35,   0    ; Draw body of "i"
dw  LINE,  0,  135
dw  LINE,  -35, 0
dw  LINE,  0,  -135

dw  REL_MOV, 42, 0    ; re-position for "N"
dw  LINE, 35, 0       ; Draw "N"
dw  LINE, 0 , 12
dw  REL_MOV, 0, 32
dw  LINE, 0, 90
dw  LINE, -35, 0
dw  LINE, 0,  -135
dw  REL_MOV, 51, 47
dw  ARC_INCL, -20, -20, 40, 0, 16
dw  REL_MOV, 12, -4
dw  ARC_INCL, -27, -50, 50, -10, 42
dw  REL_MOV, 5, 3
dw  LINE, 0, 90
dw  LINE, 35, 0
dw  LINE, 0, -105
dw  REL_MOV, 15, 90   ; re-position to draw "t"
dw  LINE, 0, -95
dw  LINE, -12,0
dw  LINE, 0, -25
dw  LINE, 12, 0
dw  LINE, 0, -45
dw  LINE, 35,0
dw  LINE, 0,45
dw  LINE, 15,0
dw  LINE, 0, 25
dw  LINE, -15, 0
dw  LINE, 0, 77
dw  LINE, 15, 0
dw  REL_MOV, 0, 30
dw  LINE, -31, 0
dw  REL_MOV, 5, -25
dw  ARC_INCL, -30, 10, -5, 35, 25       ;draw curve at lower left of "t"
dw  REL_MOV, 60, -5
dw  LINE, 45,0
dw  REL_MOV, 31,0
dw  LINE, 6,0
dw  LINE, 0, -150     ; Draw "l"
dw  LINE, 35, 0
dw  LINE, 0, 180
dw  LINE, -120, 0
dw  REL_MOV, 52, 10
dw  LINE, 37, 0
dw  REL_MOV, -65, -40
dw  ARC_INCL, -30, -30, 30, 0, 22       ; Draw "e"
dw  ARC_INCL, -65, -65, 65, 0, 54
dw  REL_MOV, 2, 30
dw  ARC_INCL, -30, 0, 25,30 ,27
dw  REL_MOV, 3, 0
dw  ARC_INCL, -65, 0, 59, 65 ,60
dw  HALT
len_gp_list1 LABEL word
```

240048-37

3.2.6 PROGRAM CODE SEGMENT HEADER

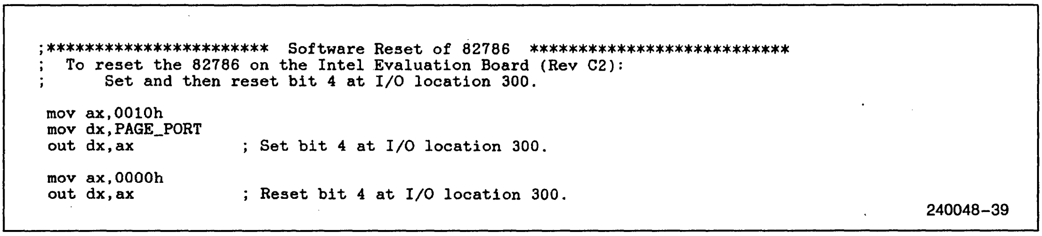


```
;************************    Program execution begins here.    *******************
main:
 mov ax,data            ; Load data segment location
 mov ds,ax              ;     into DS register
 mov ax,register
 mov es,ax
```

240048-38

This section of code provides a standard Assembly Language program header. This code loads the DS (Data Segment Register) and the ES (Extra Segment Register). The ES register is used to access the 82786 Internal Registers.

3.2.7 SOFTWARE RESET



```
;************************  Software Reset of 82786  ****************************
;  To reset the 82786 on the Intel Evaluation Board (Rev C2):
;      Set and then reset bit 4 at I/O location 300.

 mov ax,0010h
 mov dx,PAGE_PORT
 out dx,ax              ; Set bit 4 at I/O location 300.

 mov ax,0000h
 out dx,ax              ; Reset bit 4 at I/O location 300.
```

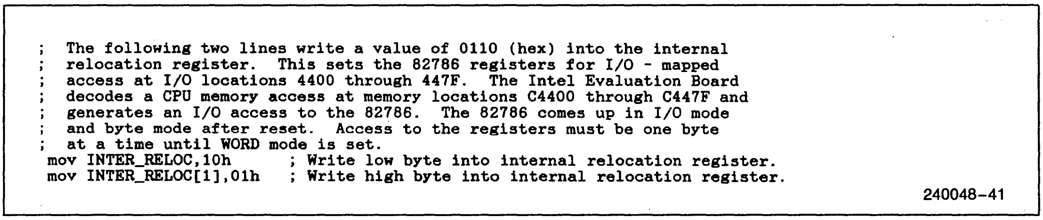
240048-39

This section of code performs a reset of the 82786 by setting and then resetting bit 4 of the CPU I/O port 300 (hex). The EVB then issues a reset signal to the 82786 RESET pin.

3.2.8. BIU INITIALIZATION

The following sections of code initialize the 82786 Bus Interface Unit (BIU). BIU initialization is accomplished by writing the correct values into each of the BIU registers. A brief description of each register follows.

3.2.8.1 Internal Relocation Register



```
;  The following two lines write a value of 0110 (hex) into the internal
;  relocation register.  This sets the 82786 registers for I/O - mapped
;  access at I/O locations 4400 through 447F.  The Intel Evaluation Board
;  decodes a CPU memory access at memory locations C4400 through C447F and
;  generates an I/O access to the 82786.  The 82786 comes up in I/O mode
;  and byte mode after reset.  Access to the registers must be one byte
;  at a time until WORD mode is set.
 mov INTER_RELOC,10h       ; Write low byte into internal relocation register.
 mov INTER_RELOC[1],01h    ; Write high byte into internal relocation register.
```

240048-41

The INTERNAL RELOCATION register is set first. The 82786 comes up in byte mode after RESET; therefore, this register is set by writing one byte at a time.

The desired base address for the 82786 registers is 004400 (hex). The base address must always be located on a 128 word boundary. (The registers are accessed at locations 004400 through 00447F.) An 82786 address is 22 bits long. The upper 15 bits of the desired base address is written into the upper 15 bits of the Internal Relocation Register.

We want to set the chip for I/O mode, therefore, a zero is written into the M/IO bit. Therefore, we write a value of 0110 (hex) into this register. See Figure 3.6.

BIU
Internal Relocation Register

00 0000 0100 0100 0 + $\overline{\underline{000\ 0000}}$

0 0 4 4 0

Desired Location of 82786 Registers = 4400–447F

Upper 15 bits of Base Address

←—————————————→				M/IO
0000	0001	0001	0000	0

Value written in register = 0110 Hex

Figure 3.6

3.2.8.2 BIU Control Register

```
;  The following two lines write a value of 0011 (hex) into the BIU control
;  register.  This sets the Internal Register Block for 16-bit WORD access
;  by the External CPU.  All subsequent access to the 82786 registers is by
;  WORD access.
mov BIU_CONTROL,10h       ; Write low byte into BIU control register
mov BIU_CONTROL[1],00h    ; Write high byte into BIU control register
```

240048-42

These two lines set the BIU Control Register. Because the 82786 is in byte mode after RESET, this register is written one byte at a time. After setting this register for word mode, all subsequent register access is by word mode.

BIU Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Unused									VR	WT	BCP	GI	DI	WP 1	WP 2

Figure 3.7

The BIU Control Register has seven one-bit fields as shown in Figure 3.7. The settings for our Example program follow:

VR = 0
Set for conventional DRAM memory cycles (not VRAM).

WT = 0
Number of wait states in synchronous 80186 interface. The synchronous 80186 interface is not used in our example; therefore, this is a "don't care" setting.

BCP = 1
This sets the External CPU for 16-bit word access.

GI and DI
When the 82786 issues an interrupt, these two bits can be read to determine which processor has issued the interrupt, then either the DP or GP Status Register can be read to determine the cause of the interrupt.

WP1 and WP2
The write protect bits are not set in our Example program.

3.2.8.3 DRAM Refresh Control Register

```
mov DRAM_REFRESH,0018h    ; Write value into DRAM refresh control register.
```

240048-43

This register is programmed with a 6-bit Refresh Scalar for controlling the frequency for DRAM refresh cycles.

The value programmed in this register depends on the refresh requirements of the DRAMs, the clock speed, and the number of DRAM row addresses. The value for the Refresh Scalar can be calculated by the following formula:

$$\frac{T_{ref} \times CLK}{16 \times \text{Refresh Rows}} - 1$$

Where:

T_{ref} = Refresh Time interval
CLK = 82786 System Clock speed
Refresh rows = Number of DRAM rows requiring refresh

In our example, we have:

$$\frac{4 \text{ ms} \times 20 \text{ MHz}}{16 \times 256} - 1 = 18.53$$

NOTE:
DRAM refresh cycles can be turned off by programming a value of 3F (hex) into the DRAM Refresh Register.

3.2.8.4 DRAM Control Register

```
mov DRAM_CONTROL, 001Dh  ; Write value into DRAM control register.
```

240048-44

Figure 3.8. DRAM/VRAM Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1
Unused									RW1	RW2	DC1	DC0	HT2	HT1	HT0

The DRAM Control Register has seven one-bit fields as shown in Figure 3.7. The settings for our Example program follow:

RW1 and RW0 indicate the number of rows of graphics memory.
RW1 = 0, RW0 = 0 indicates one row of graphics memory.
DC1 and DC0 indicate DRAM/VRAM configuration.
DC1 = 1, DC0 = 1 indicate Fast Page Mode, Interleaved.
HT2, HT1, and HT0 indicate the DRAM/VRAM Height of graphics memory.
HT2 = 1, HT1 = 0, HT0 = 1 indicates 256K x N-type DRAMs.

3.2.8.5 Display Processor, Graphics Processor and External Priority Registers

```
mov DP_PRIORITY, 003Fh     ; Write value into DP priority register.

mov GP_PRIORITY,0009h      ; Write value into GP Priority register

mov EXT_PRIORITY,0028h     ; Write value into External Priority register.
```

240048-45

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DP Priority	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
	Reserved										First Priority			Second Priority		
GP Priority	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
	Reserved										First Priority			Second Priority		
External	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
	Reserved										First Priority			Reserved		

Bus access priorities are programmable for the GP, DP and External Processor. Note that DRAM refresh is not programmable and always has highest priority. The First Priority Level (FPL) is used to obtain bus access; Secondary Priority Level (SPL) is used to keep the bus when another processor makes a request. The highest priority is 111 (binary). The lowest priority is 000. Refer to the "82786 User's Manual" Section 4.3—Bus Cycle Arbitration.

In our Example program, DP has highest priority, External CPU has second priority, and GP has lowest priority.

NOTE:

These priorities may be changed at any time during program execution.

3.2.9 CLEAR PAGE 0 OF GRAPHICS MEMORY

```
;****************  Clear Page 0 of Graphics memory (64K bytes):     ***************
 mov ax,SEG_GR_MEM            ; Graphics memory space is in the 'A' segment
 mov ds,ax

 mov ax,0
 mov dx,PAGE_PORT
 out dx,ax                    ; Select page 0 of graphics memory

 mov bx,0
 mov cx,32767                 ; 32767 words of memory to be cleared = 64K bytes
 mov si,0
 CLEAR_MEMORY:                ; Clear page 0 of graphics memory (to be
      mov [si],bx             ; used as a bitmap for drawing commands.)
      add si,2
      loop CLEAR_MEMORY
```

240048-46

Page zero of graphics memory is used for storing the bitmap. Before drawing into the bitmap, it must be cleared (filled with zeroes).

This section of code clears page 0 of graphics memory by writing zeros into each memory location. First, the segment address of Graphics Memory space is written into the CPU DS register. Next, page zero of graphics memory is selected by writing a zero into the Page Select Register on the Evaluation Board. The loop command is used to clear 32767 words (64 Kbytes) of memory.

The GP bit__blit command using logical operator 0, or the scan__lines command using color 0 may also be used as a fast technique for clearing a section of graphics memory.

3.2.10 PREPARE DS, ES, AND DIR FLAG FOR USE WITH REP MOVSB INSTRUCTION

```
;****** Prepare DS, ES, and Dir Flag for use with REP MOVSB instruction. ******
  mov ax,0Fh
  mov dx,PAGE_PORT
  out dx,ax                        ; Select page F of graphics memory

  mov ax,SEG  beg_dp_ctrl_blk
  mov ds,ax                        ; Set data segment
  mov ax,SEG_GR_MEM
  mov es,ax                        ;     and extra segment.
  cld                              ; Clear Direction Flag, sets auto-increment
                                   ;  of SI and DI when using REP instruction.
```

240048-47

This section of code performs the necessary preparation for the next three sections: moving the DP Control Block, DP Descriptor List, and GP command list from CPU memory to Graphics Memory.

Page F of graphics memory is the desired destination of these three blocks of data, therefore page F of graphics memory is selected by writing to the PAGE_PORT. Next, the Data Segment and Extra Segment registers are written. Lastly, the Direction Flag is cleared. This is necessary to cause the string instruction to auto-increment the SI and DI index registers.

3.2.11 COPY DP CONTROL BLOCK REGISTERS FROM CPU MEMORY TO GRAPHICS MEMORY

```
;***** Copy DP CONTROL BLOCK REGISTERS from CPU memory to Graphics Memory. ****
  lea cx, end_DP_ctrl_blk
  sub cx, offset beg_dp_ctrl_blk
  lea si, beg_dp_ctrl_blk
  mov di, offset DP_REG_MAP
  rep movsb                        ; Move CX bytes from DS:[SI] to ES:[DI]
                                   ; thus, copying DP Control Block Registers
                                   ; from CPU memory to Graphics memory.
```

240048-48

This section of code copies the values for the DP Control Block registers from CPU memory to Graphics Memory beginning at address FF000 (hex).

3.2.12 COPY DP DESCRIPTOR LIST FROM CPU MEMORY TO GRAPHICS MEMORY

```
;******* Copy DP Descriptor List from CPU memory to Graphics memory. ********
  lea cx, end_dp_descl
  sub cx, offset dp_descl
  lea si, dp_descl
  mov di, offset DESC_PTR_LO
  rep movsb                        ; Move CX bytes from DS:[SI] to ES:[DI]
                                   ; thus copying DP descriptor list from CPU
                                   ; memory to graphics memory.
```

240048-49

This section of code copies the values for the DP Descriptor List from CPU memory to Graphics Memory beginning at address FF100 (hex).

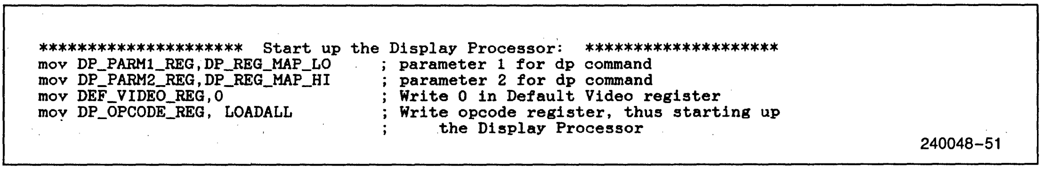
3.2.13 COPY GP COMMAND LIST FROM CPU MEMORY TO GRAPHICS MEMORY

```
;*********  Copy GP command list from CPU memory to graphics memory:  *********
  lea cx, len_gp_list1
  sub cx, offset gp_list1
  lea si, gp_list1
  mov di, offset GP_LIST_PTR_LO
  rep movsb                        ; Move CX bytes from DS:[SI] to ES:[DI]
                                   ; thus copying GP command list from CPU
                                   ; memory to graphics memory.
```

240048-50

This section of code copies the GP command list from CPU memory to Graphics Memory beginning at address FF200 (hex). The labels in the program marking the beginning (gp__list1) and ending (len__gp__list1) of the GP command list provide a convenient method for determining the length of the GP command list. Commands may be added or deleted from the command list, the program computes the number of bytes to be copied into graphics memory.

3.2.14 START THE DISPLAY PROCESSOR



```
*********************  Start up the Display Processor:  ********************
mov DP_PARM1_REG,DP_REG_MAP_LO      ; parameter 1 for dp command
mov DP_PARM2_REG,DP_REG_MAP_HI      ; parameter 2 for dp command
mov DEF_VIDEO_REG,0                 ; Write 0 in Default Video register
mov DP_OPCODE_REG, LOADALL          ; Write opcode register, thus starting up
                                    ;     the Display Processor
```

240048-51

This section of code starts up the Display Processor. First, the address of the values for the DP Control Block Registers are written into the DP Parameter registers. The lower part of the address is written into PARAMETER 1 Register; the upper part of the address is written into PARAMETER 2 Register. The Default Video Register is assigned zero. Lastly, the LOADALL opcode is written into the DP OPCODE register, thus starting operation of the DP by loading the values for the DP Control Block.

It is important to write the address for the LOADALL command into the Parameter registers before the LOADALL command is written into the opcode register. If the LOADALL command is written first, the registers will be loaded immediately, from an erroneous location.

Now, all the pointers and data structures for the Display Processor are in place. The Descriptor Pointer now points to a valid Descriptor List which points to a valid bitmap area in graphics memory. Refer to Figure 3.9.

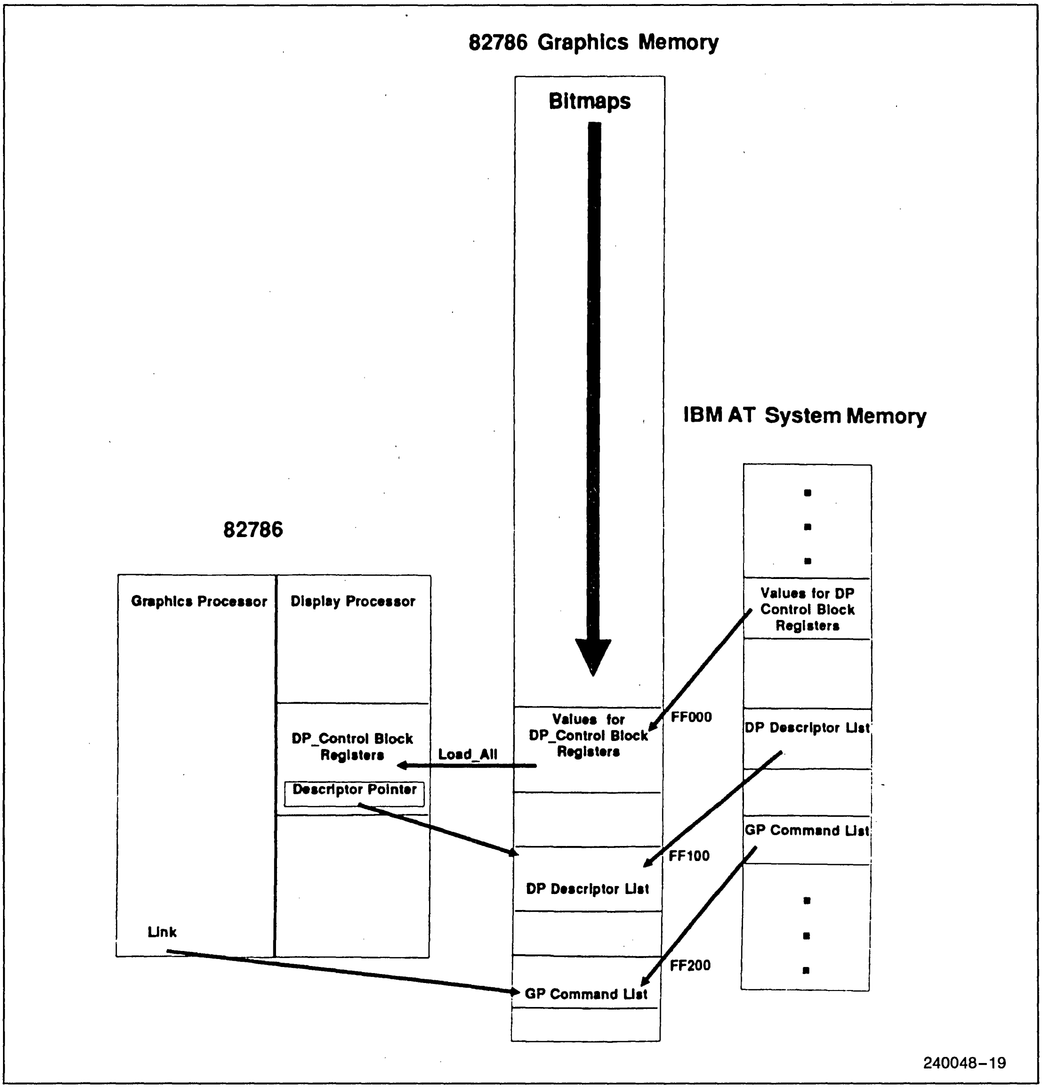


Figure 3.9

3.2.15 EXECUTE THE GRAPHICS PROCESSOR COMMAND LIST

```
********************** Execute the GP command list: *******************
mov GP_PARM1_REG,GP_LIST_PTR_LO    ; parameter 1 for GP command
mov GP_PARM2_REG,GP_LIST_PTR_HI    ; parameter 2 for GP command
mov GP_OPCODE_REG, LINK            ; Write opcode register, thus starting
                                   ;    execution of the GP command list.
```

240048-52

This section of code starts up the Graphics Processor. First the lower and upper address of the GP command list are written into the GP Parameter Registers 1 and 2, respectively. Next, the opcode for the GP LINK command is written into the GP opcode register. When a zero is written into the End of Command List (ECL) bit (lowest bit) the GP begins execution. The LINK command causes the GP execution to continue at the indicated address.

It is important to write the link address into the parameter registers before writing the LINK opcode into the opcode register. If the LINK command is written first, the GP will begin execution immediately, executing an erroneous command list. See Figure 3.10.

Execute GP Command List

- Write addresses of GP Command List into GP Parameter Registers

GP Parameter Registers

Lower Part of Address → Parameter Register 1
Upper Part of Address → Parameter Register 2

- Write GP opcode for Link command into GP Opcode Register

	Offset	
Opcode	20	Opcode
Parameter 1	22	Link Address Lower
Parameter 2	24	Link Address Upper
Status	26	

Figure 3.10

3.2.16 TERMINATE PROGRAM

```
;************************      Terminate program:     *************************
  mov ah,4Ch                           ; Call BIOS terminate function
  int 21h                              ;     to return to MS-DOS operating system.

 code ENDS
```

These two lines call the BIOS routine to return control to the operating system.

3.3 Example Source Code Listing

This section provides the complete source code listing of the EXAMPLE program.

```
;******************************************************************************
; Program name:  EXAMPLE1.ASM
;
; Description:   Initialize the 82786 registers, program the Display
;                Processor (DP) for one full-screen window, and draw a
;                simple graphics image using the Graphics Processor (GP).
;
; Direct questions to your nearest Intel Sales Office.
;
;******************************************************************************
```

```
;******************  Program Constant definitions:       *****************
SEG_GR_MEM        equ 0A000h       ; Segment to access graphics memory.
SEG_786_REG       equ 0C000h       ; Segment to access 82786 registers.
DP_REG_MAP        equ 0F000h       ; Address in graphics memory used to load
                                   ; DP control values to/from DP registers
DP_REG_MAP_LO     equ DP_REG_MAP
DP_REG_MAP_HI     equ 0000Fh
DESC_PTR_LO       equ 0F100h       ; DP Discriptor List address in graphics memory
DESC_PTR_HI       equ 0000Fh
GP_LIST_PTR_LO    equ 0F200h       ; Address in graphics memory of GP command list
GP_LIST_PTR_HI    equ 0000fh
BITMAP_0_LO       equ 0000h        ; Starting address of bitmap_0 (lower byte)
BITMAP_0_HI       equ 0000h        ; Starting address of bitmap_0 (high  byte)
PAGE_PORT         equ 0300h        ; I/O address for graphics mem page select reg.



;*********************    Display Processor opcodes:    *************************
LOADREG    equ 400h
LOADALL    equ 500h
DUMPREG    equ 600h
DUMPALL    equ 700h

;********************    Graphics Processor opcodes:    ************************
ABS_MOV           equ 4F00h
ARC_EXCL          equ 6800h
ARC_INCL          equ 6900h
CIRCLE            equ 8E00h
DEF_BITMAP        equ 1A00h
DEF_COLORS        equ 3D00h
DEF_LOGICAL_OP    equ 4100h
DEF_TEXTURE_OP    equ 0600h
LINE              equ 5400h
LINK              equ 0200h
POINT             equ 5300h
REL_MOV           equ 5200h
HALT              equ 0301h
```

```
;***********  Locations for the 82786 Internal Register Block:      *************
register SEGMENT at 0C440h
INTER_RELOC       db 2 DUP(?)      ; Internal Relocation Register
                  dw (?)           ; reserved location is 82786 Register Block
BIU_CONTROL       db 2 DUP (?)     ; BIU Control Register
DRAM_REFRESH      dw (?)           ; DRAM Refresh control register
DRAM_CONTROL      dw (?)           ; DRAM control register
DP_PRIORITY       dw (?)           ; DP priority register
GP_PRIORITY       dw (?)           ; GP priority register
EXT_PRIORITY      dw (?)           ; External Priority Register
                  dw 8 DUP (?)     ; reserved locations in 82786 Register Block
GP_OPCODE_REG     dw (?)           ; GP opcode register
GP_PARM1_REG      dw (?)           ; GP Parameter 1 Register
GP_PARM2_REG      dw (?)           ; GP Parameter 2 Register
GP_STAT_REG       dw (?)           ; GP Status Register
                  dw 12 DUP (?)    ; reserved locations in 82786 Register Block
DP_OPCODE_REG     dw (?)           ; DP opcode register
DP_PARM1_REG      dw (?)           ; DP Parameter 1 Register
DP_PARM2_REG      dw (?)           ; DP Parameter 2 Register
DP_PARM3_REG      dw (?)           ; DP Parameter 3 Register
DP_STAT_REG       dw (?)           ; DP Status Register
DEF_VIDEO_REG     dw (?)           ; DP Default Video Register
register ENDS
```

```
data SEGMENT

;*************   Values for the Display Processor Control Block:   *************
beg_dp_ctrl_blk LABEL word
               ; REGISTER NAME   :          SETTING
               ;---------------   -------------------------------------
    dw 3       ; Video Status    : cursor ON, and display ON
    dw 1111h   ; Interrupt Mask  : all interrupts disabled
    dw 00010h  ; Trip Point      : controls when DP fifo is loaded
    dw 00000h  ; Frame Interrupt : no interrupts on frame count
    dw 00000h  ; Reserved
    dw 00000h  ; CRT Mode        : non-interlaced, window status off,
               ;                 : DP master mode Blank master mode,
               ;                 : acceleration mode off

; The following 8 registers contain the video timing parameters for a screen
; resolution of 640 X 381 pixels.  These values assume  VCLOCK = 18MHz.
; These values achieve a screen refresh of 60 Hz.
    dw    86  ; Hsyncstp
    dw    95  ; Hfldstrt
    dw   735  ; Hfldstp
    dw   753  ; Linelength
    dw    11  ; Vsynstp
    dw    15  ; Vfldstrt
    dw   396  ; Vfldstp
    dw   398  ; Framelength

    dw DESC_PTR_LO ; DP descr ptr low
    dw DESC_PTR_HI ; DP descr ptr high
    dw 00000h ; Reserved
    dw 00101h ; Zoom factor     : X-zoom = 2, Y-zoom = 2
    dw 00006h ; Field color
    dw 00003h ; Border color
    dw 00000h ; 1 BPP pad
    dw 00000h ; 2 BPP pad
    dw 00000h ; 4 BPP pad
    dw 0A0FFh ; Cursor Style   : Size = 16 X 16, transparent, cursor pad
    dw 500   ; Cursor X-position
    dw 180   ; Cursor Y-position

    The following 16 registers define the cursor bit pattern (an upward arrow):
    dw 0000000100000000b
    dw 0000001110000000b
    dw 0000011111000000b
    dw 0000111111100000b
    dw 0001111111110000b
    dw 0011111111111000b
    dw 0111011111011100b
    dw 1100001110000110b
    dw 0000001110000000b
    dw 0000001110000000b
    dw 0000001110000000b
    dw 0000001110000000b
    dw 0000001110000000b
    dw 0000001110000000b
    dw 0000001110000000b
    dw 0000001110000000b
 end_DP_ctrl_blk LABEL word
```

```
;************   Definition of Display Processor Descriptor List:   ************
dp_desc1 LABEL word
; Header of DP descriptor:
    dw 380           ;  (number of lines - 1)
    dw DESC_PTR_LO+20 ;  lower link to next strip descriptor (there is none,
                     ;       but if one were added, this is the link)
    dw DESC_PTR_HI   ;  upper link to next strip descriptor (there is none)
    dw 0             ;  (number of tiles - 1)
; First (and only) Tile Descriptor
    dw 0080          ;  Bitmap width (number of bytes)
    dw 0000h         ;  Bitmap start address lower
    dw 0000h         ;  Bitmap start address upper
    dw 01F0h         ;  1 bpp, start bit F, stop bit 0
    dw 0078          ;  Fetch count = (number of bytes - 2)
    dw 0F000h        ;  All 4 borders on,window status=0,PC mode off,field off
end_dp_desc1 LABEL word ; ***********  End of DP descriptor list. *********
```

```
;**********    Definition of Graphics Processor Command List:    ************
gp_list1 LABEL word
dw DEF_BITMAP, BITMAP_0_LO, BITMAP_0_HI, 639 , 380 , 1
;              address lo , address hi , xmax, ymax, bits per pixel

dw DEF_TEXTURE_OP, OFFFFh            ; solid texture
dw DEF_COLORS, OFFFFh, 00000h
dw DEF_LOGICAL_OP, OFFFFh, 00005h    ; replace destination with source

X equ 10               ; X-coordinate of starting location for drawing
Y equ 10               ; Y-coordinate of starting location for drawing

;*************************          Draw Intel logo:      ************************
dw  ABS_MOV, X, Y      ; Move to beginning position for drawing.
dw  LINE,  35,   0     ; Dot the "i"
dw  LINE,   0,  35
dw  LINE, -35,   0
dw  LINE,   0, -35
dw  REL_MOV, 0, 45
dw  LINE,  35,   0     ; Draw body of "i"
dw  LINE,  0,  135
dw  LINE,  -35, 0
dw  LINE,  0,  -135

dw  REL_MOV, 42, 0     ; re-position for "N"
dw  LINE, 35, 0        ; Draw "N"
dw  LINE, 0 , 12
dw  REL_MOV, 0, 32
dw  LINE, 0, 90
dw  LINE, -35, 0
dw  LINE, 0,  -135
dw  REL_MOV, 51, 47
dw  ARC_INCL, -20, -20, 40, 0, 16
dw  REL_MOV, 12, -4
dw  ARC_INCL, -27, -50, 50, -10, 42
dw  REL_MOV, 5, 3
dw  LINE, 0, 90
dw  LINE, 35, 0
dw  LINE, 0, -105
dw  REL_MOV, 15, 90    ; re-position to draw "t"
dw  LINE, 0, -95
dw  LINE, -12,0
dw  LINE, 0, -25
dw  LINE, 12, 0
dw  LINE, 0, -45
dw  LINE, 35,0
dw  LINE, 0,45
dw  LINE, 15,0
dw  LINE, 0, 25
dw  LINE, -15, 0
dw  LINE, 0, 77
dw  LINE, 15, 0
dw  REL_MOV, 0, 30
dw  LINE, -31, 0
dw  REL_MOV, 5, -25
dw  ARC_INCL, -30, 10, -5, 35, 25       ;draw curve at lower left of "t"
dw  REL_MOV, 60, -5
dw  LINE, 45,0
dw  REL_MOV, 31,0
dw  LINE, 6,0
dw  LINE, 0, -150      ; Draw "l"
dw  LINE, 35, 0
dw  LINE, 0, 180
dw  LINE, -120, 0
dw  REL_MOV, 52, 10
dw  LINE, 37, 0
dw  REL_MOV, -65, -40
dw  ARC_INCL, -30, -30, 30, 0, 22       ; Draw "e"
dw  ARC_INCL, -65, -65, 65, 0, 54
dw  REL_MOV, 2, 30
dw  ARC_INCL, -30, 0, 25,30 ,27
dw  REL_MOV, 3, 0
dw  ARC_INCL, -65, 0, 59, 65 ,60
dw  HALT
len_gp_list1 LABEL word

data ENDS
```

```
code SEGMENT
ASSUME cs:code,ds:data,es:register


;*************************    Program execution begins here.    ******************
main:
 mov ax,data          ; Load data segment location
 mov ds,ax            ;     into DS register
 mov ax,register
 mov es,ax


;************************ Software Reset of 82786 ***************************
;  To reset the 82786 on the Intel Evaluation Board (Rev C2):
;      Set and then reset bit 4 at I/O location 300.

 mov ax,0010h
 mov dx,PAGE_PORT
 out dx,ax             ; Set bit 4 at I/O location 300.

 mov ax,0000h
 out dx,ax             ; Reset bit 4 at I/O location 300.


;************************    BIU initialization:    ***************************


;  The following two lines write a value of 0110 (hex) into the internal
;  relocation register.  This sets the 82786 registers for I/O - mapped
;  access at I/O locations 4400 through 447F.  The Intel Evaluation Board
;  decodes a CPU memory access at memory locations C4400 through C447F and
;  generates an I/O access to the 82786.  The 82786 comes up in I/O mode
;  and byte mode after reset.  Access to the registers must be one byte
;  at a time until WORD mode is set.
 mov INTER_RELOC,10h      ; Write low byte into internal relocation register.
 mov INTER_RELOC[1],01h   ; Write high byte into internal relocation register.


;  The following two lines write a value of 0011 (hex) into the BIU control
;  register.  This sets the Internal Register Block for 16-bit WORD access
;  by the External CPU.  All subsequent access to the 82786 registers is by
;  WORD access.
 mov BIU_CONTROL,10h      ; Write low byte into BIU control register
 mov BIU_CONTROL[1],00h   ; Write high byte into BIU control register


mov DRAM_REFRESH,0018h   ; Write value into DRAM refresh control register.


mov DRAM_CONTROL, 001Dh  ; Write value into DRAM control register.


mov DP_PRIORITY, 003Fh   ; Write value into DP priority register.

mov GP_PRIORITY,0009h    ; Write value into GP Priority register

mov EXT_PRIORITY,0028h   ; Write value into External Priority register.


;***************  Clear Page 0 of Graphics memory (64K bytes):    ***************
 mov ax,SEG_GR_MEM               ; Graphics memory space is in the 'A' segment
 mov ds,ax

 mov ax,0
 mov dx,PAGE_PORT
 out dx,ax                       ; Select page 0 of graphics memory

 mov bx,0
 mov cx,32767                    ; 32767 words of memory to be cleared = 64K bytes
 mov si,0
 CLEAR_MEMORY:                   ; Clear page 0 of graphics memory (to be
      mov [si],bx                ; used as a bitmap for drawing commands.)
      add si,2
      loop CLEAR_MEMORY
```

```
;****** Prepare DS, ES, and Dir Flag for use with REP MOVSB instruction. ******
  mov ax,0Fh
  mov dx,PAGE_PORT
  out dx,ax                       ; Select page F of graphics memory

  mov ax,SEG  beg_dp_ctrl_blk
  mov ds,ax                       ; Set data segment
  mov ax,SEG_GR_MEM
  mov es,ax                       ;     and extra segment.
  cld                             ; Clear Direction Flag, sets auto-increment
                                  ;  of SI and DI when using REP instruction.


;***** Copy DP CONTROL BLOCK REGISTERS from CPU memory to Graphics Memory. ****
  lea cx, end_DP_ctrl_blk
  sub cx, offset beg_dp_ctrl_blk
  lea si, beg_dp_ctrl_blk
  mov di, offset DP_REG_MAP
  rep movsb                       ; Move CX bytes from DS:[SI] to ES:[DI]
                                  ; thus, copying DP Control Block Registers
                                  ; from CPU memory to Graphics memory.


;******* Copy DP Descriptor List from CPU memory to Graphics memory. ********
  lea cx, end_dp_descl
  sub cx, offset dp_descl
  lea si, dp_descl
  mov di, offset DESC_PTR_LO
  rep movsb                       ; Move CX bytes from DS:[SI] to ES:[DI]
                                  ; thus copying DP descriptor list from CPU
                                  ; memory to graphics memory.


;*********  Copy GP command list from CPU memory to graphics memory:  *********
  lea cx, len_gp_listl
  sub cx, offset gp_listl
  lea si, gp_listl
  mov di, offset GP_LIST_PTR_LO
  rep movsb                       ; Move CX bytes from DS:[SI] to ES:[DI]
                                  ; thus copying GP command list from CPU
                                  ; memory to graphics memory.


mov ax,register
mov es,ax


*********************  Start up the Display Processor:  ********************
mov DP_PARM1_REG,DP_REG_MAP_LO     ; parameter 1 for dp command
mov DP_PARM2_REG,DP_REG_MAP_HI     ; parameter 2 for dp command
mov DEF_VIDEO_REG,0                ; Write 0 in Default Video register
mov DP_OPCODE_REG, LOADALL         ; Write opcode register, thus starting up
                                   ;     the Display Processor


; *********************  Execute the GP command list:  *******************
  mov GP_PARM1_REG,GP_LIST_PTR_LO    ; parameter 1 for GP command
  mov GP_PARM2_REG,GP_LIST_PTR_HI    ; parameter 2 for GP command
  mov GP_OPCODE_REG, LINK            ; Write opcode register, thus starting
                                     ;    execution of the GP command list.


;************************     Terminate program:     ************************
  mov ah,4Ch                      ; Call BIOS terminate function
  int 21h                         ;     to return to MS-DOS operating system.

 code ENDS


 stack SEGMENT stack              ; Program stack segment
       DW 64 DUP(?)               ; Define unitialized data space for stack.
 stack ENDS

END main
```

3.4 Exercises

This section provides some exercises for the reader in the form of suggested modifications to the Example 1 program. By working through these exercises in succession, the reader will gain an understanding of important concepts and valuable experience in programming the 82786.

Solutions to the Exercises are provided in the Appendix.

Exercise 1:

- Turn cursor off
- Video Status Register, Register 0 in DP Control Block, controls the cursor

Register	Offset			
VSTAT	00	Zero unused upper bits	C	D

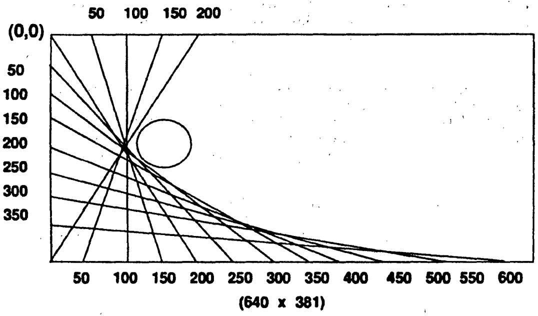
C = 1	Cursor On
C = 0	Cursor Off
D = 1	Display On
D = 0	Display Off

Exercise 2:

Replace GP Command List in Example 1 with new GP Command List to draw the straight lines in the graphic

Center of Circle at (200, 182)

Radius = 50



240048-26

Hint: Replace GP Command List in Exercise 1 with a new Command List. See description of Abs_Mov, Line, and Circle commands.

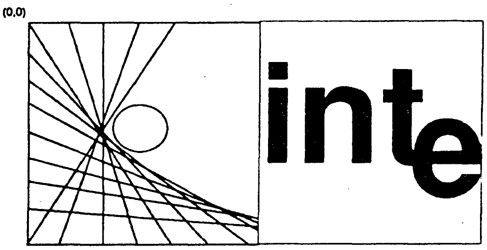
Exercise 3:

- Modify the program from Exercise 2.
- Change from 1-bit per pixel (bpp) to 4 bpp.
- Use the Def__Color Command to change the color of each line in the drawing.

Hint: Change Def__Bitmap parameters and Tile Descriptors. Clear an additional page (Page 1) of Graphics Memory to allow room for the larger bitmap.

Exercise 4:

- Write a new DP Descriptor List and turn on the borders for two windows, not overlapping, as shown below.
- The left window should contain the 4 bpp multi-colored image drawn in Exercise 3.
- The right window should contain the 1 bpp image drawn in Exercise 1 (the Intel Logo).
- Change 1 bpp pad register to accentuate the two different windows.

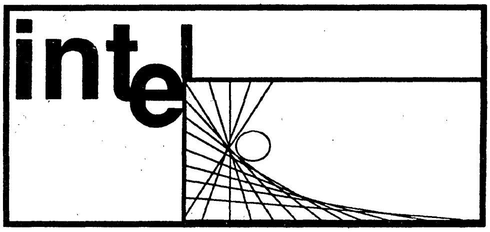


240048-27

Hint: Change Def__Bitmap parameters. Modify strip and tile descriptors. Combine the two GP command lists from Exercise 3 and Example 1 programs. Before starting the second command list, be sure to use a new Def__Bitmap command. Clear page 2 of Graphics memory.

Exercise 5:

Same as Exercise 4, except make the two windows overlap as shown below.



240048-28

Hint: Create two strips.

Strip 1: Contains 1 tile consisting of 100 lines.

Strip 2: Contains 2 tiles consisting of 281 lines; the first tile is 320 pixels wide.

CHAPTER 4 QUICK REFERENCE SECTION

4.0 INTRODUCTION

This Chapter provides a compilation of data frequently used by 82786 programmers. It contains data for all 82786 registers, commands, command parameters, opcodes, strip and tile descriptor format, video timing parameters.

4.1 82786 Directly Accessible (Internal) Registers

	Register	Offset (H)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Internal Relocation	00	Base Address															MIO
BIU '00-OFH	Reserved	02	Reserved (zero for future compatibility)															
	BIU Control	04	Reserved (zero for future compatibility)									VR	WT	BCP	GI	DI	WP1	WP2
	Refresh Control	06	Reserved (zero for future compatibility)									Refresh Scaler						
	DRAM/VRAM Control	08	Reserved (zero for future compatibility)									RW1	RW0	DC1	DC0	HT2	HT1	HT0
	Display Priority	0A	Reserved (zero for future compatibility)										FPL			SPL		
	GP Priority	0C	Reserved (zero for future compatibility)										FPL			SPL		
	External Priority	0E	Reserved (zero for future compatibility)										FPL			Reserved		
Reserved '10-1FH	Reserved	10	Reserved (zero for future compatibility)															
		12	Reserved (zero for future compatibility)															
		14	Reserved (zero for future compatibility)															
		16	Reserved (zero for future compatibility)															
		18	Reserved (zero for future compatibility)															
		1A	Reserved (zero for future compatibility)															
		1C	Reserved (zero for future compatibility)															
		1E	Reserved (zero for future compatibility)															
GP '20-2BH	GR0 Opcode	20	Opcode								Reserved (zero for future compatibility)							GECL
	GR1 Parameter 1	22	Link Address (Lower)															
	GR2 Parameter 2	24	Reserved										Link Address (Upper)					
	Status Register (GSTAT)	26	Reserved								GPOLL	GRCD	GINT	GPSC	GBCOV	GBMOV	GCTP	GIBMD
	Instruction Pointer	28	Instruction Pointer (Lower)															
		2A	Reserved (zero for future compatibility)										Instruction Pointer (Upper)					
Reserved '2C-3FH		2C	Reserved (zero for future compatibility)															
		2E	Reserved (zero for future compatibility)															
		30	Reserved (zero for future compatibility)															
		32	Reserved (zero for future compatibility)															
		34	Reserved (zero for future compatibility)															
		36	Reserved (zero for future compatibility)															
		38	Reserved (zero for future compatibility)															
		3A	Reserved (zero for future compatibility)															
		3C	Reserved (zero for future compatibility)															
		3E	Reserved (zero for future compatibility)															
DP '40-4BH	Opcode	40	Opcode								Reserved (zero for future compatibility)							ECL
	Parameter 1	42	Memory Address (Lower)															
	Parameter 2	44	Reserved (zero for future compatibility)										Memory Address (Upper)					
	Parameter 3	46	Reserved (zero for future compatibility)										Register Identification					
	Status Register	48	Reserved								FRI	RCD	DOV	FMT	BLK	EVN	ODD	ECL
	Default Video	4A	Reserved								Default Video							
Reserved '4C-7FH	Reserved	4C	Reserved (zero for future compatibility)															
		4E	Reserved (zero for future compatibility)															
		50	Reserved (zero for future compatibility)															
		52	Reserved (zero for future compatibility)															
		54	Reserved (zero for future compatibility)															
		56	Reserved (zero for future compatibility)															
		58	Reserved (zero for future compatibility)															
		5A	Reserved (zero for future compatibility)															
		5C	Reserved (zero for future compatibility)															
		5E	Reserved (zero for future compatibility)															
		60	Reserved (zero for future compatibility)															
		62	Reserved (zero for future compatibility)															
		64	Reserved (zero for future compatibility)															
		66	Reserved (zero for future compatibility)															
		68	Reserved (zero for future compatibility)															
		6A	Reserved (zero for future compatibility)															
		6C	Reserved (zero for future compatibility)															
		6E	Reserved (zero for future compatibility)															
		70	Reserved (zero for future compatibility)															
		72	Reserved (zero for future compatibility)															
		74	Reserved (zero for future compatibility)															
		76	Reserved (zero for future compatibility)															
		78	Reserved (zero for future compatibility)															
		7A	Reserved (zero for future compatibility)															
		7C	Reserved (zero for future compatibility)															
		7E	Reserved (zero for future compatibility)															

240048-29

82786 128-byte Internal Register Block

4.2 GP Indirectly Accessible Registers

Context Registers

Name	ID	Bits	Function
GCOMM	0001	(16)	Command
GPOEM	0003	(6)	Poll Mask
GIMR	0004	(6)	Interrupt Mask
GCHOR	0007	(2,2)	Character Orientation and Path*
GCHA	010B	(21)	Character Font Base Address
GSP	010C	(21)	Stack Pointer
GCA	010D	(21)	Memory Address of Current Position (x, y)
GBORG	010F	(21)	Bitmap Origin Address
GCX	0010	(16)	Current X Position
GCY	0011	(16)	Current Y Position
GPAT	0012	(16)	Line Pattern
GSPAC	0013	(16)	Spacing between Characters and All Bitblts
GCNT	0014	(16)	Character Count**
GN	0016	(16)	Number of 16-bit Words Spanning Width of Bitmap
GVERS	0017	(16)	Version Number*** (D Step Value = 5)
GXMAX	0090	(16)	Maximum X for Clipping Rectangle
GYMAX	0091	(16)	Maximum Y for Clipping Rectangle
GXMIN	0094	(16)	Minimum X for Clipping Rectangle^
GYMIN	0095	(16)	Minimum Y for Clipping Rectangle^
GMASK	0099	(16)	Pixel Mask
GBGC	009B	(16)	Background Color
GFGC	009C	(16)	Foreground Color
GFCODE	009E	(4)	Function Code for Pixel Updates^^
GCIP	01AC	(21)	Current Instruction Pointer
GBPP (RO)	009F	(4)	Used with Dump Register command to get Current Bits per Pixel Address ^^^
GBPP (WO)	0008	(4)	Used with Load Register command to write Current Bits per Pixel Address ^^^

* These bits are right justified in each byte of the word in which they are stored. Two bits are stored in bits 1 and 0 and two bits are stored in bits 8 and 9; the remaining upper bits in each byte are zeroed.

** GCNT ID reassigned from 0015 to 0014 in D-Step.

*** In D-Step, valid after RESET and prior to drawing or drawing control commands.

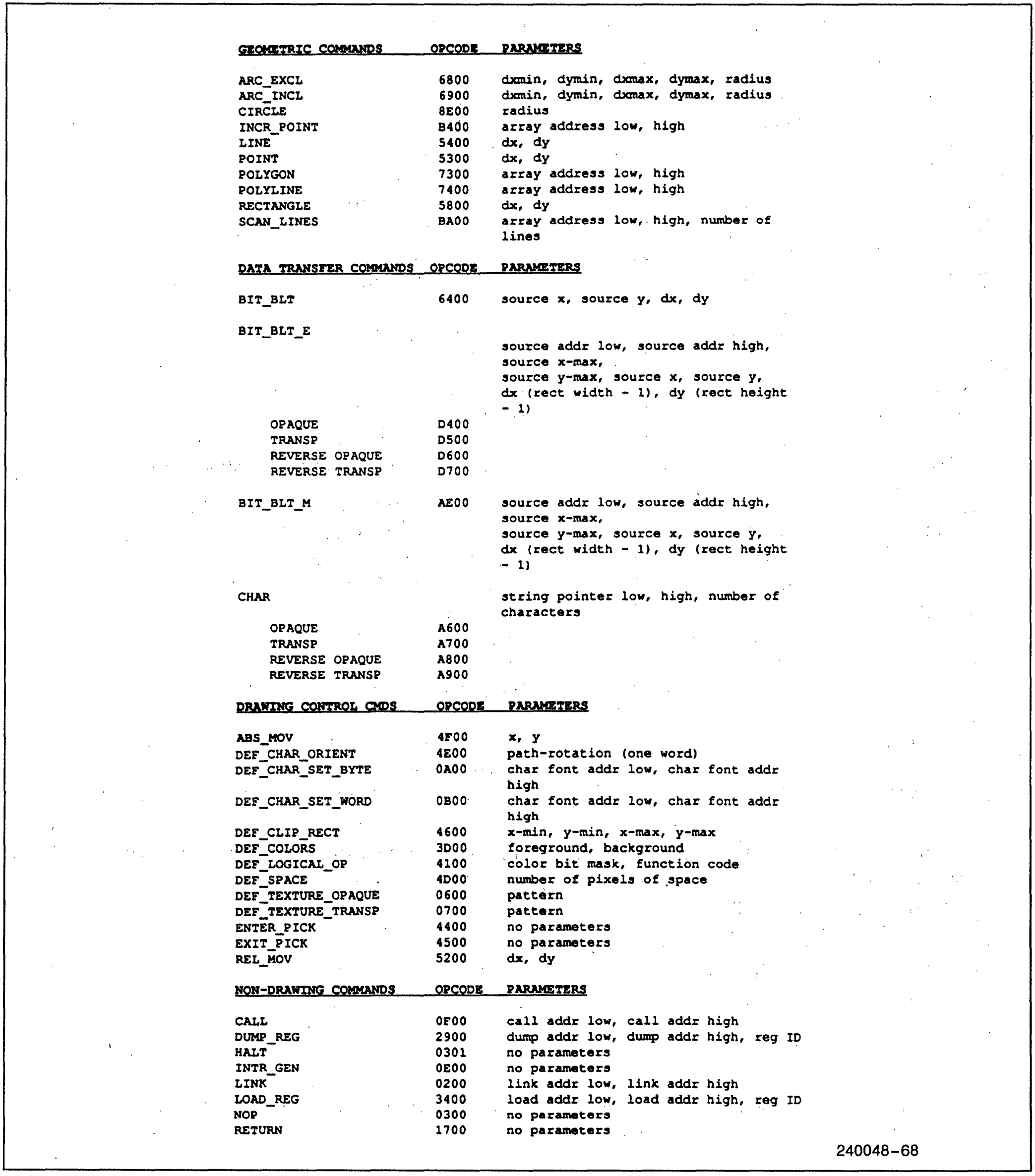
^ Correction to previous GXMIN ID 0096 and GYMIN 0097 assignments.

^^ GFCODE ID reassigned from 001C to 009F in D-Step.

^^^ New D-Step Bpp Registersd.

NOTE:
Simply saving and restoring the context registers is not sufficient to restore the state of the graphics processor.

4.3 GP Commands, Opcodes, Parameters



GEOMETRIC COMMANDS	OPCODE	PARAMETERS
ARC_EXCL	6800	dxmin, dymin, dxmax, dymax, radius
ARC_INCL	6900	dxmin, dymin, dxmax, dymax, radius
CIRCLE	8E00	radius
INCR_POINT	B400	array address low, high
LINE	5400	dx, dy
POINT	5300	dx, dy
POLYGON	7300	array address low, high
POLYLINE	7400	array address low, high
RECTANGLE	5800	dx, dy
SCAN_LINES	BA00	array address low, high, number of lines
DATA TRANSFER COMMANDS	**OPCODE**	**PARAMETERS**
BIT_BLT	6400	source x, source y, dx, dy
BIT_BLT_E		source addr low, source addr high, source x-max, source y-max, source x, source y, dx (rect width - 1), dy (rect height - 1)
OPAQUE	D400	
TRANSP	D500	
REVERSE OPAQUE	D600	
REVERSE TRANSP	D700	
BIT_BLT_M	AE00	source addr low, source addr high, source x-max, source y-max, source x, source y, dx (rect width - 1), dy (rect height - 1)
CHAR		string pointer low, high, number of characters
OPAQUE	A600	
TRANSP	A700	
REVERSE OPAQUE	A800	
REVERSE TRANSP	A900	
DRAWING CONTROL CMDS	**OPCODE**	**PARAMETERS**
ABS_MOV	4F00	x, y
DEF_CHAR_ORIENT	4E00	path-rotation (one word)
DEF_CHAR_SET_BYTE	0A00	char font addr low, char font addr high
DEF_CHAR_SET_WORD	0B00	char font addr low, char font addr high
DEF_CLIP_RECT	4600	x-min, y-min, x-max, y-max
DEF_COLORS	3D00	foreground, background
DEF_LOGICAL_OP	4100	color bit mask, function code
DEF_SPACE	4D00	number of pixels of space
DEF_TEXTURE_OPAQUE	0600	pattern
DEF_TEXTURE_TRANSP	0700	pattern
ENTER_PICK	4400	no parameters
EXIT_PICK	4500	no parameters
REL_MOV	5200	dx, dy
NON-DRAWING COMMANDS	**OPCODE**	**PARAMETERS**
CALL	0F00	call addr low, call addr high
DUMP_REG	2900	dump addr low, dump addr high, reg ID
HALT	0301	no parameters
INTR_GEN	0E00	no parameters
LINK	0200	link addr low, link addr high
LOAD_REG	3400	load addr low, load addr high, reg ID
NOP	0300	no parameters
RETURN	1700	no parameters

240048-68

4.4 DP Indirectly Accessible Registers (DP Control Block Registers)

Register	Offset (H)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Video Status	00	Reserved														C	D
Interrupt Mask	01	Reserved								FRI	RCD	DOV	FMT	BLK	EVN	ODD	ECL
	02	Reserved											Trip Point				
	03	Reserved								Frame Interrupt							
	04	Reserved															
CRTMode	05	Reserved									I	L	W	S	B	A	A

Interlace - (2) IL
01 Reserved
00 Noninterlace
10 Interlace
11 Interlace-Sync

Window Status Enable (1)

HSync/VSync Slave Mode (1)

Blank Slave Mode

Accelerated Video (2)
00 Normal (25 MHz)
01 High-Speed (50 MHz)
10 Very High-Speed (100 MHz)
11 Super High-Speed (200 MHz)

Register	Offset (H)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	06	Reserved				Horizontal Synchronization Stop (HSyncStp)											
	07	Reserved				Horizontal Field Start (HFldStrt)											
	08	Reserved				Horizontal Field Stop (HFldStp)											
	09	Reserved				Line Length (LineLen)											
	0A	Reserved				Vertical Synchronization Stop (VSyncStp)											
	0B	Reserved				Vertical Field Start (VFldStrt)											
	0C	Reserved				Vertical Field Stop (VFldStp)											
	0D	Reserved				Frame Length (FrameLen)											
	0E	Descriptor Address Pointer (Lower)															
	0F	Reserved											Descriptor Address Pointer (Upper)				
	10	Reserved															
	11	Reserved		XZoom						Reserved			YZoom				
	12	Reserved								Field Color (FldColor)							
	13	Reserved								Border Color (BdrColor)							
	14	Reserved								1 Bpp Pad							Res.
	15	Reserved								2 Bpp Pad						Reserved	
	16	Reserved								4 Bpp Pad				Reserved			
CsrMode	17	S	X	T	CSt		CSC		Res.	CsrPad							Res.

CsrStyle: S = Cursor Size (1)
CsrSize 0 = 8x8 Csr
1 = 16x16 Csr
X = Crosshair Cursor (1)
T = Transparent Cursor (1)
CSt = Cursor Status to Window Status Output (2)
CSC Cursor Status Control (2)
00 = Current Window Status
01 = Foreground
10 = Background
11 = Block

Register	Offset (H)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	18	Reserved				Cursor Position X (CsrPosX)											
	19	Reserved				Cursor Position Y (CsrPosY)											
	1A	Cursor Pattern 0 (CsrPat0)															
	1B	Cursor Pattern 1 (CsrPat1)															
	1C	Cursor Pattern 2 (CsrPat2)															
	1D	Cursor Pattern 3 (CsrPat3)															
	1E	Cursor Pattern 4 (CsrPat4)															
	1F	Cursor Pattern 5 (CsrPat5)															
	20	Cursor Pattern 6 (CsrPat6)															
	21	Cursor Pattern 7 (CsrPat7)															
	22	Cursor Pattern 8 (CsrPat8)															
	23	Cursor Pattern 9 (CsrPat9)															
	24	Cursor Pattern A (CsrPatA)															
	25	Cursor Pattern B (CsrPatB)															
	26	Cursor Pattern C (CsrPatC)															
	27	Cursor Pattern D (CsrPatD)															
	28	Cursor Pattern E (CsrPatE)															
	29	Cursor Pattern F (CsrPatF)															

240048–30

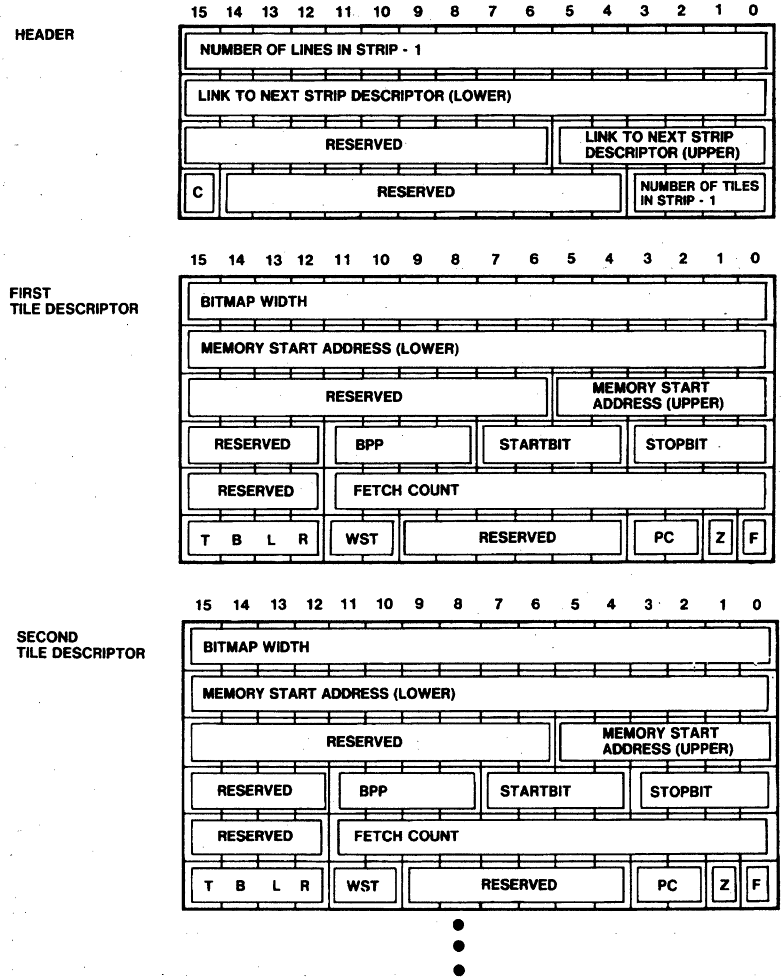
4.5 DP Commands, Opcodes, Parameters

DP COMMANDS	OPCODE	PARAMETERS
LOAD_REG	0400	load addr low, load addr high, reg ID
LOAD_ALL	0500	load addr low, load addr high
DUMP_REG	0600	dump addr low, load addr high, reg ID
DUMP_ALL	0700	dump addr low, load addr high

240048-69

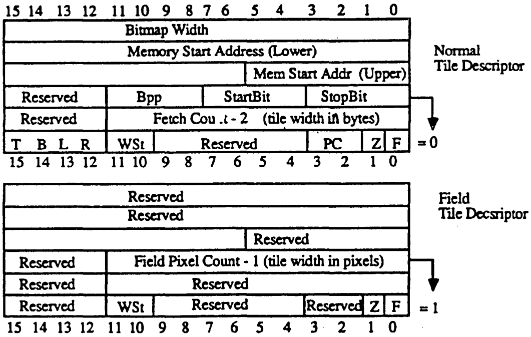
4.6 Strip and Tile Descriptor Formats

Strip and Tile Descriptors



240048-20

Field Tile Descriptor Format



240048-22

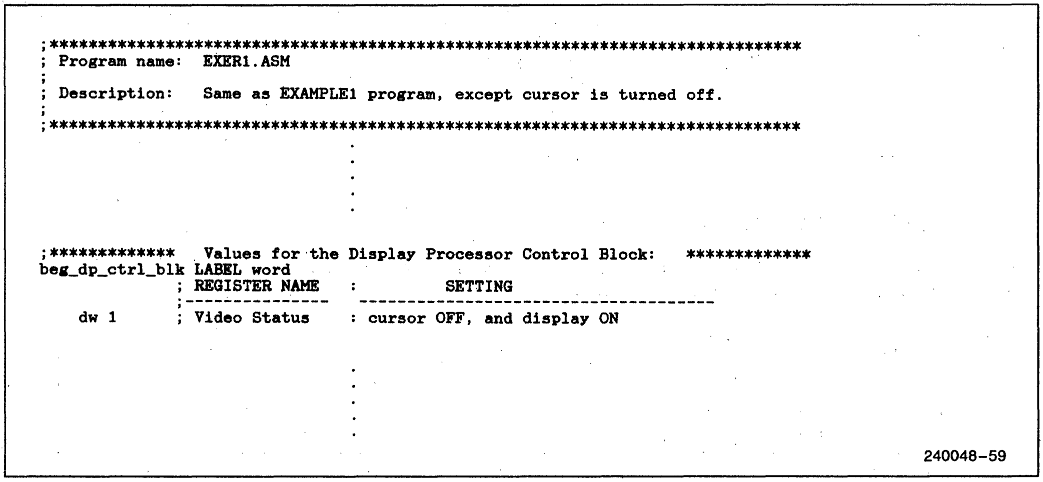
4.7 Example Video Timing Parameters

The following table of Video Timing Parameters satisfy the requirements of NEC Multisync-compatible monitors. These parameters provide the given resolution and refresh rate when using the VCLK frequency indicated in the table.

VCLK (MHz)	25	25	20	20	20
Xmax	640	512	512	640	640
Ymax	480	512	512	350	455
Screen Refresh (Hz)	60	60	60	60	60
HSync Stop	75	97	47	89	37
HFld Start	145	184	94	168	77
HFld Stop	785	696	606	808	717
Line Len	827	752	633	861	737
VSync Stop	3	6	2	5	0
VFld Start	14	26	8	23	3
VFld Stop	494	538	520	373	458
Frame Len	501	551	524	385	458

APPENDIX A
SOLUTIONS TO EXERCISES

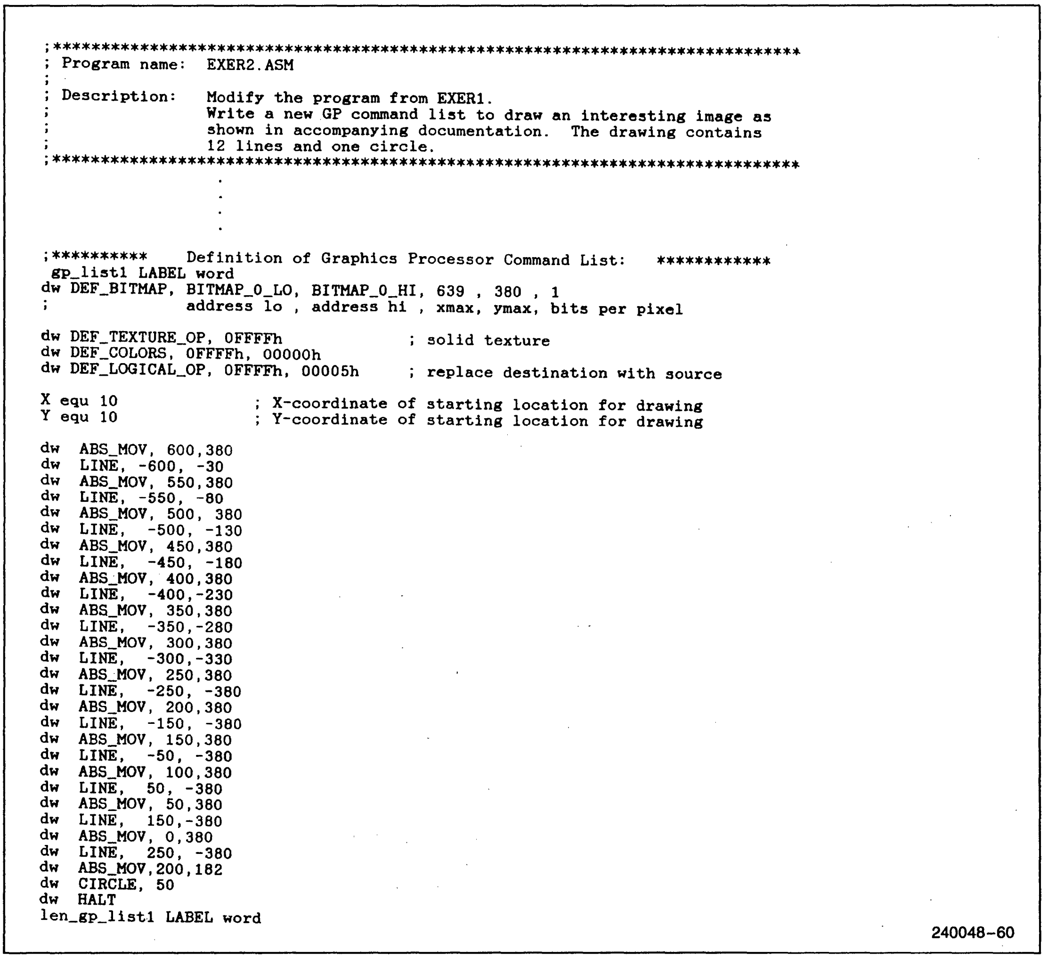
SOLUTION FOR EXERCISE 1



```
;*******************************************************************************
; Program name:  EXER1.ASM
;
; Description:   Same as EXAMPLE1 program, except cursor is turned off.
;
;*******************************************************************************
                                .
                                .
                                .
                                .
                                .


;**************  Values for the Display Processor Control Block:   *************
beg_dp_ctrl_blk LABEL word
              ; REGISTER NAME   :         SETTING
              ;---------------   -------------------------------------
    dw 1      ; Video Status    : cursor OFF, and display ON


                                .
                                .
                                .
                                .
                                .
```

240048-59

SOLUTION FOR EXERCISE 2



```
;****************************************************************************
; Program name:  EXER2.ASM
;
; Description:   Modify the program from EXER1.
;                Write a new GP command list to draw an interesting image as
;                shown in accompanying documentation.  The drawing contains
;                12 lines and one circle.
;****************************************************************************
                         .
                         .
                         .
                         .

;**********     Definition of Graphics Processor Command List:    ************
 gp_list1 LABEL word
dw DEF_BITMAP, BITMAP_0_LO, BITMAP_0_HI, 639 , 380 , 1
;              address lo , address hi , xmax, ymax, bits per pixel

dw DEF_TEXTURE_OP, 0FFFFh              ; solid texture
dw DEF_COLORS, 0FFFFh, 00000h
dw DEF_LOGICAL_OP, 0FFFFh, 00005h      ; replace destination with source

X equ 10                 ; X-coordinate of starting location for drawing
Y equ 10                 ; Y-coordinate of starting location for drawing

dw  ABS_MOV, 600,380
dw  LINE, -600, -30
dw  ABS_MOV, 550,380
dw  LINE, -550, -80
dw  ABS_MOV, 500, 380
dw  LINE,  -500, -130
dw  ABS_MOV, 450,380
dw  LINE,  -450, -180
dw  ABS_MOV, 400,380
dw  LINE,  -400,-230
dw  ABS_MOV, 350,380
dw  LINE,  -350,-280
dw  ABS_MOV, 300,380
dw  LINE,  -300,-330
dw  ABS_MOV, 250,380
dw  LINE,  -250, -380
dw  ABS_MOV, 200,380
dw  LINE,  -150, -380
dw  ABS_MOV, 150,380
dw  LINE,  -50, -380
dw  ABS_MOV, 100,380
dw  LINE,  50, -380
dw  ABS_MOV, 50,380
dw  LINE,  150,-380
dw  ABS_MOV, 0,380
dw  LINE,  250, -380
dw  ABS_MOV,200,182
dw  CIRCLE, 50
dw  HALT
len_gp_list1 LABEL word
```

240048-60

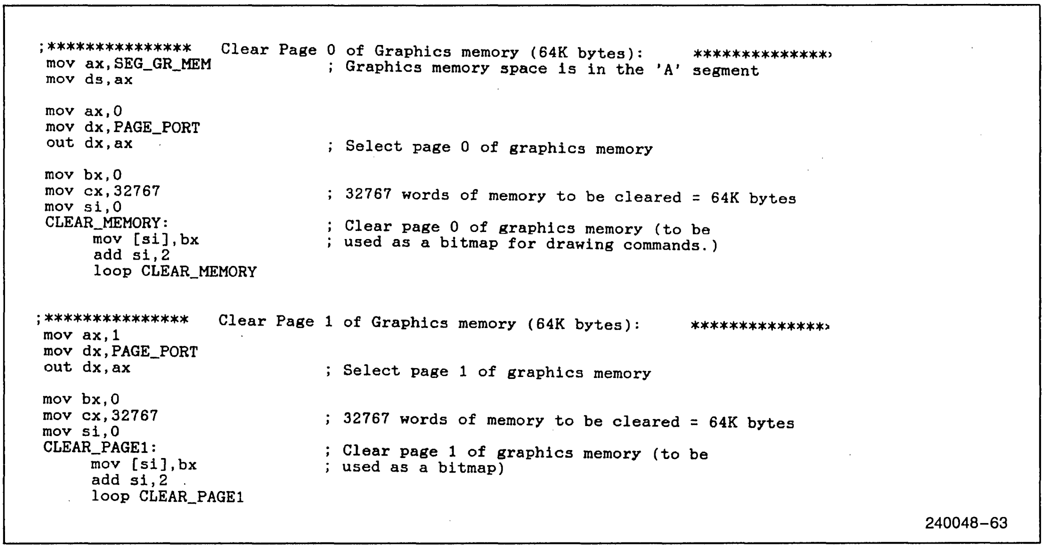
SOLUTION FOR EXERCISE 3

```
;*****************************************************************************
; Program name:  EXER3.ASM
; Description:   Modify the program from Exercise 2.  Change the bitmap
;                from 1 bit per pixel to 4 bits per pixel.
;                Use the DEF_COLORS command to change the color of each line
;                in the drawing.
;*****************************************************************************
                     .
                     .
                     .
                     .

;************   Definition of Display Processor Descriptor List:   ************
dp_desc1 LABEL word
; Header of DP descriptor:
    dw 380           ;  (number of lines - 1)
    dw DESC_PTR_LO+20 ;  lower link to next strip descriptor (there is none,
                     ;      but if one were added, this is the link)
    dw DESC_PTR_HI   ;  upper link to next strip descriptor (there is none)
    dw 0             ;  (number of tiles - 1)
; First (and only) Tile Descriptor
    dw 0320          ;  Bitmap width (number of bytes)
    dw 0000h         ;  Bitmap start address lower
    dw 0000h         ;  Bitmap start address upper
    dw 04F0h         ;  4 bpp, start bit F, stop bit 0
    dw 0318          ;  Fetch count = (number of bytes - 2)
    dw 0F000h        ;  All 4 borders on,window status=0,PC mode off,field off
end_dp_desc1 LABEL word ; ***********  End of DP descriptor list. *********
```

240048-61

```
;**********   Definition of Graphics Processor Command List:   ************
 gp_list1 LABEL word
dw DEF_BITMAP, BITMAP_0_LO, BITMAP_0_HI, 639 , 380 , 4
;              address lo , address hi , xmax, ymax, bits per pixel
dw DEF_TEXTURE_OP, 0FFFFh               ; solid texture
dw DEF_COLORS, 0FFFFh, 0
dw DEF_LOGICAL_OP, 0FFFFh, 00005h       ; replace destination with source

X equ 10               ; X-coordinate of starting location for drawing
Y equ 10               ; Y-coordinate of starting location for drawing

dw  ABS_MOV, 600,380
dw  LINE, -600, -30
dw  ABS_MOV, 550,380
dw  DEF_COLORS, 1110111011101110b, 0   ; foreground color is 1110 (binary); must
                                       ;   be repeated to fill the entire word.
dw  LINE, -550, -80
dw  ABS_MOV, 500, 380
dw  DEF_COLORS, 1101110111010010b, 0   ; foreground color is 1101 (binary).
dw  LINE,  -500, -130
dw  ABS_MOV, 450,380
dw  DEF_COLORS, 1100110011001100b, 0   ; foreground color is 1100 (binary).
dw  LINE,  -450, -180
dw  ABS_MOV, 400,380
dw  DEF_COLORS, 1011101110111011b, 0   ; foreground color is 1011 (binary).
dw  LINE,  -400,-230
dw  ABS_MOV, 350,380
dw  DEF_COLORS, 1010101010101010b, 0   ; foreground color is 1010 (binary).
dw  LINE,  -350,-280
dw  ABS_MOV, 300,380
dw  DEF_COLORS, 1001100110011001b, 0   ; foreground color is 1001 (binary).
dw  LINE,  -300,-330
dw  ABS_MOV, 250,380
dw  DEF_COLORS, 1000100010001000b, 0   ; foreground color is 1000 (binary).
dw  LINE,  -250, -380
dw  ABS_MOV, 200,380
dw  DEF_COLORS, 0111011101110111b, 0   ; foreground color is 0111 (binary).
dw  LINE,  -150, -380
dw  ABS_MOV, 150,380
dw  DEF_COLORS, 0110011001100110b, 0   ; foreground color is 0110 (binary).
dw  LINE,  -50, -380
dw  ABS_MOV, 100,380
dw  DEF_COLORS, 0101010101010101b, 0   ; foreground color is 0101 (binary).
dw  LINE,  50, -380
dw  ABS_MOV, 50,380
dw  DEF_COLORS, 0100010001000100b, 0   ; foreground color is 0100 (binary).
dw  LINE,  150,-380
dw  ABS_MOV, 0,380
dw  DEF_COLORS, 0011001100110011b, 0   ; foreground color is 0011 (binary).
dw  LINE,  250, -380
dw  ABS_MOV,200,182
dw  DEF_COLORS, 0010001000100010b, 0   ; foreground color is 0010 (binary).
dw  CIRCLE, 50
dw  HALT
len_gp_list1 LABEL word
```

240048-62



```
;****************   Clear Page 0 of Graphics memory (64K bytes):      ****************
 mov ax,SEG_GR_MEM              ; Graphics memory space is in the 'A' segment
 mov ds,ax

 mov ax,0
 mov dx,PAGE_PORT
 out dx,ax                      ; Select page 0 of graphics memory

 mov bx,0
 mov cx,32767                   ; 32767 words of memory to be cleared = 64K bytes
 mov si,0
 CLEAR_MEMORY:                  ; Clear page 0 of graphics memory (to be
      mov [si],bx               ; used as a bitmap for drawing commands.)
      add si,2
      loop CLEAR_MEMORY

;****************   Clear Page 1 of Graphics memory (64K bytes):      ****************
 mov ax,1
 mov dx,PAGE_PORT
 out dx,ax                      ; Select page 1 of graphics memory

 mov bx,0
 mov cx,32767                   ; 32767 words of memory to be cleared = 64K bytes
 mov si,0
 CLEAR_PAGE1:                   ; Clear page 1 of graphics memory (to be
      mov [si],bx               ; used as a bitmap)
      add si,2
      loop CLEAR_PAGE1
```

240048-63

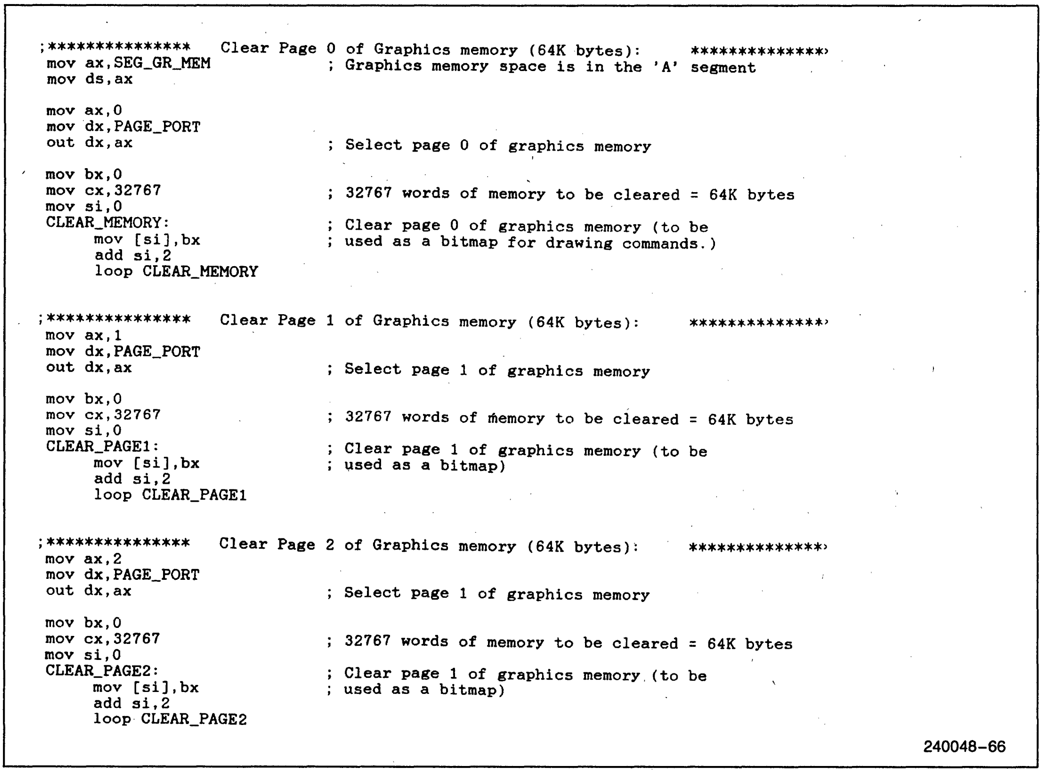
SOLUTION FOR EXERCISE 4

```
;*******************************************************************************
; Program name:  EXER4.ASM
;
; Description:   Modify the program from Exercise 3.  Modify the DP descriptor
;                list for 2 windows, not overlapping, as shown in the
;                accompanying documentation.  The left window should contain
;                the 4 BPP multi-colored image drawn in EXERCISE 3.
;                The right window should contain the 1 BPP image drawn in the
;                EXAMPLE1 program (the Intel logo).
;*******************************************************************************
                          .
                          .
                          .
                          .
                          .



;************   Definition of Display Processor Descriptor List:   ************
dp_desc1 LABEL word
; Header of DP descriptor:
    dw 380           ;  (number of lines - 1)
    dw DESC_PTR_LO+20 ;  lower link to next strip descriptor (there is none,
                     ;      but if one were added, this is the link)
    dw DESC_PTR_HI   ;  upper link to next strip descriptor (there is none)
    dw 1             ;  (number of tiles - 1)
; First Tile Descriptor:
    dw 0320          ;  Bitmap width (number of bytes)
    dw 7720h         ;  Bitmap start address lower
    dw 0000h         ;  Bitmap start address upper
    dw 04F0h         ;  4 bpp, start bit F, stop bit 0
    dw 0158          ;  Fetch count = (number of bytes - 2)
    dw 0F000h        ;  All 4 borders on,window status=0,PC mode off,field off
; Second Tile Descriptor:
    dw 0080          ;  Bitmap width (number of bytes)
    dw 0000h         ;  Bitmap start address lower
    dw 0000h         ;  Bitmap start address upper
    dw 01F0h         ;  1 bpp, start bit F, stop bit 0
    dw 0038          ;  Fetch count = (number of bytes - 2)
    dw 0F000h        ;  All 4 borders on,window status=0,PC mode off,field off
end_dp_desc1 LABEL word ; ***********  End of DP descriptor list. *********
```

240048-64

```
;**********    Definition of Graphics Processor Command List:    ************
 gp_list1 LABEL word




dw DEF_BITMAP, BITMAP_0_LO, BITMAP_0_HI, 639 , 380 , 1
;              address lo , address hi , xmax, ymax, bits per pixel

dw DEF_TEXTURE_OP, 0FFFFh              ; solid texture
dw DEF_COLORS, 0FFFFh, 00000h
dw DEF_LOGICAL_OP, 0FFFFh, 00005h      ; replace destination with source

X equ 10               ; X-coordinate of starting location for drawing
Y equ 10               ; Y-coordinate of starting location for drawing

;*************************        Draw Intel logo:      ***********************
dw  ABS_MOV, X, Y      ; Move to beginning position for drawing.
dw  LINE,  35,   0     ; Dot the "i"
dw  LINE,   0,  35
           .
           .
           .
           .
           .


;*****************   Draw figure from EXER3 program:      ***********************
dw DEF_BITMAP, 7720h, 0, 639 , 380 , 4
;              address lo , address hi , xmax, ymax, bits per pixel
dw  ABS_MOV, 600,380
dw  LINE, -600, -30
dw  ABS_MOV, 550,380
dw  DEF_COLORS, 1110111011101110b, 0   ; foreground color is 1110 (binary); must
                                       ;    be repeated to fill the entire word.
dw  LINE, -550, -80

           .
           .
           .
           .
           .
```

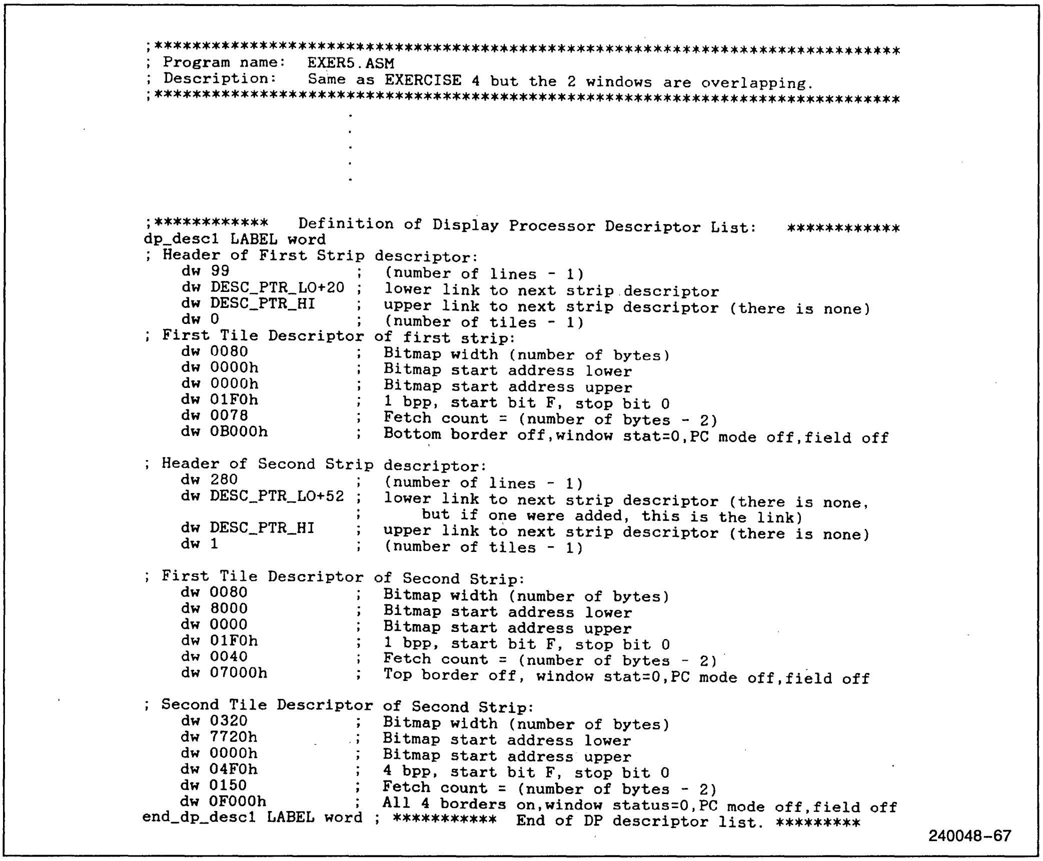
240048-65



```
;****************   Clear Page 0 of Graphics memory (64K bytes):     ***************›
 mov ax,SEG_GR_MEM             ; Graphics memory space is in the 'A' segment
 mov ds,ax

 mov ax,0
 mov dx,PAGE_PORT
 out dx,ax                     ; Select page 0 of graphics memory

 mov bx,0
 mov cx,32767                  ; 32767 words of memory to be cleared = 64K bytes
 mov si,0
 CLEAR_MEMORY:                 ; Clear page 0 of graphics memory (to be
      mov [si],bx              ; used as a bitmap for drawing commands.)
      add si,2
      loop CLEAR_MEMORY

;****************   Clear Page 1 of Graphics memory (64K bytes):     ***************›
 mov ax,1
 mov dx,PAGE_PORT
 out dx,ax                     ; Select page 1 of graphics memory

 mov bx,0
 mov cx,32767                  ; 32767 words of memory to be cleared = 64K bytes
 mov si,0
 CLEAR_PAGE1:                  ; Clear page 1 of graphics memory (to be
      mov [si],bx              ; used as a bitmap)
      add si,2
      loop CLEAR_PAGE1

;****************   Clear Page 2 of Graphics memory (64K bytes):     ***************›
 mov ax,2
 mov dx,PAGE_PORT
 out dx,ax                     ; Select page 1 of graphics memory

 mov bx,0
 mov cx,32767                  ; 32767 words of memory to be cleared = 64K bytes
 mov si,0
 CLEAR_PAGE2:                  ; Clear page 1 of graphics memory (to be
      mov [si],bx              ; used as a bitmap)
      add si,2
      loop CLEAR_PAGE2
```

240048-66

SOLUTION FOR EXERCISE 5



```
;******************************************************************************
; Program name:  EXER5.ASM
; Description:   Same as EXERCISE 4 but the 2 windows are overlapping.
;******************************************************************************
                     .
                     .
                     .
                     .
                     .


;************   Definition of Display Processor Descriptor List:   ************
dp_desc1 LABEL word
; Header of First Strip descriptor:
    dw 99             ;  (number of lines - 1)
    dw DESC_PTR_LO+20 ;  lower link to next strip descriptor
    dw DESC_PTR_HI    ;  upper link to next strip descriptor (there is none)
    dw 0              ;  (number of tiles - 1)
; First Tile Descriptor of first strip:
    dw 0080           ;  Bitmap width (number of bytes)
    dw 0000h          ;  Bitmap start address lower
    dw 0000h          ;  Bitmap start address upper
    dw 01F0h          ;  1 bpp, start bit F, stop bit 0
    dw 0078           ;  Fetch count = (number of bytes - 2)
    dw 0B000h         ;  Bottom border off,window stat=0,PC mode off,field off

; Header of Second Strip descriptor:
    dw 280            ;  (number of lines - 1)
    dw DESC_PTR_LO+52 ;  lower link to next strip descriptor (there is none,
                      ;      but if one were added, this is the link)
    dw DESC_PTR_HI    ;  upper link to next strip descriptor (there is none)
    dw 1              ;  (number of tiles - 1)

; First Tile Descriptor of Second Strip:
    dw 0080           ;  Bitmap width (number of bytes)
    dw 8000           ;  Bitmap start address lower
    dw 0000           ;  Bitmap start address upper
    dw 01F0h          ;  1 bpp, start bit F, stop bit 0
    dw 0040           ;  Fetch count = (number of bytes - 2)
    dw 07000h         ;  Top border off, window stat=0,PC mode off,field off

; Second Tile Descriptor of Second Strip:
    dw 0320           ;  Bitmap width (number of bytes)
    dw 7720h          ;  Bitmap start address lower
    dw 0000h          ;  Bitmap start address upper
    dw 04F0h          ;  4 bpp, start bit F, stop bit 0
    dw 0150           ;  Fetch count = (number of bytes - 2)
    dw 0F000h         ;  All 4 borders on,window status=0,PC mode off,field off
end_dp_desc1 LABEL word ; ***********  End of DP descriptor list. *********
```

240048-67

October 1987

82786 Design Example Interfacing to the IBM PC/AT* Computer

RICHARD SHANKMAN
APPLICATIONS ENGINEER

*IBM PC/AT is a trademark of International Business Machines Corporation.

Order Number: 240049-001

1.0 INTRODUCTION

Many applications require greater graphics capability than is available through IBM's CGA or EGA. The 82786 allows the design of very high performance graphics systems at low cost, both in terms of component count and development time.

This application note will present a basic design interfacing a graphics board based on the 82786 to the IBM PC/AT computer. Only those portions of the design related to the interface itself will be covered in detail. Other aspects of graphics system design using the 82786, such as graphics memory design and video interfacing are covered in detail in the Hardware Configuration Application Note (AP-270—refer to section 1.1 below on related literature).

Throughout this application note the following naming convention applies:

The term "PC" will be used throughout this document to refer to both IBM's 8-bit PC and their 16-bit AT computer systems.

1.1 Related Literature

Additional material concerning the 82786 can be found in the following Intel publications:

82786 Graphics Coprocessor User's Manual, Order Number 231933

82786 CHMOS Graphics Coprocessor Data Sheet, Order Number 231676

82786 Hardware Configuration Application Note, Order Number 292007

An Introduction To Programming the 82786 Graphics Coprocessor, Order Number 240048

2.0 I/O CHANNEL

2.1 Overview

There are eight connector slots on the mother board of the PC into which peripheral cards may be inserted. All interface to the PC is through these connectors, which are known as the I/O CHANNEL.

The I/O CHANNEL supports 24-bit memory addresses, data accesses of either 8- of 16-bits, interrupts, DMA channels, and wait state generation. The connectors consist of eight 62-pin and six 36-pin connector sockets. The two positions that have only the 62-pin connectors can only support an 8-bit IBM PC interface.

2.2 Address Map

When placed in Protected Mode, a full 24 bits (16 Mbytes) of addressing are available. However, most applications use Real Mode, providing 20 bits of addressing. This provides a usable address space of 1 Mbyte. Our design example will use Real Mode.

As shown in Figure 1, the lower 512 kb of the 1 Mb address space is reserved for system memory. 384 kbytes of the upper 512 kb are available for our use, although various adaptor cards which use some of this space may be installed in the system. We need to use care in selecting where our graphics card resides in the PC memory space in order to remain compatible with most system configurations.

The 128 kb section of memory located at address 80000H–9FFFH is normally reserved for expansion memory, so using this space for our design would preclude adding memory to the system. The 128 kb address range of C0000H through DFFFFH is also available. This section of the memory space is reserved for ROM on I/O adapters, such as our card. Since many commercially available peripheral cards use portions of this address space, we would like to avoid using a large portion of this area in order to remain compatible with them. We will map the 82786 Internal Registers into address C4400H–C447FH.

There is one other section of memory available to us without going into Protected Mode. This is the address range A0000H–BFFFFH, which is reserved for the graphics display adapters. The A000 segment is used by the EGA, whereas the B000 segment is used by the CGA (and MDA). Since we are designing a graphics card, we will use a portion of this memory space. It is desirable to use as large a portion of the PC's memory space as possible in order to reduce the amount of paging required to access graphics memory. Let us choose the 64 kbyte A000 segment. This means that our design will work along with a CGA card in the system, but not with an EGA. This is a reasonable choice since, if people require a higher performance graphics system, the 82786 based design will provide much more power than the EGA. The CGA can still be used for most text and low resolution graphics applications.

The 80286 microprocessor can address a full 64 kbyte I/O space. However, the PC only supports I/O addressing from 000–3FFH, as shown in Figure 2. I/O addresses 000–0FFH are reserved for the system board I/O, leaving addresses 100H–3FFH available on the I/O CHANNEL. A look at the I/O address map will show that most of this space is reserved for various peripheral devices that might be installed in the system. Once again, if I/O addressing is required, we must be careful in choosing which portion of I/O space we use in order to remain compatible with these peripherals.

Address	Name	Function
000000 to 07FFFF	512 kb System Board	System Board Memory
080000 to 09FFFF	128 kb	I/O Channel Memory—IBM Personal Computer AT 128 kb Memory Expansion Option
0A0000 to 0BFFFF	128 kb Video RAM	Reserved for Graphics Display Buffer
0C0000 to 0DFFFF	128 kb I/O Expansion ROM	Reserved for ROM on I/O Adapters
0E0000 to 0EFFFF	64 kb Reserved on System Board	Duplicated Code Assignment at Address FE0000
0F0000 to 0FFFFF	64 kb ROM on the System Board	Duplicated Code Assignment at Address FF0000
100000 to FDFFFF	Maximum Memory 15 Mb	I/O Channel Memory—IBM Personal Computer AT 512 kb Memory Expansion Option
FE0000 to FEFFFF	64 kb Reserved on System Board	Duplicated Code Assignment at Address 0E0000
FF0000 to FFFFFF	64 kb ROM on the System Board	Duplicated Code Assignment at Address 0F0000

Figure 1. IBM AT System Memory Address Map

Hex Range	Device
000–01F	DMA Controller 1, 8237A-5
020–03F	Interrupt Controller 1, 8258A, Master
040–05F	Timer, 8254.2
060–06F	8042 (Keyboard)
070–07F	Real-Time Clock, NMI (Non-Maskable Interrupt) Mask
080–09F	DMA Page Register, 74LS612
0A0–0BF	Interrupt Controller 2, 8259A
0C0–0DF	DMA Controller 2, 8237A-5
0F0	Clear Math Coprocessor Busy
0F1	Reset Math Coprocessor
0F8-0FF	Math Coprocessor
1F0–1F8	Fixed Disk
200–207	Game I/O
278–27F	Parallel Printer Port 2
2F8–2FF	Serial Port 2
300–31F	Prototype Card
360–36F	Reserved
378–37F	Parallel Printer Port 1
380–38F	SDLC, Bisynchronous 2
3A0–3AF	Bisynchronous 1
3B0–3BF	Monochrome Display and Printer Adapter
3C0–3CF	Reserved
3D0–3DF	Color/Graphics Monitor Adapter
3F0–3F7	Diskette Controller
3F8–3FF	Serial Port 1

Figure 2. IBM AT System I/O Address Map

I/O address range 300H–31FH is reserved for prototype cards, so we will use a portion of this space in our design, as will be discussed later. Another possibility would be to use the game controller address range 200H–207H, if it is known that the game controller will not be used.

2.3 Signal Description

Interfacing to the PC is quite simple since all address, data, and control signals are decoded and demultiplexed for us. In addition, wait states can be inserted, in which case these signals are held valid as long as we wish. Wait states must last no longer than 2.5 microseconds (2.1 microseconds for the 8-bit PC), in order to meet IBM specifications.

The signals used in this basic interface are listed in Table 1. Other signals, incorporating other features, could be used, such as interrupt and DMA control lines. We will discuss only the signals used in this design.

Address Latch Enable, BALE, is used on the system board to latch valid addresses. Address lines SA0–SA19 are used to address the memory and I/O devices in the system. They are gated onto the system bus when BALE is high and are latched on the falling edge of BALE.

There is another set of address lines, LA17–LA23, which gives the system up to 16 Mb of addressability. These signals are unlatched and remain valid only as long as BALE is high. They become valid earlier than the SA lines and are intended to generate decodes for memory or I/O cycles. They should be latched by I/O adapters on the falling edge of BALE when needed.

There are 16 data lines, SD0–SD15, which are demultiplexed (the 80286 in IBM AT computer has separate address and data lines) and held valid as long as the system is held in a wait state. 8-bit interfaces will only use SD0–SD7. Data transfers on the upper byte of the data bus are indicated by a low signal on the SBHE pin.

The control signals have been decoded and, like the data lines, are held valid as long as the PC is held in a wait state. $\overline{\text{IOR}}$ and $\overline{\text{IOW}}$ are active low signals that indicate an I/O read and write, respectively. Similarly, $\overline{\text{MEMR}}$ and $\overline{\text{MEMW}}$ indicate a memory read or write bus cycle.

"I/O CH RDY", I/O CHANNEL ready, is pulled low by a peripheral device in order to insert wait states. This signal must be driven low very quickly upon detecting a valid address and a Read or Write command. This timing will be discussed in more detail in a subsequent section. As mentioned earlier, this signal should be held low for no more than 2.5 microseconds.

MEM CS16 is pulled low to signal the system board that the current data transfer is a 1 wait state, 16-bit memory cycle. If this signal is not brought low in time to be recognized by the system, the memory access will automatically be broken into two 8-bit accesses, even if a 16-bit access was desired. In addition, this signal is an input to the CMDLY pin of the 82288 bus controller chip on the system board. It can delay the issuance of the $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{IOR}}$, and $\overline{\text{IOW}}$ signals in order to allow more address setup time. As will be discussed later, this signal should be derived from the decode of LA17 through LA23.

The final signal we have used in this design is RESET DRV. This is the active high power on reset signal.

Table 1. I/O CHANNEL Signal Description

SA0–SA19	Latched Address Lines
LA17–LA23	Unlatched Address Lines Used to Generate Decodes for 1 Wait-State Memory Cycles
RESET DRV	Power On Reset Signal from the PC/AT
SD0–SD15	Latched Data Lines
I/O CH RDY	Ready Signal to Generate PC/AT Wait-States
IOR	Indicates an I/O Read
IOW	Indicates an I/O Write
MEMR	Indicates a Memory Read
MEMW	Indicates a Memory Write
MEMCS16	Signals the AT to Perform a 1 Wait-State 16-Bit Memory Cycle
SBHE	Indicates Data Transfer on Upper Byte of Data Bus

3.0 82786 BUS INTERFACE

3.1 Overview

The Bus Interface Unit (BIU) controls all communication between the 82786, the external bus master, and memory. The 82786 is capable of being a bus master (Master Mode) or a bus slave (Slave Mode). The 82786 operates as a master whenever it accesses external system memory and the bus timings are similar to 80286 style bus timings. It acts as a slave when the host CPU accesses graphics memory or the 82786 registers.

In Master Mode, the 82786 drives the Hold Request (HREQ) pin high to indicate it is requesting the bus. The 82786 drives the external bus only after it receives a Hold Acknowledge (HLDA) from the external bus master and drives HREQ low when it no longer needs the bus or when it detects an inactive HLDA. The 82786 indicates it has the bus through a high level on the Master Enable (MEN) pin.

The state of the $\overline{BHE}$ pin at the trailing edge of RESET determines whether the interface is synchronous or aysnchronous. A high $\overline{BHE}$ sets the 82786 in synchronous operation. In Master Mode, synchronous/asynchronous operation affects the sampling of the HLDA signal only. In Slave Mode, synchronous/asynchronous operation affects the sampling of the $\overline{RD}$ and $\overline{WR}$ signals, as will be seen, and allows for direct connection to an 80286 (80186 and 80386 can also be supported).

Every design must support Slave Mode. The great majority of applications do not require Master Mode and it need not be supported. Our design uses an asynchronous slave interface, which we will focus on in more detail.

3.2 Aysnchronous Slave Interface

The following pins make up the 82786 Slave Interface:

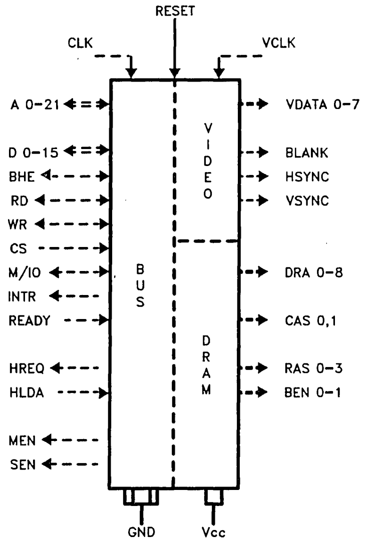
1. 22 address inputs, A21:0.
2. $\overline{BHE}$ input used to indicate valid data on the upper data bus, D8–D15, of the 82786 graphics memory.
3. Bus command input signals $\overline{RD}$, $\overline{WR}$, and M/$\overline{IO}$.
4. Chip select input, $\overline{CS}$.
5. Slave Enable output, SEN, is used to signal the system that the requested slave access is currently being serviced. This signal is used to enable the connection of the 82786 data bus to the external data bus and also as a source of READY to the external master.

All of the input signals, with the exception of $\overline{CS}$, are bidirectional pins driven by the 82786 when it is executing Master Mode cycles. Whenever the 82786 is in Slave Mode, these signals are monitored by the Slave Interface logic. The correct combination of bus commands on these pins generate a slave cycle request.

Figure 4 shows the timing relations for the Asynchronous Slave interface. When either $\overline{RD}$ or $\overline{WR}$ are detected low, $\overline{CS}$ is sampled. If $\overline{CS}$ is found to be low, the 82786 will generate a slave cycle request. Note that the address pins, along with $\overline{BHE}$ and M/$\overline{IO}$ have the same setup and hold timing as $\overline{CS}$. Once the setup and hold times have been met, the valid addresses may be removed since they will have been latched internally by the 82786.

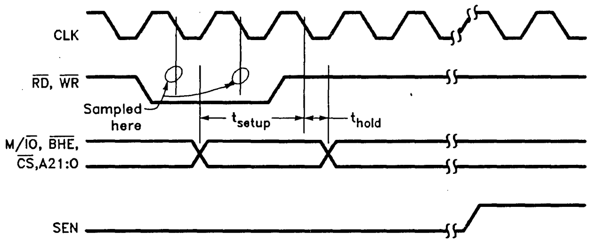
A slave cycle request is arbitrated between DRAM refresh, Display Processor requests, and Graphics Processor requests for bus bandwidth and is serviced according to the programmed priority of each type of request. Notice the break in Figure 4 between the control signals going active and SEN going high, indicating the indeterminate amount of time before the 82786 begins to execute the slave cycle. Even if external slave accesses are programmed to be higher priority than graphics or display processor requests for the bus, DRAM refresh cycles always have highest priority and can occur at any time. This can hold off execution of the slave cycle for a few clocks. Therefore, once the PC makes a slave request to the 82786, it will have to be held in a wait state (by pulling IOCHRDY low) until SEN goes high and the slave cycle begins.

Figure 5 shows the timing relations for the slave cycle during the SEN active high time. SEN remains high for the entire cycle, which lasts four clocks for a write and five clocks for a read.



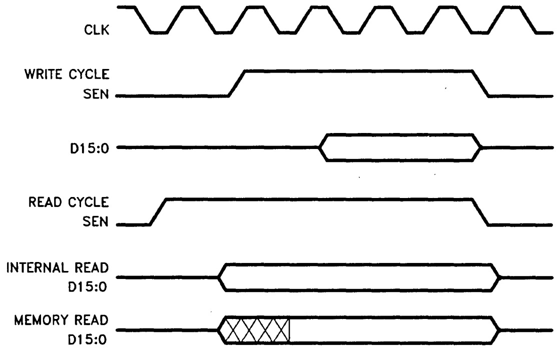
240049-1

Figure 3. 82786 Pins


CLK
$\overline{RD}$, $\overline{WR}$
Sampled here
t_{setup}
t_{hold}
M/$\overline{IO}$, $\overline{BHE}$,
$\overline{CS}$,A21:0
SEN

240049-2

Figure 4. 82786 Asynchronous Slave Interface



240049-3

Figure 5. 82786 SEN/DATA Slave Interface

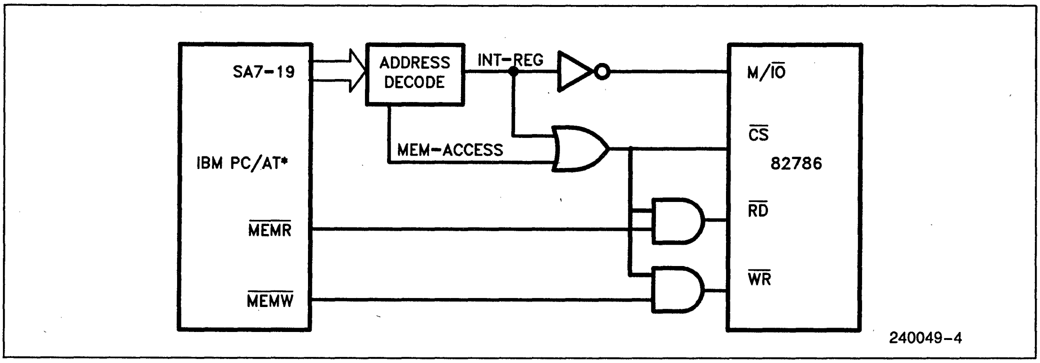


Figure 6. Generating 82786 Control Signals

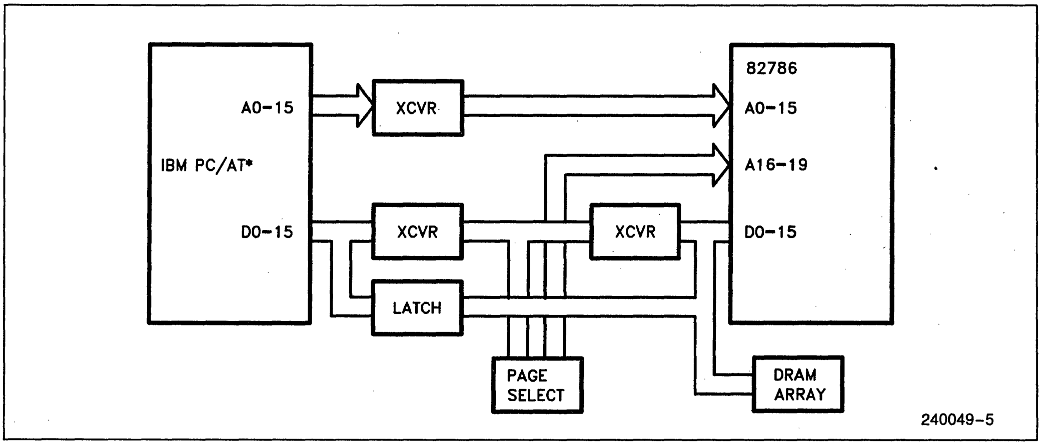


Figure 7. Block Diagram of Data/Address Bus

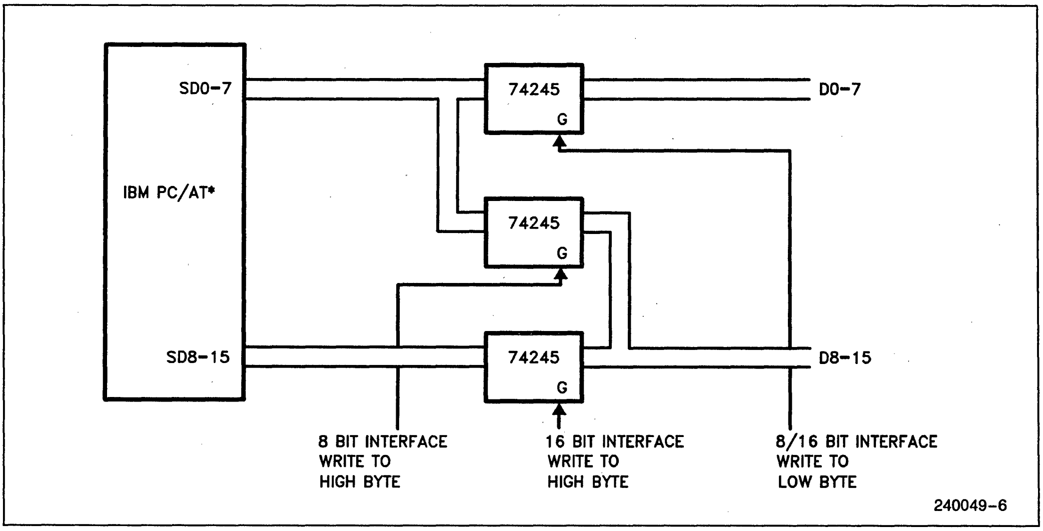

SD0-7
IBM PC/AT*
SD8-15
74245
G
74245
G
74245
G
D0-7
D8-15
8 BIT INTERFACE WRITE TO HIGH BYTE
16 BIT INTERFACE WRITE TO HIGH BYTE
8/16 BIT INTERFACE WRITE TO LOW BYTE
240049-6

Figure 8. 8/16-Bit Crosser for Write Data

4.0 INTERFACING THE 82786 TO THE I/O CHANNEL

4.1 General Considerations

Our graphics board will decode the address, $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{IOR}}$, and $\overline{\text{IOW}}$ signals from the PC and, if a slave cycle request is detected, generate the proper control signals ($\overline{\text{RD}}$, $\overline{\text{WR}}$, M/$\overline{\text{IO}}$, $\overline{\text{BHE}}$ and $\overline{\text{CS}}$) along with gating the address and data to the 82786. Refer to Figure 6 for a block diagram of generating the 82786 control signals. The IOCHRDY signal will immediately be pulled low in order to place the PC into a wait state. If our design is to a 16-bit interface, we must also pull the MEMCS16 signal low. This signal is not used for an 8-bit interface.

Once IOCHRDY has been pulled low, the PC will be held in a wait state. This will cause the demultiplexed address, data, and control signals to be held valid for us. This means that we do not have to latch these signals on our board. Figure 7 shows a block diagram of the address and data bus interface for our design. Once our board is selected, control logic can turn on the address and data transceivers/latches.

4.2 Loading

IBM specifies no more than two TTL loads per pin per slot for the I/O CHANNEL. This is to assure that the PC system board can properly drive all peripheral cards that may be plugged into the I/O CHANNEL. In order to meet this spec, it is necessary to buffer any signals that drive more than two loads in our design.

4.3 Write Data

Figure 7 shows a data path consisting of two levels of transceivers for the write path. Working our way from the I/O CHANNEL side, the first bank of transceivers encountered buffer and gate the data bus onto our board. The second bank of transceivers isolate the 82786 from the board's data bus.

Upon receiving a decode, the first transceivers will turn on, gating data onto the board. The second transceivers only turn on when SEN goes high, indicating the start of the slave cycle. In this way, the PC's data can gate onto the board without interfering with any other 82786 memory cycles in progress to the graphics memory. This allows the PC to access the Page Select register (which will be discussed later) without disturbing 82786 memory cycles. In addition, if we were to insert a dedicated on-board CPU, it would interface between these two sets of transceivers, allowing the PC to talk to the local CPU without disturbing 82786 memory activity.

A crosser network for write data is shown in Figure 8. All three transceivers are needed only if the design will interface to both an 8-bit and a 16-bit system, as in our example. If the design is only to the 8-bit PC or to the IBM AT computer, then only two transceivers are required.

4.4 Read Data

Read data from the 82786 or graphics memory must be latched. This is shown in Figure 5, where it can be seen that read data will only be valid for 3–4 clocks. Depending upon when IOCHRDY is released, read data may come and go before the PC can come out of its wait state (refer to Section 4.5), so it must be latched and held.

The read data latches shown in Figure 9 are configured similarly to the write data transceivers of Figure 8. Once again, all three latches are needed only if the design will interface to both a PC and an IBM AT computer, as in our example.

The data latch control can be implemented by counting 82786 clocks from the rising edge of SEN until read data is valid, as shown in Figure 5, and then latching the data. This data can be held and made available for when the I/O CHANNEL exits its wait state.

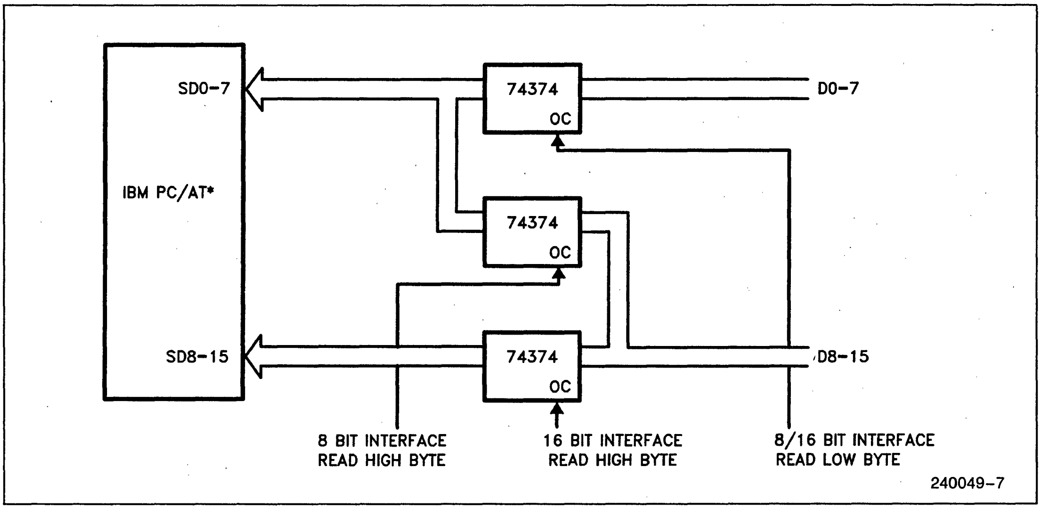


Figure 9. Read Data Latches

4.5 Exiting Wait States

There are many choices of clock speeds to run the 82786 in a graphics design. In addition, our design may be interfaced to several different speed PC's. As a result, once IOCHRDY is released you cannot know exactly when the PC will come out of its wait state in relation to SEN and the slave cycle.

For example, if the PC is writing to our graphics board, write data will be held valid as long as the PC is kept in a wait state. We need the write data to remain valid to meet the data hold time as shown in Figure 5. If IOCHRDY is released at the rising edge of SEN and the PC is running at a fast clock rate, it is possible for the PC to exit its wait state and remove the write data too early in the SEN/DATA cycle. This can, of course, be predicted exactly if all possible combinations of the 82786 and PC clock speeds are known. The exact time to release IOCHRDY in order to hold data long enough, yet not insert extra wait states, can be calculated.

A more conservative approach is to release IOCHRDY off the falling edge of SEN, which has been done in this design. For write cycles, this guarantees that write data will be held past the entire SEN/DATA cycle. Since we are latching read data, it is held valid for whenever the wait state ends. The tradeoff here is that performance will be degraded somewhat since extra clocks are inserted into every slave access.

4.6 Page Selection

As previously mentioned, our design uses the 64 kbyte section of the PC's memory space located at the A000 segment. Our graphics board will contain 1 Mbyte of memory, however. We must have some method of accessing the entire 1 Mbyte of graphics memory from the PC's 64 kbyte window. This is accomplished through the use of a simple paging scheme, as shown in Figure 10.

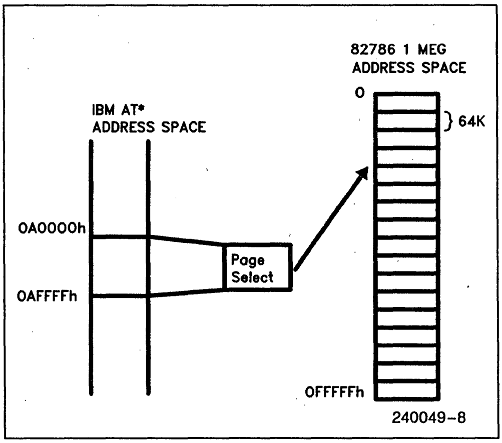


Figure 10. Page Selection

Pins SA0–SA15 directly drive 82786 pins A0–A15 to address within a given 64 kbyte section of memory. 82786 pins A16–A19 determine which 64 kbyte memory section, or page, we address. We can drive these page selection address lines with a latch which will contain the desire page number from 0 to FH. Figure 11 shows a block diagram of the page select circuit.

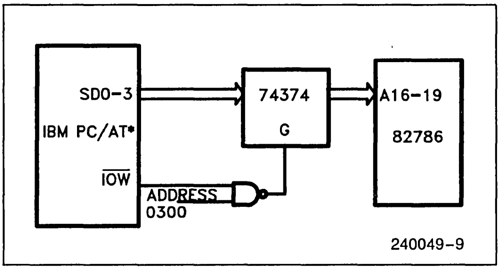


Figure 11. Page Select Circuit

Our design latches the page number from data bits D0–D3 by writing to I/O address 0300H. The output of the latch will then drive 82786 pins A16–A19.

5.0 SPECIAL CONSIDERATIONS

5.1 IOCHRDY and MEMCS16 Timing

For 16-bit accesses, IOCHRDY and MEMCS16 must be brought low very quickly upon decoding an access to the peripheral board. The purpose of signals LA17–LA23 is to provide address lines for such decodes. These address lines become valid at the I/O CHANNEL earlier than the latched address lines SA0–SA19, providing more setup time for the decode and generation of MEMCS16. For this reason it is desirable to use the LA lines for decodes whenever possible.

Lines LA17–LA23 provide for address decoding down to 128 kbyte resolution. Decoding addresses in 64 kbyte sections would require an LA16 pin, which is not provided. Recall, however, that we were unable to find a convenient 128 kbyte section of the IBM system memory space to use for our design. This means that we must use address pin SA16 for part of our decode. Since the SA lines become valid later than the LA lines, we have less time in which to decode an address and generate a MEMCS16.

Although both IOCHRDY and MEMCS16 must be brought low quickly, MEMCS16 is the most critical timing of the two. For an 8 MHz PC, we have 24 ns from SA0–SA16 valid to issue a MEMCS16 low signal in order to cause a 16-bit access and −11 ns to cause a command delay. Fortunately we do not care about the command delay. If we miss the window, any 16-bit accesses will automatically be broken into two 8-bit accesses.

The circuit used to generate IOCHRDY and MEMCS16 can be found in the complete board schematic in the Appendix. Figure 12 focuses on the particular circuitry of interest here. Fast logic devices are required and the decode PAL must have a short propagation delay. This is particularly important since an extra flip-flop is used to cause IOCHRDY to be released from the falling edge of SEN.

5.2 Maximum Wait States

For medium and high resolution displays using DRAM's, a significant portion of the available memory bandwidth is required for the display process. To accomodate this, the display processor is programmed for highest priority access into the graphics memory. In the worst case the display processor can stay on the memory bus continuously for more than 5 microseconds. This will happen while fetching 16 tile descriptors (96 words).

The IBM AT Technical Reference Manual warns us not to hold IOCHRDY low for more than 2.5 microseconds, which is 15 wait states in a 6 MHz machine and 20 wait states in an 8 MHz machine. During wait states the DMA controller can't obtain the bus to refresh memory, which results in delayed memory refresh cycles.

If the 82786 is programmed with highest priority for the display processor, then the external CPU will have to stay in a wait state until the display processor releases the memory bus and SEN goes high. Although this may not result in any erroneous operation, it will certainly violate the bus spec for wait state duration. In order to meet the bus spec, the 82786 must be programmed with the highest priority for the external CPU.

This will be acceptable if the CPU accesses the 82786 and graphics memory only during noncritical periods. However, if the CPU needs to access the 82786 very frequently, then the display processor may not be able to refresh the screen in the required amount of time. Therefore, if the CPU is programmed to highest priority, it must be restricted in its access to the 82786.

The latency between successive accesses from the CPU to the 82786 can be approximated by the expression.

latency (in 82786 system clock cycles) ≥
2 * [8/((MMXR/DPXR) − 1)]

where: MMXR = maximum memory data transfer rate (40 Mbytes/s if using interleaved, fast page mode DRAM's at maximum 82786 bus clock speed)

DPXR = ((Xmax * Ymax * Bpp)/8) * Display refresh rate

Xmax = no. of pixels in the x direction

Ymax = no. of pixels in the y direction

Bpp = no. of bits/pixel

NOTE:
"System clock cycles" refers to internal 82786 clock cycles. 2 pin clocks = 1 system clock. For example, 20 MHz pin clock is equivalent to an 82786 10 MHz system clock rate.

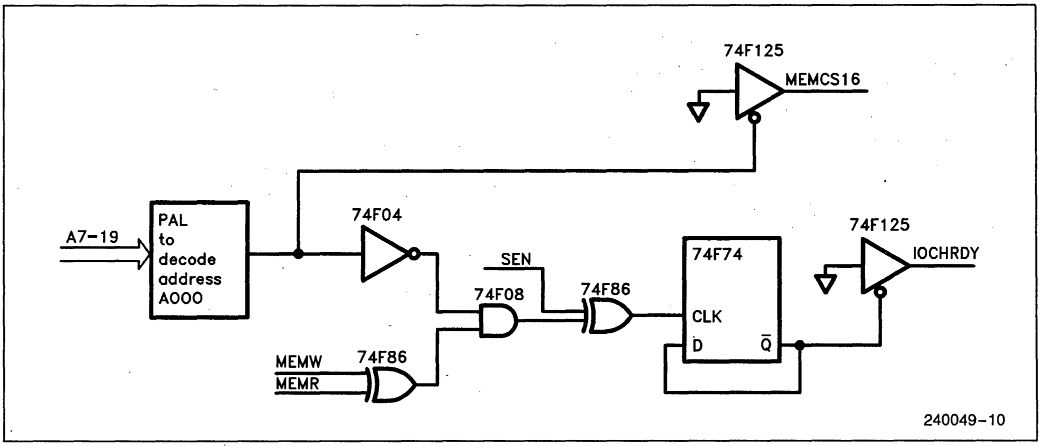


Figure 12. IOCHRDY and MEMCS16 Circuitry

One way to force this restriction on the CPU is to ensure that this latency is built into the software. This implies that:

1) there are no block accesses into the graphics subsystem and

2) all single accesses are separated by instructions (e.g. NOP) that will guarantee that the CPU stays away from the 82786 for the desired time.

This software method to control the CPU accesses will allow single accesses into the graphics memory to be serviced quickly, but at the same time, it imposes a lot of restrictions on the programmer. Additionally, the software becomes very specific to the hardware environment and portability of the software becomes limited. An alternative is to design the hardware to restrict CPU access to the 82786.

The scheme shown in Figure 13 delays the $\overline{CS}$ input into the 82786 by some delay time. When the CPU begins a memory cycle to the 82786, the address decode logic pulls IOCHRDY low, putting the PC into a wait state. The 82786 does not see this request immediately and so the SEN output stays low, which will cause the PC to remain in the wait state. Therefore, the display processor, which is programmed for second priority, can use the memory bus. After the delay time, the 82786 sees the $\overline{CS}$ from the external CPU and services it immediately since the request comes from a higher priority source. The CPU does not stay in a wait state for more than the maximum specified period.

This hardware method of restricting CPU accesses effectively increases the priority of the display processor even though the external CPU is programmed for highest priority. At the same time, this method ensures adequate bus sharing between the display processor and the external CPU.

A feature has been added to the D-stepping of the 82786 which allows for altered priority for external CPU slave requests. The CPU will assume this altered priority every 42 CLK's and will switch back to the standard priority only upon execution of the slave CPU bus cycle.

By programming the CPU to have a standard priority below that of the display processsor and an altered priority higher than the display processor, the previously discussed software and hardware methods of restricting CPU slave accesses to the 82786 are not necessary. This new feature will allow the CPU to get a bus cycle every 2.1 microseconds (assuming an 82786 pin clock rate of 20 MHz) and still give the display processor highest priority the rest of the time.

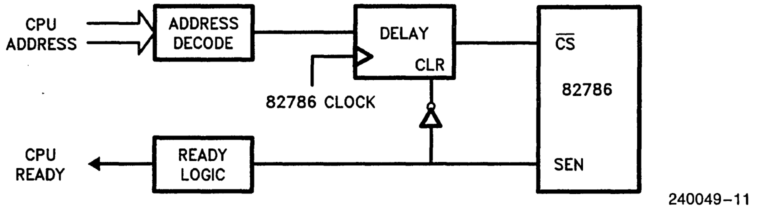


Figure 13. Limiting the Rate of CPU Accesses to the 82786

APPENDIX A
82786 GRAPHICS BOARD DESCRIPTION

The 82786 based graphics board presented in this application note contains 1 Mbyte of fast page mode DRAM's, which comprise the graphics memory for the board. The memory is visible to the PC at address space A0000H–AFFFFH, which is a 64 kbyte section of memory. The entire 1 Mbyte of graphics memory is addressable as 16 64 kbyte banks, which are selected by 82786 address bits A16–A19. The desired bank is selected by writing the value on the lower four bits of the data bus (D0–D3) to I/O address 0300H.

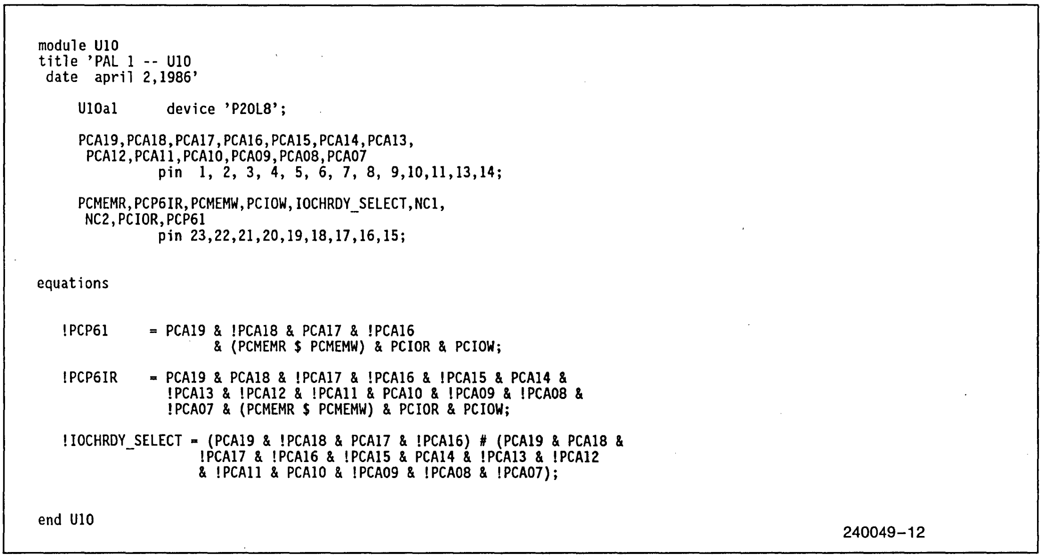
There is one other section of the PC address space that can access the board. That is address C4400H–C447FH. Accesses to this section of memory will cause I/O accesses to be seen by the 82786. This allows access to the Internal Registers, which reside in I/O space when the 82786 comes out of RESET. The Internal Registers remain I/O mapped and must be relocated to I/O space base address 4400H (I/O space for the 82786 is only 64 kbytes—the upper address bits are ignored. However, the upper 7 bits of the Internal Relocation register must be programmed to 0's when locating the Internal Registers in I/O space).

The board supports a software reset. This is accomplished by writing a "1" on data bit 4 to I/O address 0300H and then a "0" on the same data bit to the same address. When using software reset, care must be taken to assure the proper values appear on data bits D0–D3 in order to not reprogram the page select register.

The board contains several jumpers which are described here:

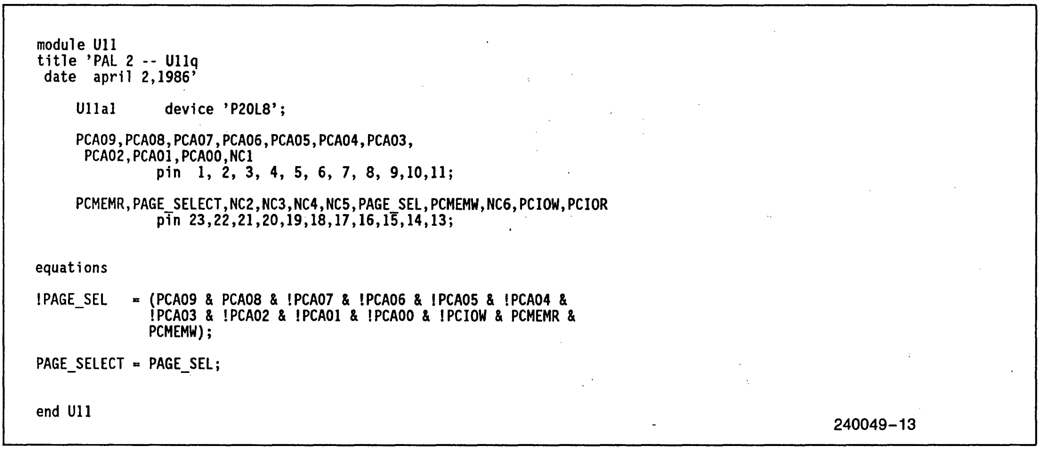
1) There is a jumper to select a 16-bit vs. 8-bit interface. 16-bit interface is selected by inserting the jumper.
2) There are two jumpers to select between synchronous and asynchronous VCLK operation. In synchronous operation, VCLK is tied to CLK on the 82786. In asynchronous operation, VCLK comes from a separate clock oscillator source. Only one of these jumpers may be inserted at any one time.

APPENDIX B
82786 GRAPHICS BOARD PAL EQUATIONS



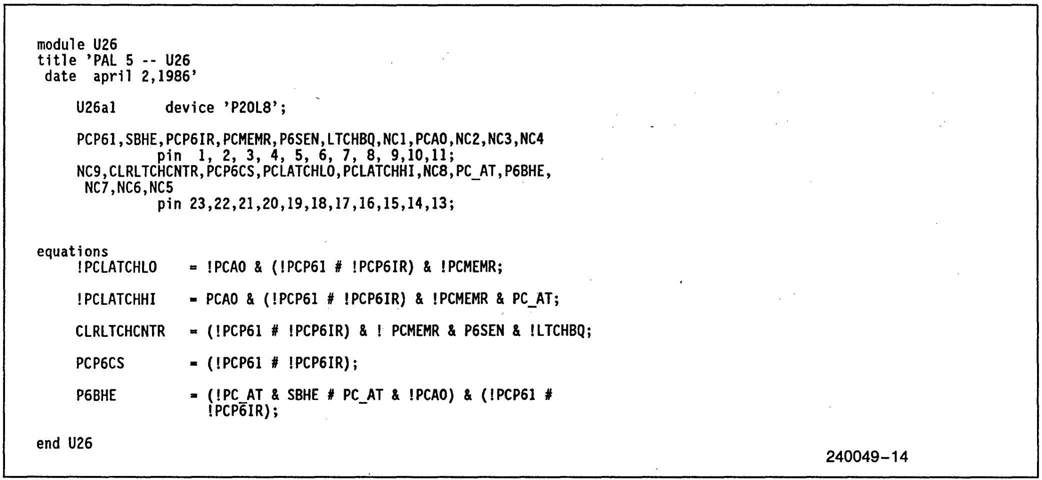
```
module U10
title 'PAL 1 -- U10
 date  april 2,1986'

     U10al      device 'P20L8';

     PCA19,PCA18,PCA17,PCA16,PCA15,PCA14,PCA13,
      PCA12,PCA11,PCA10,PCA09,PCA08,PCA07
               pin  1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,13,14;

     PCMEMR,PCP6IR,PCMEMW,PCIOW,IOCHRDY_SELECT,NC1,
      NC2,PCIOR,PCP61
               pin 23,22,21,20,19,18,17,16,15;


equations


   !PCP61      = PCA19 & !PCA18 & PCA17 & !PCA16
                       & (PCMEMR $ PCMEMW) & PCIOR & PCIOW;

   !PCP6IR     = PCA19 & PCA18 & !PCA17 & !PCA16 & !PCA15 & PCA14 &
                 !PCA13 & !PCA12 & !PCA11 & PCA10 & !PCA09 & !PCA08 &
                 !PCA07 & (PCMEMR $ PCMEMW) & PCIOR & PCIOW;

   !IOCHRDY_SELECT = (PCA19 & !PCA18 & PCA17 & !PCA16) # (PCA19 & PCA18 &
                     !PCA17 & !PCA16 & !PCA15 & PCA14 & !PCA13 & !PCA12
                     & !PCA11 & PCA10 & !PCA09 & !PCA08 & !PCA07);


end U10
```

240049-12



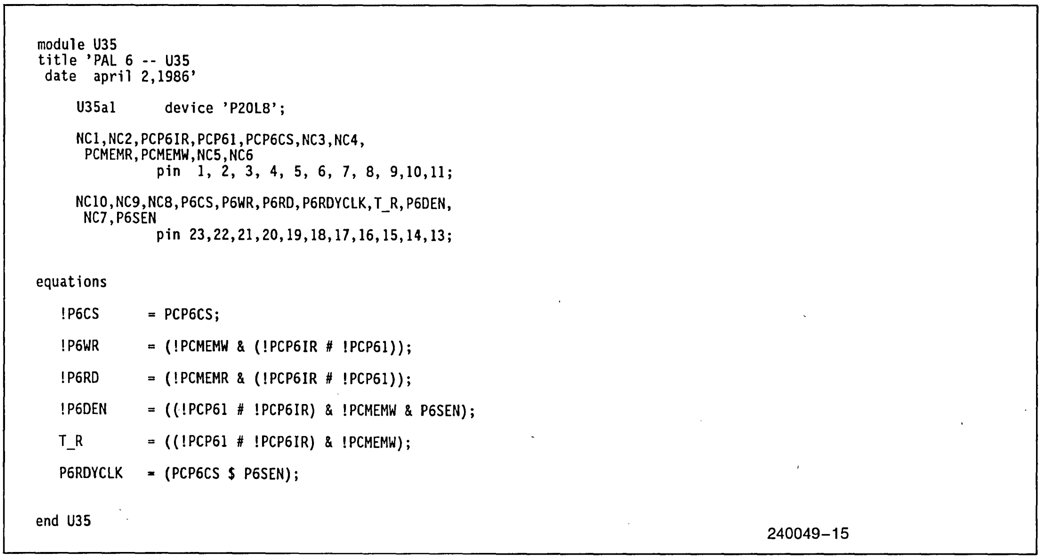
```
module U11
title 'PAL 2 -- U11q
 date  april 2,1986'

     U11a1      device 'P20L8';

     PCA09,PCA08,PCA07,PCA06,PCA05,PCA04,PCA03,
      PCA02,PCA01,PCA00,NC1
               pin  1, 2, 3, 4, 5, 6, 7, 8, 9,10,11;

     PCMEMR,PAGE_SELECT,NC2,NC3,NC4,NC5,PAGE_SEL,PCMEMW,NC6,PCIOW,PCIOR
               pin 23,22,21,20,19,18,17,16,15,14,13;


equations

!PAGE_SEL   = (PCA09 & PCA08 & !PCA07 & !PCA06 & !PCA05 & !PCA04 &
               !PCA03 & !PCA02 & !PCA01 & !PCA00 & !PCIOW & PCMEMR &
               PCMEMW);

PAGE_SELECT = PAGE_SEL;


end U11
```

240049-13



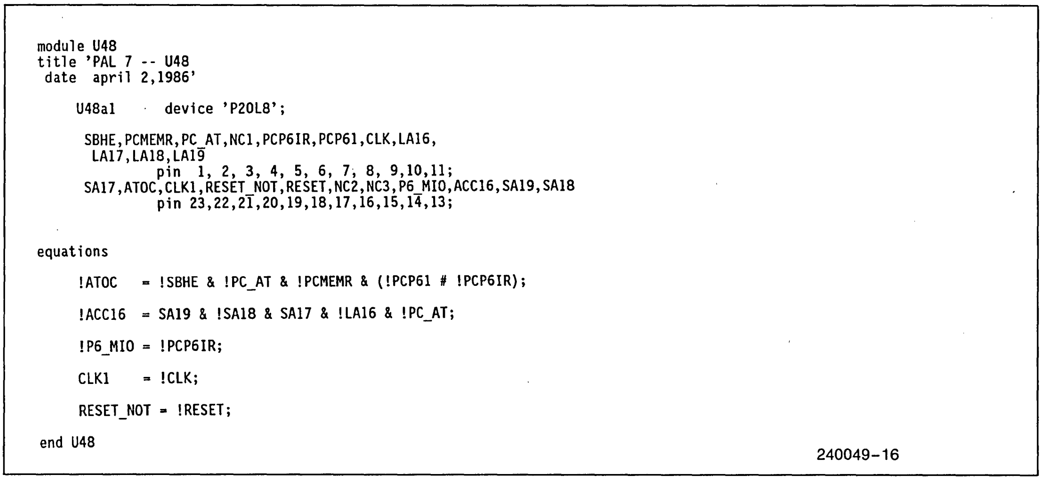
```
module U26
title 'PAL 5 -- U26
 date  april 2,1986'

     U26a1      device 'P20L8';

     PCP61,SBHE,PCP6IR,PCMEMR,P6SEN,LTCHBQ,NC1,PCA0,NC2,NC3,NC4
               pin  1, 2, 3, 4, 5, 6, 7, 8, 9,10,11;
     NC9,CLRLTCHCNTR,PCP6CS,PCLATCHLO,PCLATCHHI,NC8,PC_AT,P6BHE,
      NC7,NC6,NC5
               pin 23,22,21,20,19,18,17,16,15,14,13;


equations
     !PCLATCHLO    = !PCA0 & (!PCP61 # !PCP6IR) & !PCMEMR;

     !PCLATCHHI    = PCA0 & (!PCP61 # !PCP6IR) & !PCMEMR & PC_AT;

     CLRLTCHCNTR   = (!PCP61 # !PCP6IR) & ! PCMEMR & P6SEN & !LTCHBQ;

     PCP6CS        = (!PCP61 # !PCP6IR);

     P6BHE         = (!PC_AT & SBHE # PC_AT & !PCA0) & (!PCP61 #
                     !PCP6IR);

end U26
```

240049-14



```
module U35
title 'PAL 6 -- U35
 date  april 2,1986'

    U35a1      device 'P20L8';

    NC1,NC2,PCP6IR,PCP61,PCP6CS,NC3,NC4,
     PCMEMR,PCMEMW,NC5,NC6
             pin  1, 2, 3, 4, 5, 6, 7, 8, 9,10,11;

    NC10,NC9,NC8,P6CS,P6WR,P6RD,P6RDYCLK,T_R,P6DEN,
     NC7,P6SEN
             pin 23,22,21,20,19,18,17,16,15,14,13;


equations

   !P6CS      = PCP6CS;

   !P6WR      = (!PCMEMW & (!PCP6IR # !PCP61));

   !P6RD      = (!PCMEMR & (!PCP6IR # !PCP61));

   !P6DEN     = ((!PCP61 # !PCP6IR) & !PCMEMW & P6SEN);

   T_R        = ((!PCP61 # !PCP6IR) & !PCMEMW);

   P6RDYCLK   = (PCP6CS $ P6SEN);


end U35
```

240049–15



```
module U48
title 'PAL 7 -- U48
 date  april 2,1986'

    U48a1      device 'P20L8';

     SBHE,PCMEMR,PC_AT,NC1,PCP6IR,PCP61,CLK,LA16,
      LA17,LA18,LA19
             pin  1, 2, 3, 4, 5, 6, 7, 8, 9,10,11;
     SA17,ATOC,CLK1,RESET_NOT,RESET,NC2,NC3,P6_MIO,ACC16,SA19,SA18
             pin 23,22,21,20,19,18,17,16,15,14,13;


equations

    !ATOC   = !SBHE & !PC_AT & !PCMEMR & (!PCP61 # !PCP6IR);

    !ACC16  = SA19 & !SA18 & SA17 & !LA16 & !PC_AT;

    !P6_MIO = !PCP6IR;

    CLK1    = !CLK;

    RESET_NOT = !RESET;

end U48
```

240049–16

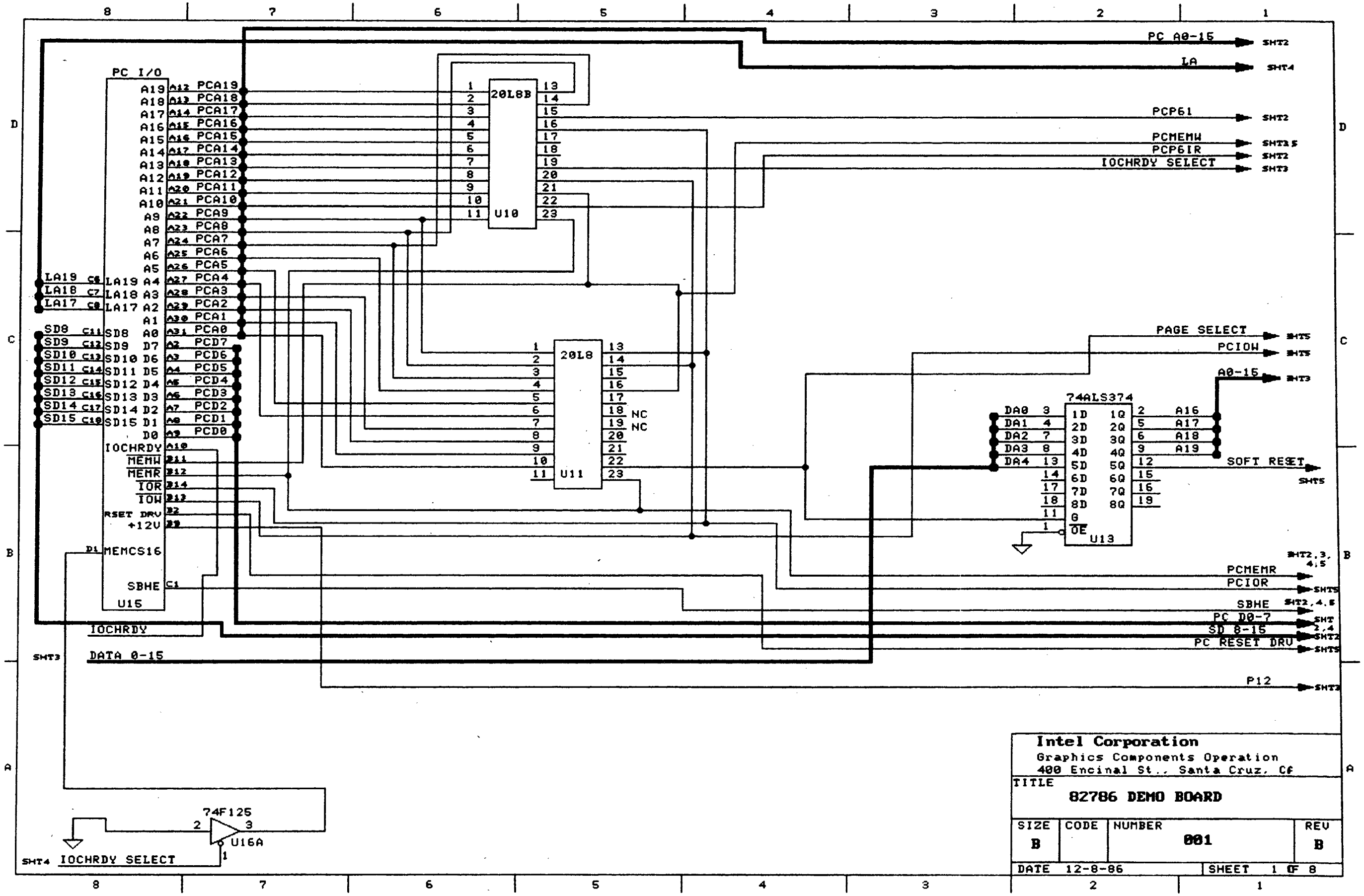
```
module U52
title 'PAL 8 -- U52
 date  april 2,1986'

     U52al      device 'P20L8';

     NC1,PAGE_SELECT,NC2,P6SEN,PCP6IR,PCP61,PCMEMW,
      PCA0,PCIOW,PCMEMR,PCIOR
               pin  1, 2, 3, 4, 5, 6, 7, 8, 9,10,11;

     MANURESET,ADDR_ENABLE,NC5,PCDLO,RESET,IOCHRDY,NC4,
      NC3,_245T_R1,PCRESETDRV,SOFT_RESET
               pin 23,22,21,20,19,18,17,16,15,14,13;


equations

   _245T_R1    = (PCMEMR & PCIOR) # (!PCMEMW # !PCIOW);

   !PCDLO      = ((!PCP61 # !PCP6IR) & !PCA0 & (!PCMEMW # !PCIOW))
                 # !PAGE_SELECT;

   RESET       = (MANURESET # PCRESETDRV # SOFT_RESET);

   !IOCHRDY    = (!PCP61 # !PCP6IR) & !P6SEN;

   !ADDR_ENABLE = (!PCP61 # !PCP6IR);

end U52
```

240049–17

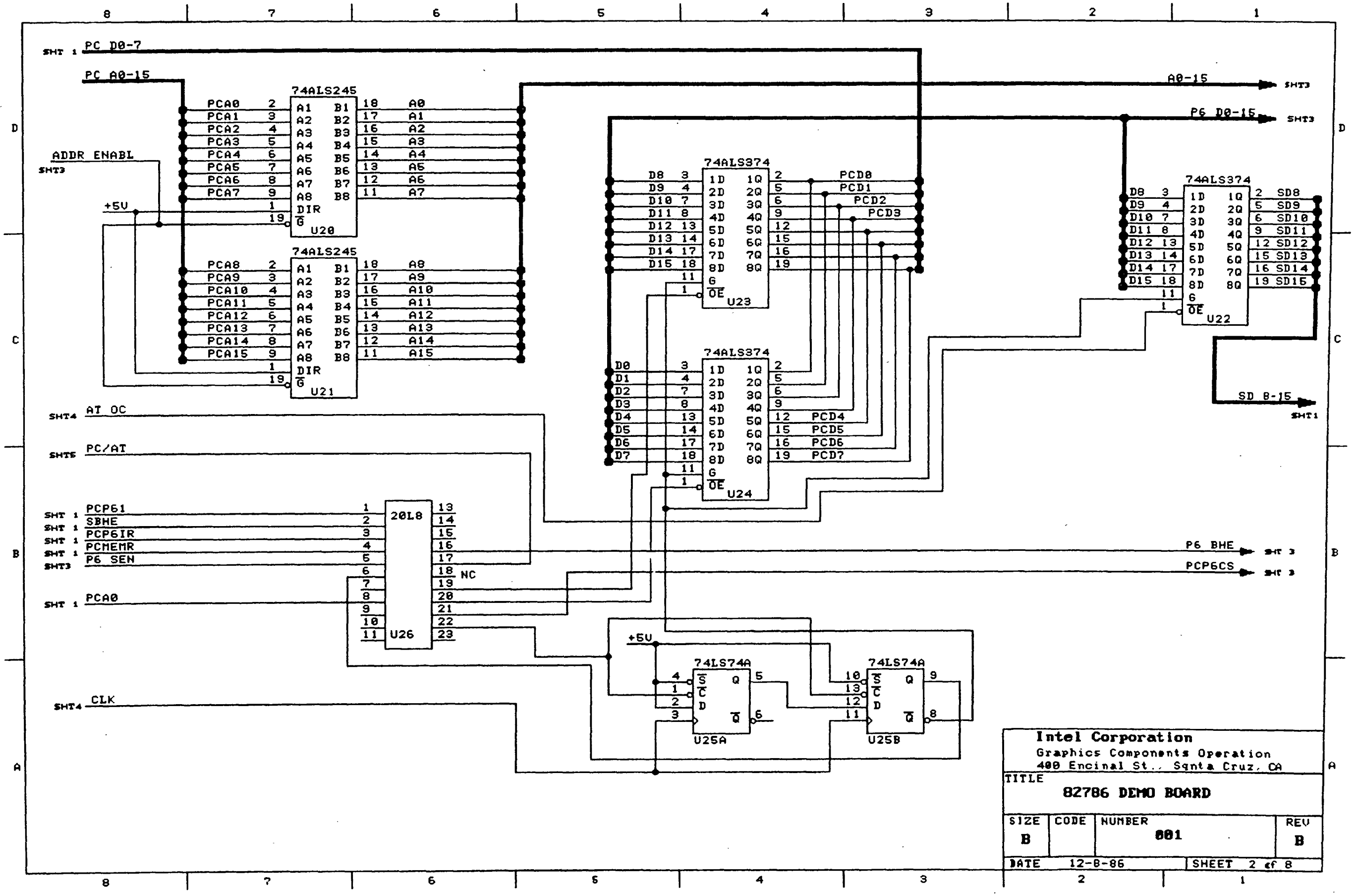
```
module U54
title 'PAL 9 -- U54
 date  april 2,1986'

     U54al      device 'P20L8';

     PCP61,PCMEMR,PCP6IR,NC1,NC2,NC3,PCA0,
      PCMEMW,NC4,P6SEN,SBHE
               pin  1, 2, 3, 4, 5, 6, 7, 8, 9,10,11;
     PCIOR,NC10,NC9,NC8,PCD_AT,PCD_HI,
      NC7,NC6,NC5,PC_AT,PCIOW
               pin 23,22,21,20,19,18,17,16,15,14,13;


equations

     !PCD_HI = (!PCP61 # !PCP6IR) & PCA0 & PC_AT & PCMEMR;

     !PCD_AT = (!PCP61 # !PCP6IR) & !PC_AT & !SBHE & PCMEMR;

end U54
```

240049–18

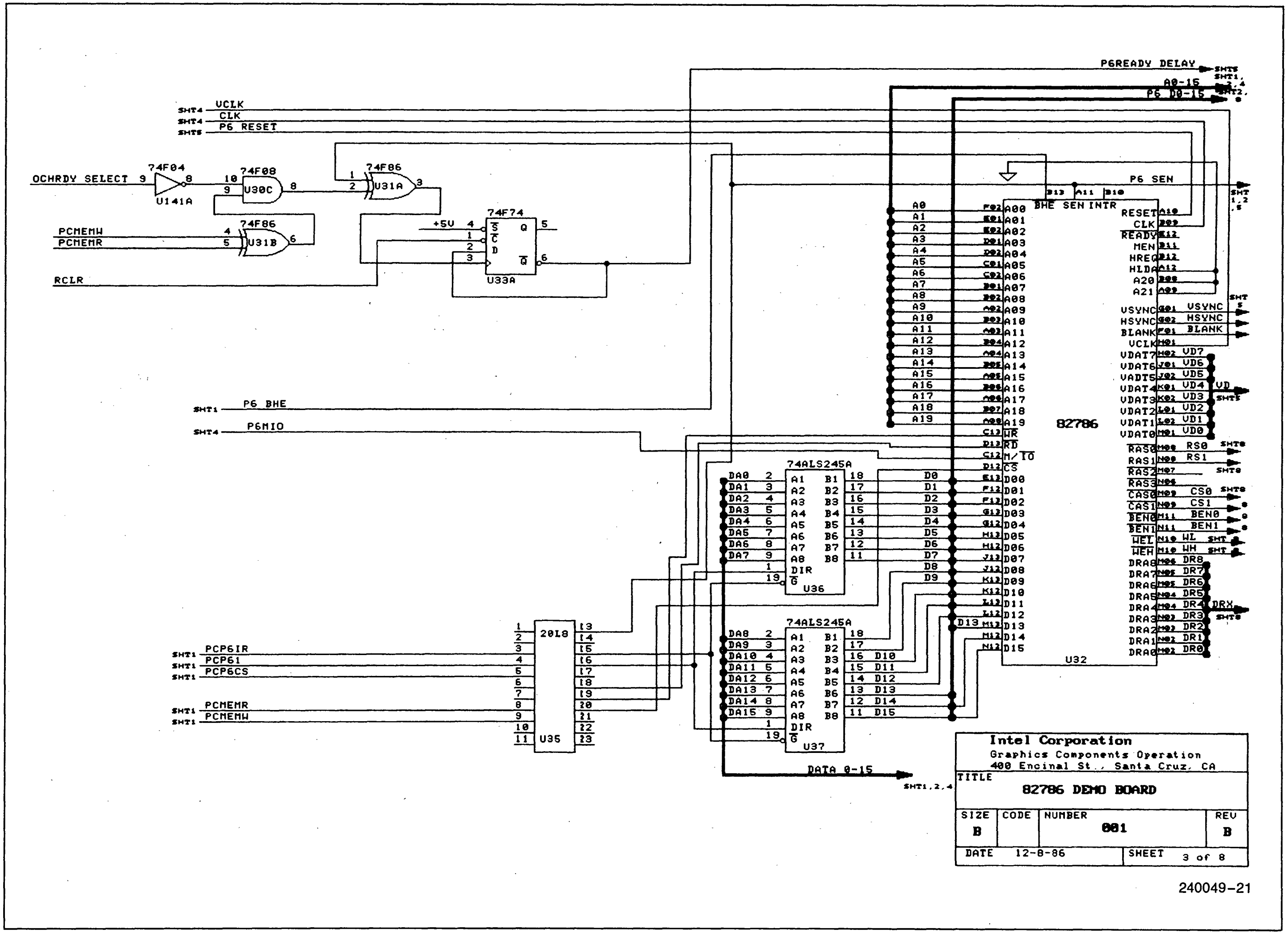
APPENDIX C
82786 GRAPHICS BOARD SCHEMATICS


PC A0-15
SHT2
LA
SHT4
PCP61
PCMEMW
SHT3,5
PCP6IR
IOCHRDY SELECT
SHT3
PC I/O
A19 A18 A17 A16 A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
PCA19 PCA18 PCA17 PCA16 PCA15 PCA14 PCA13 PCA12 PCA11 PCA10 PCA9 PCA8 PCA7 PCA6 PCA5 PCA4 PCA3 PCA2 PCA1 PCA0
LA19 LA18 LA17
SD8 SD9 SD10 SD11 SD12 SD13 SD14 SD15
D7 D6 D5 D4 D3 D2 D1 D0
PCD7 PCD6 PCD5 PCD4 PCD3 PCD2 PCD1 PCD0
IOCHRDY
MEMW
MEMR
IOR
IOW
RSET DRV
+12V
MEMCS16
SBHE
U15
20L8B
U10
20L8
U11
NC
NC
PAGE SELECT
PCIOW
A0-15
SHT5
74ALS374
DA0 DA1 DA2 DA3 DA4
1D 2D 3D 4D 5D 6D 7D 8D G OE
1Q 2Q 3Q 4Q 5Q 6Q 7Q 8Q
A16 A17 A18 A19
SOFT RESET
U13
PCMEMR
SHT2,3,4,5
PCIOR
SBHE
SHT2,4,5
PC D0-7
SD 8-15
PC RESET DRV
IOCHRDY
DATA 0-15
P12
74F125
U16A
IOCHRDY SELECT
Intel Corporation
Graphics Components Operation
400 Encinal St., Santa Cruz, CA
TITLE
82786 DEMO BOARD
SIZE B
CODE
NUMBER 001
REV B
DATE 12-8-86
SHEET 1 OF 8

240049-19


PC D0-7
PC A0-15
SHT 1
74ALS245
U20
U21
PCA0
PCA1
PCA2
PCA3
PCA4
PCA5
PCA6
PCA7
PCA8
PCA9
PCA10
PCA11
PCA12
PCA13
PCA14
PCA15
A0-15
SHT3
P6 D0-15
ADDR ENABL
+5V
DIR
74ALS374
U23
U24
U22
PCD0
PCD1
PCD2
PCD3
PCD4
PCD5
PCD6
PCD7
SD8
SD9
SD10
SD11
SD12
SD13
SD14
SD15
SD 8-15
SHT4
AT OC
SHT5
PC/AT
PCP61
SBHE
PCP6IR
PCMEMR
P6 SEN
PCA0
20L8
U26
NC
P6 BHE
PCP6CS
SHT 3
74LS74A
U25A
U25B
CLK
Intel Corporation
Graphics Components Operation
400 Encinal St., Santa Cruz, CA
TITLE
82786 DEMO BOARD
SIZE
B
CODE
NUMBER
001
REV
B
DATE
12-8-86
SHEET 2 of 8

240049-20



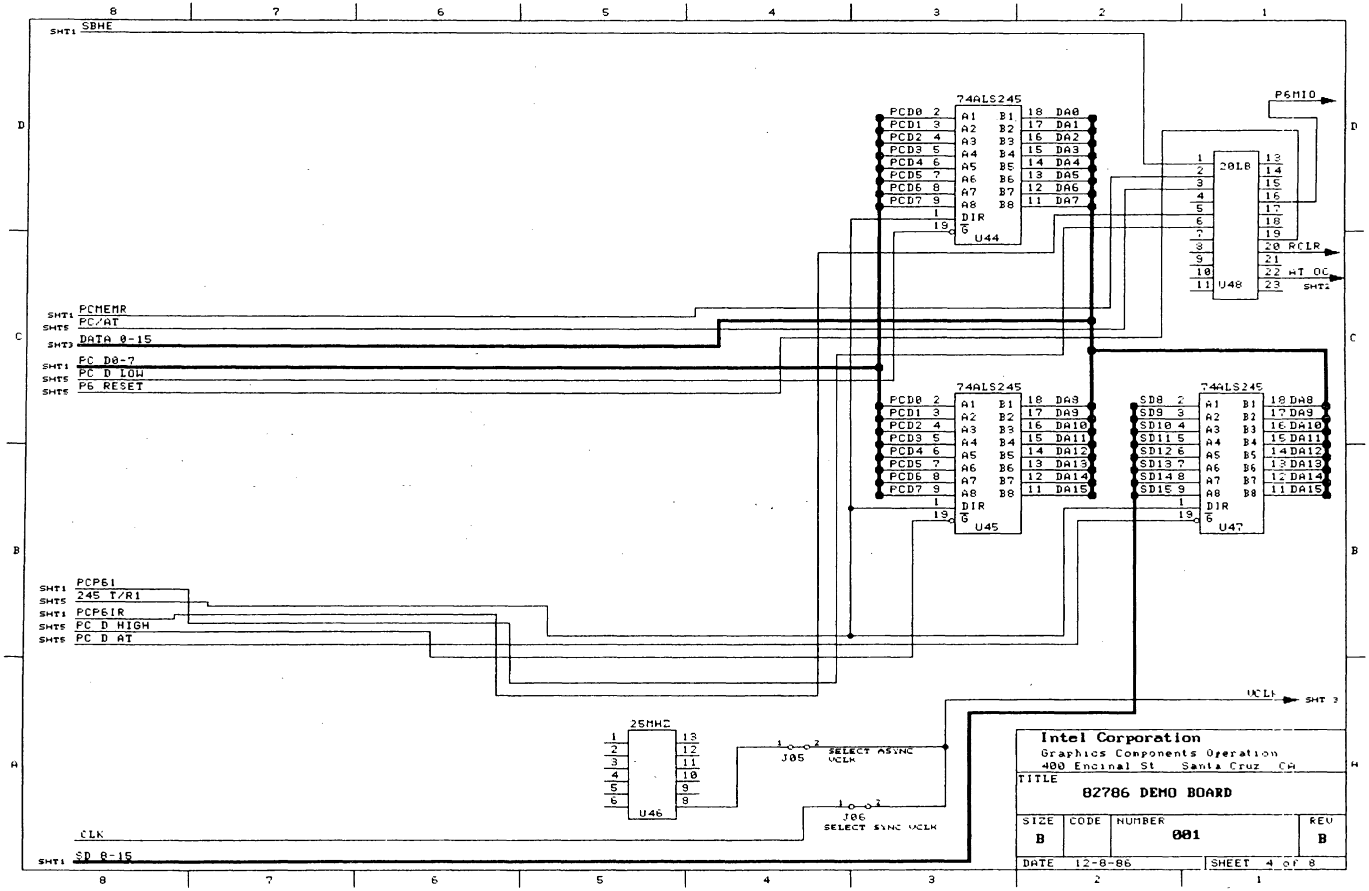
Intel Corporation
Graphics Components Operation
400 Encinal St., Santa Cruz, CA

TITLE
82786 DEMO BOARD

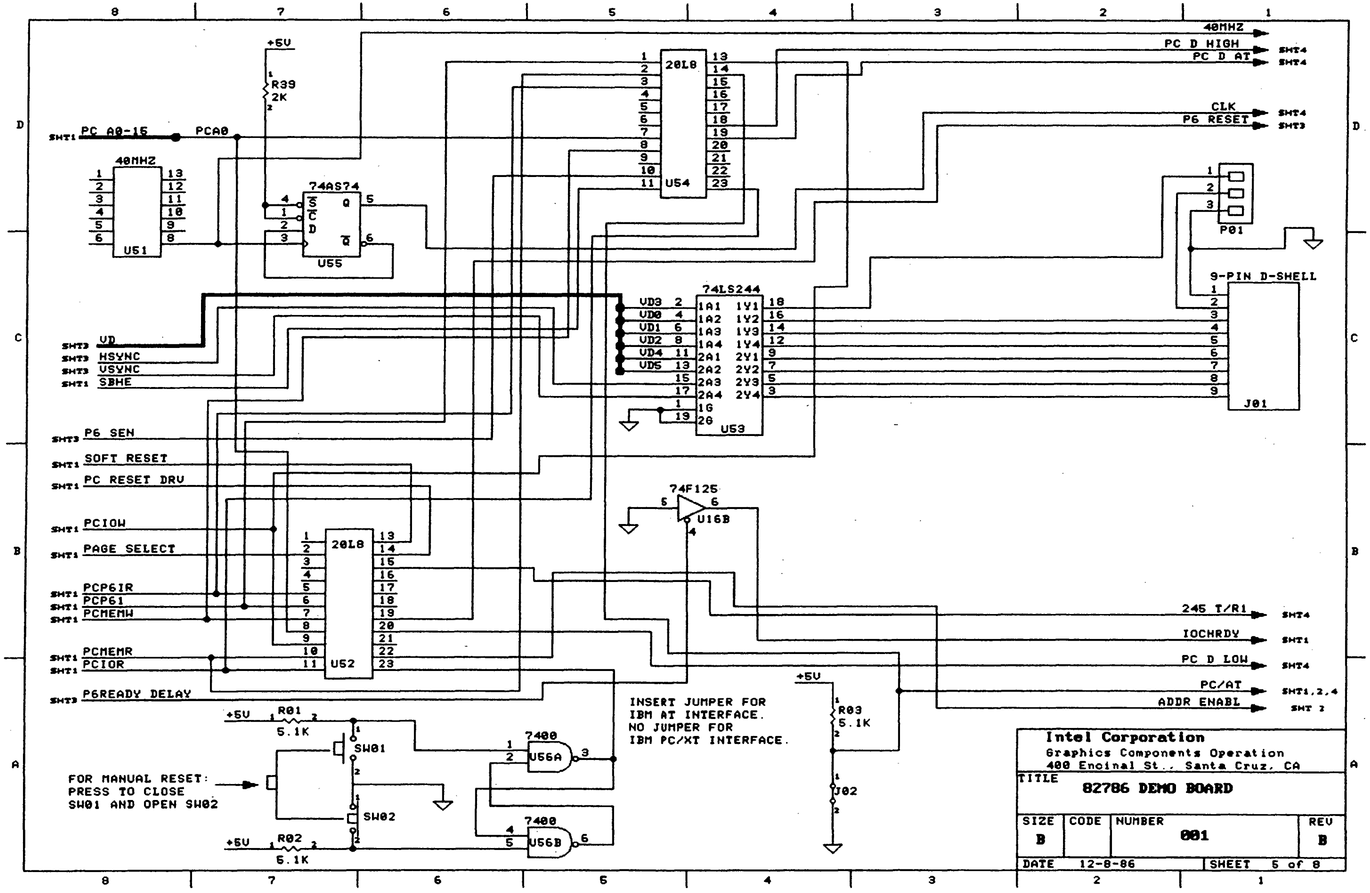
SIZE	CODE	NUMBER	REV
B		001	B

DATE 12-8-86 SHEET 3 of 8

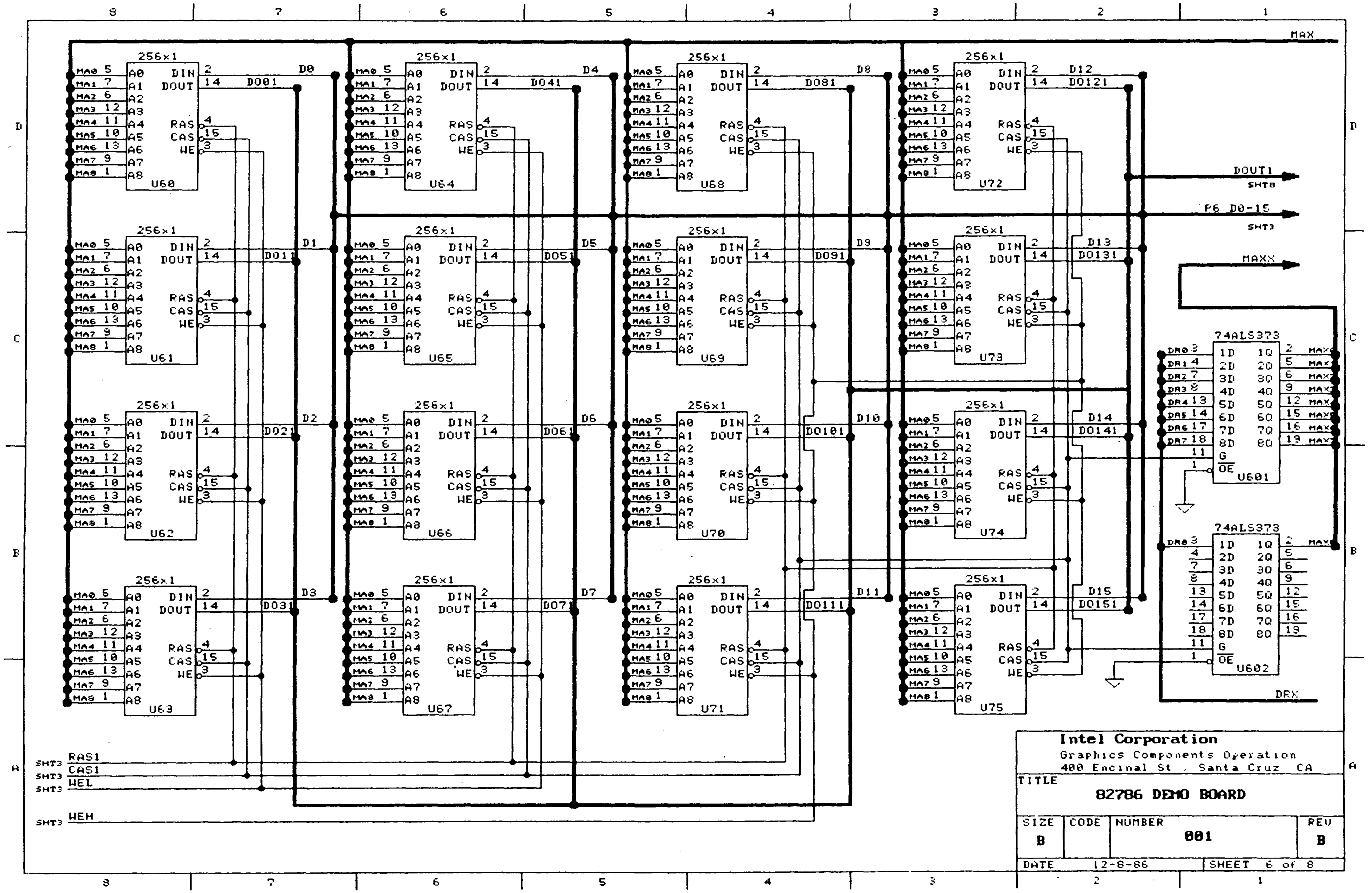
240049-21


SHT1 SBHE
74ALS245
PCD0 2
PCD1 3
PCD2 4
PCD3 5
PCD4 6
PCD5 7
PCD6 8
PCD7 9
A1 A2 A3 A4 A5 A6 A7 A8
B1 B2 B3 B4 B5 B6 B7 B8
DIR
G
18 DA0
17 DA1
16 DA2
15 DA3
14 DA4
13 DA5
12 DA6
11 DA7
U44
P6MIO
20L8
U48
20 RCLR
22 AT OC
SHT2
SHT1 PCMEMR
SHT5 PC/AT
SHT3 DATA 0-15
SHT1 PC D0-7
SHT5 PC D LOW
SHT5 P6 RESET
18 DA8
17 DA9
16 DA10
15 DA11
14 DA12
13 DA13
12 DA14
11 DA15
U45
SD8 2
SD9 3
SD10 4
SD11 5
SD12 6
SD13 7
SD14 8
SD15 9
U47
SHT1 PCP61
SHT5 245 T/R1
SHT1 PCP6IR
SHT5 PC D HIGH
SHT5 PC D AT
VCLK SHT 3
25MHZ
U46
J05 SELECT ASYNC VCLK
J06 SELECT SYNC VCLK
CLK
SHT1 SD 8-15
Intel Corporation
Graphics Components Operation
400 Encinal St Santa Cruz CA
TITLE 82786 DEMO BOARD
SIZE B
CODE
NUMBER 001
REV B
DATE 12-8-86
SHEET 4 of 8
8 7 6 5 4 3 2 1
D C B A

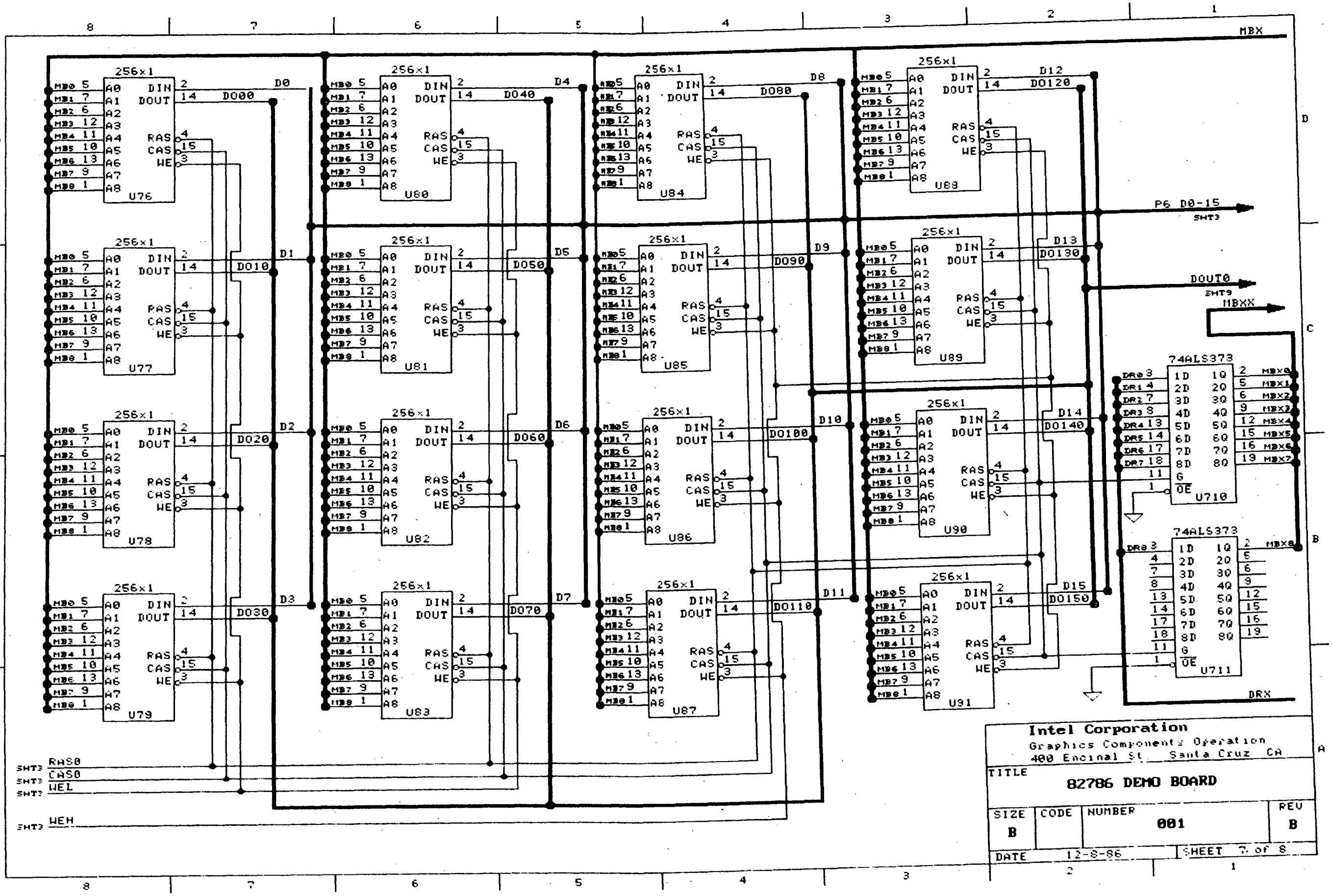
240049-22


40MHZ
PC D HIGH
PC D AT
CLK
P6 RESET
SHT4
SHT3
20L8
U54
PC A0-15
PCA0
SHT1
+5V
R39
2K
40MHZ
U51
74AS74
U55
S
C
D
Q
P01
9-PIN D-SHELL
J01
74LS244
U53
VD3
VD0
VD1
VD2
VD4
VD5
1A1
1A2
1A3
1A4
2A1
2A2
2A3
2A4
1G
2G
1Y1
1Y2
1Y3
1Y4
2Y1
2Y2
2Y3
2Y4
VD
HSYNC
VSYNC
SBHE
P6 SEN
SOFT RESET
PC RESET DRV
PCIOW
PAGE SELECT
PCP6IR
PCP61
PCMEMW
PCMEMR
PCIOR
P6READY DELAY
20L8
U52
74F125
U16B
245 T/R1
IOCHRDY
PC D LOW
PC/AT
ADDR ENABL
SHT1,2,4
SHT 2
INSERT JUMPER FOR
IBM AT INTERFACE.
NO JUMPER FOR
IBM PC/XT INTERFACE.
+5V
R03
5.1K
J02
+5V
R01
5.1K
SW01
SW02
R02
5.1K
7400
U56A
7400
U56B
FOR MANUAL RESET:
PRESS TO CLOSE
SW01 AND OPEN SW02
Intel Corporation
Graphics Components Operation
400 Encinal St., Santa Cruz, CA
TITLE
82786 DEMO BOARD
SIZE
B
CODE
NUMBER
001
REV
B
DATE
12-8-86
SHEET 5 of 8
8
7
6
5
4
3
2
1
D
C
B
A

240049-23


MAX
256x1
MA0 5 A0
MA1 7 A1
MA2 6 A2
MA3 12 A3
MA4 11 A4
MA5 10 A5
MA6 13 A6
MA7 9 A7
MA8 1 A8
DIN 2
DOUT 14
RAS 4
CAS 15
WE 3
U60 U61 U62 U63
U64 U65 U66 U67
U68 U69 U70 U71
U72 U73 U74 U75
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15
DO01 DO11 DO21 DO31 DO41 DO51 DO61 DO71 DO81 DO91 DO101 DO111 DO121 DO131 DO141 DO151
DOUT1
SHT8
P6 D0-15
SHT3
MAXX
74ALS373
DR0 DR1 DR2 DR3 DR4 DR5 DR6 DR7 DR8
1D 2D 3D 4D 5D 6D 7D 8D
1Q 2Q 3Q 4Q 5Q 6Q 7Q 8Q
G
OE
U601
U602
MAX
DRX
SHT3 RAS1
SHT3 CAS1
SHT3 WEL
SHT3 WEH
Intel Corporation
Graphics Components Operation
400 Encinal St . Santa Cruz CA
TITLE
82786 DEMO BOARD
SIZE B
CODE
NUMBER 001
REV B
DATE 12-8-86
SHEET 6 of 8
8 7 6 5 4 3 2 1
D C B A

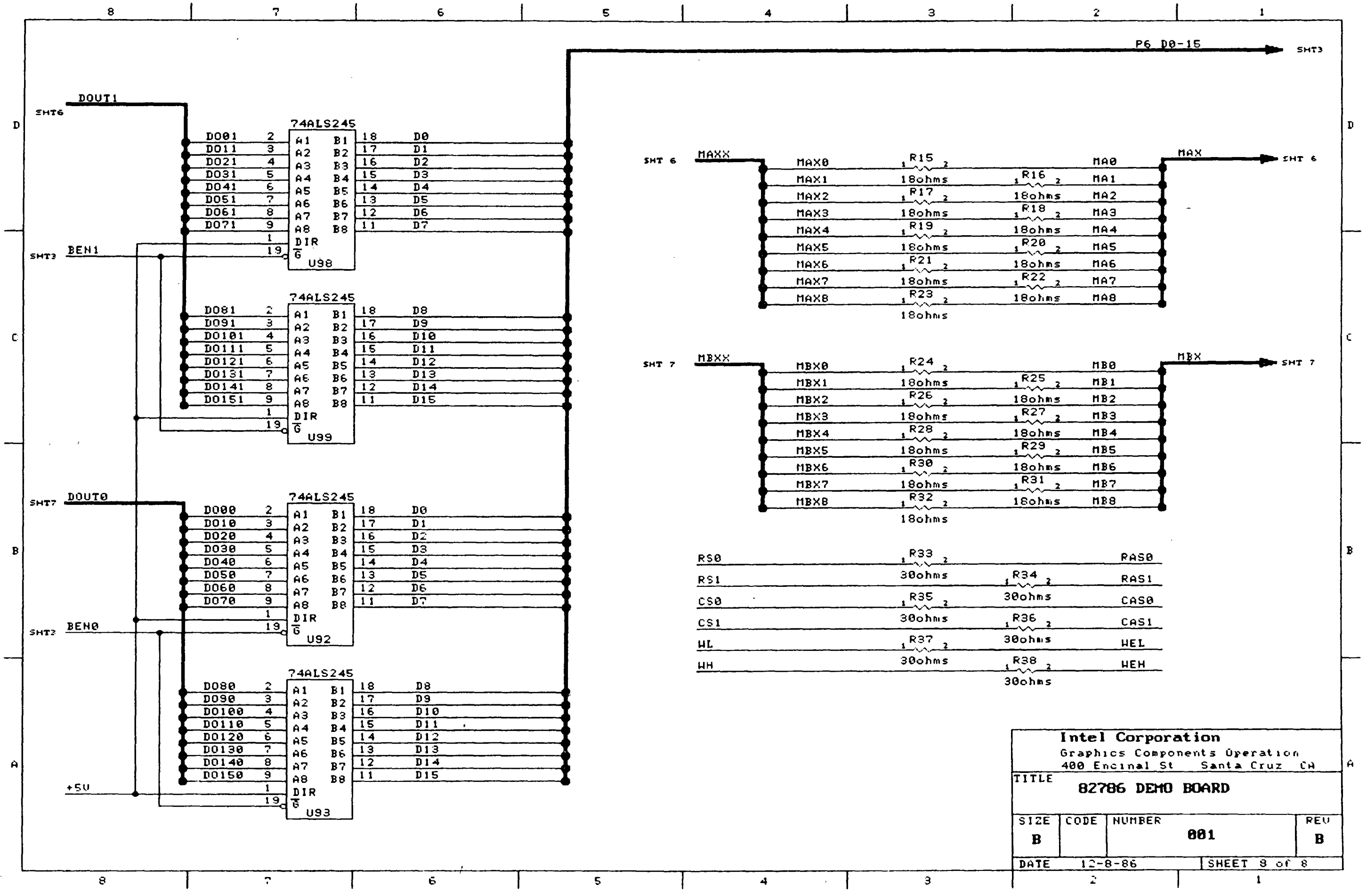
240049-24



Intel Corporation
Graphics Components Operation
400 Encinal St Santa Cruz CA

TITLE	82786 DEMO BOARD		
SIZE B	CODE	NUMBER 001	REV B
DATE	12-8-86	SHEET 7 of 8	

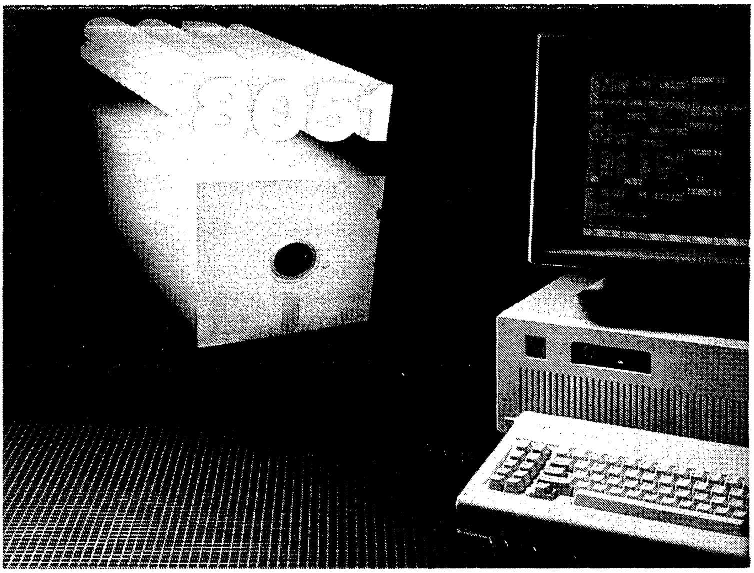
240049-25


P6 D0-15
SHT3
DOUT1
SHT6
74ALS245
DO01 DO11 DO21 DO31 DO41 DO51 DO61 DO71
A1 A2 A3 A4 A5 A6 A7 A8
B1 B2 B3 B4 B5 B6 B7 B8
D0 D1 D2 D3 D4 D5 D6 D7
DIR
G
U98
BEN1
SHT3
74ALS245
DO81 DO91 DO101 DO111 DO121 DO131 DO141 DO151
D8 D9 D10 D11 D12 D13 D14 D15
U99
DOUT0
SHT7
74ALS245
DO00 DO10 DO20 DO30 DO40 DO50 DO60 DO70
D0 D1 D2 D3 D4 D5 D6 D7
U92
BEN0
SHT3
74ALS245
DO80 DO90 DO100 DO110 DO120 DO130 DO140 DO150
D8 D9 D10 D11 D12 D13 D14 D15
U93
+5V
SHT 6
MAXX
MAX0 MAX1 MAX2 MAX3 MAX4 MAX5 MAX6 MAX7 MAX8
R15 R16 R17 R18 R19 R20 R21 R22 R23
18ohms
MA0 MA1 MA2 MA3 MA4 MA5 MA6 MA7 MA8
MAX
SHT 6
SHT 7
MBXX
MBX0 MBX1 MBX2 MBX3 MBX4 MBX5 MBX6 MBX7 MBX8
R24 R25 R26 R27 R28 R29 R30 R31 R32
18ohms
MB0 MB1 MB2 MB3 MB4 MB5 MB6 MB7 MB8
MBX
SHT 7
RS0 RS1 CS0 CS1 WL WH
R33 R34 R35 R36 R37 R38
30ohms
RAS0 RAS1 CAS0 CAS1 WEL WEH
Intel Corporation
Graphics Components Operation
400 Encinal St Santa Cruz CA
TITLE 82786 DEMO BOARD
SIZE B
CODE
NUMBER 001
REV B
DATE 12-8-86
SHEET 8 of 8

240049-26

Development Tools 11

8051 SOFTWARE DEVELOPMENT PACKAGES



COMPLETE SOFTWARE DEVELOPMENT SUPPORT FOR THE MCS®-51 FAMILY OF MICROCONTROLLERS

Intel supports application development for its MCS®-51 family of microcontrollers with a complete set of development languages and utilities. These tools include a macroassembler, a PL/M compiler, linker/relocator program, a librarian utility, and an object-to-hex utility. Develop code in the language(s) you desire, then combine object modules from different languages into a single, fast program. These tools were designed to work with each other, with the MCS-51 architecture, and with the Intel ICE5100 in-circuit emulator.

FEATURES

- Support for all members of the Intel MCS-51 family of embedded microcontrollers
- ASM-51 Macroassembler
- PL/M-51 high-level language
- Linker/Relocator program
- Library utility
- Object to hexadecimal converter
- Hosted on IBM PC XT/AT V. 3.0 or later
- Worldwide service and support

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

September, 1988
Order Number: 280819-001

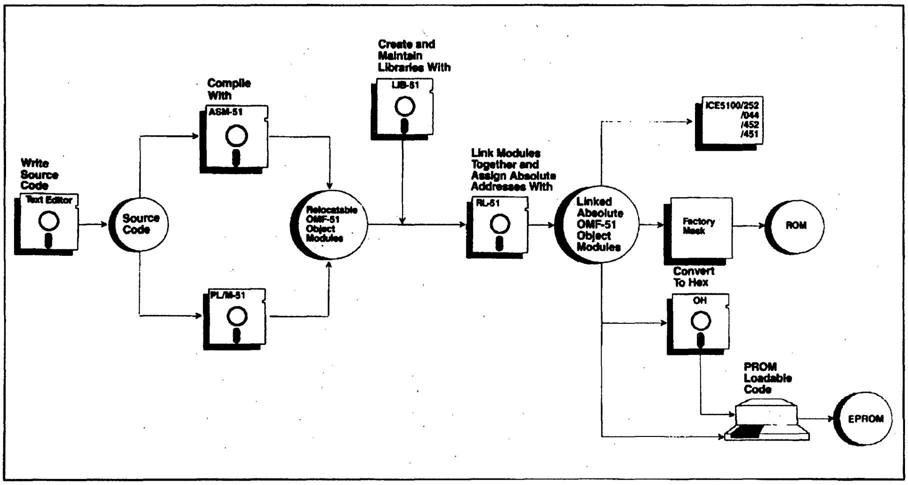

Create and Maintain Libraries With
LIB-51
Compile With
ASM-51
ICE5100/252 /044 /452 /451
Write Source Code
Text Editor
Source Code
Relocatable OMF-51 Object Modules
Link Modules Together and Assign Absolute Addresses With
RL-51
Linked Absolute OMF-51 Object Modules
Factory Mask
ROM
Convert To Hex
OH
PL/M-51
PROM Loadable Code
EPROM

Figure 1. MCS®-51Application Development Proccss

ASM-51 MACROASSEMBLER

ASM-51 is the macroassembler for the MCS-51 family of microcontrollers. ASM-51 provides full and accurate support for all of the specific component's instructions. It also provides symbolic access to the many features of the MCS-51 family of microcontrollers. Also provided is an "include" file with all the appropriate component registers and memory spaces defined.

The macro facility in ASM-51 saves development and maintenance time, since common code sequences need only be developed once.

PL/M-51 COMPILER

PL/M-51 is a high-level language designed to support the software requirements of the MCS-51 family of microcontrollers. The PL/M-51 compiler translates PL/M high-level language statements into MCS-51 relocatable object code. Major features of the PL/M-51 compiler include:

- **Structured programming for ease of maintenance and enhancement.** The PL/M-51 language supports modular and structured programming, making programs easier to understand, maintain, and debug.
- **Data types facilitate various common functions.** PL/M-51 supports three data types to facilitate various arithmetic, logic and address functions. The language also uses BASED variables that map more than one variable to the same memory location to save memory space.
- **Interrupt attribute speeds coding effort.** The INTERRUPT attribute allows you to easily define interrupt handling procedures. The compiler will generate code to save and restore the program status word for INTERRUPT procedures.
- **Code optimization reduces memory requirements.** The PL/M-51 compiler has four different levels of optimization for significantly reducing the size of the program.
- **Language compatibility saves development time.** PL/M-51 object modules are compatible with object modules generated by all other MCS-51 language translators. This compatibility allows for easy linking of all modules and the ability to do symbolic debugging with the Intel ICE5100 in-circuit emulator.

RL-51 Linker/Relocator

Intel's RL-51 utility is used to link multiple MCS-51 object modules into a single program, resolve all references between modules and assign absolute addresses to all relocatable segments. Modules can be written in either ASM-51 or PL/M-51.

LIB-51

The Intel LIB-51 utility creates and maintains libraries of software object modules. Standard modules can be placed in a library and linked into your applications programs using RL-51. When using libraries, the linker will link only those modules that are required to satisfy external references.

LIB-51 and RL-51 make it easy to reuse software modules that are fully debugged and used by various applications, thus shortening the software development cycle.

OH OBJECT TO HEXADECIMAL CONVERTER

The OH utility converts Intel OMF-51 object modules into standard hexadecimal format. This allows the code to be loaded directly into PROM via non-Intel PROM programmers.

SERVICE, SUPPORT, AND TRAINING

Intel augments its MCS-51 architecture family of development tools with a full array of seminars, classes, and workshops; on-site consulting services; field application engineering expertise; telephone hot-line support; and software and hardware maintenance contracts. This full line of services will ensure your design success.

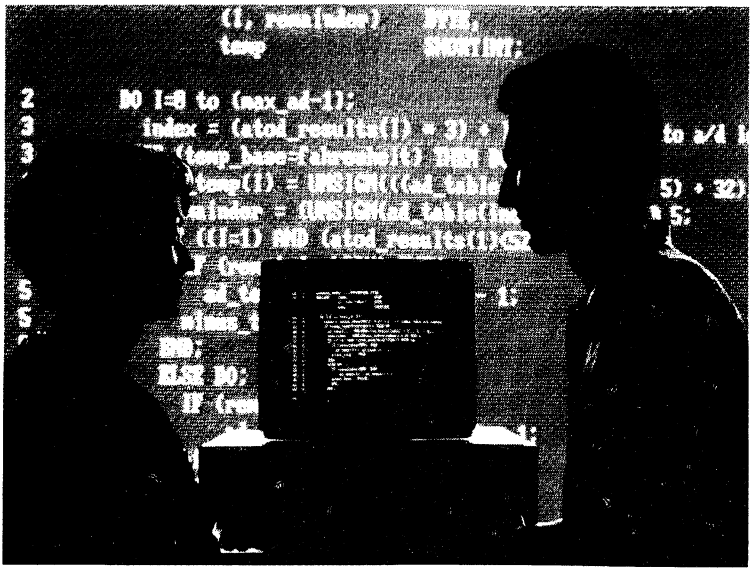
ORDERING INFORMATION

D86ASM51*	MCS-51 Assembler for PC XT or AT system (or compatible), running DOS 3.0 or higher
D86PLM51*	PL/M-51 Software Package for PC XT or AT system (or compatible), running DOS 3.0 or higher

*Also Includes: Relocator/Linker, Object-to-hex converter, and Librarian.

MCS is a registered trademark and ICE is a trademark of Intel Corporation
IBM and PC/AT are registered trademarks of International Business Machines Corporation

8096/196 Software Development Packages

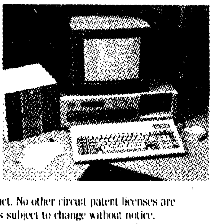


COMPLETE SOFTWARE DEVELOPMENT SUPPORT FOR THE 8096/196 FAMILY OF MICROCONTROLLERS

Intel supports application development for its 8096 and 80C196 family of microcontrollers with a complete set of development languages and utilities. These tools include a macroassembler, a PL/M compiler, a C compiler, linker/relocator program, floating point arithmetic library, a librarian utility, and an object-to-hex utility. Develop code in the language(s) you desire, then combine object modules from different languages into a single, fast program.

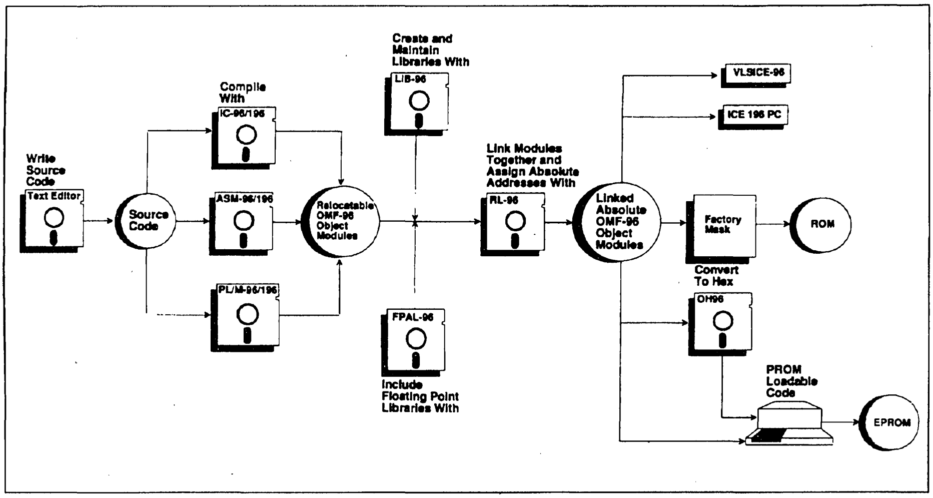
FEATURES

- Software Tools support all members of Intel's MCS®-96 family
- ASM-96/196 macroassembler for speed critical code
- PL/M-96/196 package for the maintainability and reliability of a high-level language with support for many low-level hardware functions
- iC-96/196 package for structured C language programming, with many hardware specific extensions
- Linker/Relocator program for linking modules generated in assembler, PL/M or C and assigning absolute addresses to relocatable code. RL-96 prepares your code for execution in target with a simple, one-step operation
- 32-bit Floating Point Arithmetic Library to reduce your development effort and to allow fast, highly optimized numerics-intensive processing
- Library utility for creating and maintaining software object module libraries
- PROM building utility that converts object modules into standard hexadecimal format for easy download into a non-Intel PROM Programmer
- Hosted on IBM PC XT/AT with PC-DOS 3.0 or above



Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

September, 1988
Order Number: 280793-002



Create and Maintain Libraries With
LIB-96
Compile With
IC-96/196
VLSICE-96
ICE 196 PC
Write Source Code
Text Editor
Source Code
ASM-96/196
Relocatable OMF-96 Object Modules
Link Modules Together and Assign Absolute Addresses With
RL-96
Linked Absolute OMF-96 Object Modules
Factory Mask
ROM
Convert To Hex
OH96
PL/M-96/196
FPAL-96
Include Floating Point Libraries With
PROM Loadable Code
EPROM

Figure 1. MCS®-96 Application Development Process

ASM-96/196 MACROASSEMBLER

ASM-96/196 is the macroassembler for the MCS-96 family of microcontrollers, including the 80C196. ASM-96/196 translates symbolic assembly language mnemonics into relocatable object code.

The macro facility in ASM-96/196 saves development and maintenance time, since common code sequences need only be developed once. The assembler also supports symbolic access to the many features of the 8096/196 and provides an "include" file with all 8096/196 registers defined.

PL/M-96/196 SOFTWARE PACKAGE

PL/M-96/196 is a high-level programming language designed to support the software requirements of advanced 16-bit microcontrollers. The PL/M-96/196 compiler translates PL/M high-level language statements into 8096/196 relocatable object code. Major features of the PL/M-96/196 compiler include:

- **Structured programming.** The PL/M language supports modular and structured programming, making programs easier to understand, maintain, and debug.
- **Built-in functions.** PL/M-96/196 includes an extensive list of functions, including TYPE CONVERSION functions, STRING manipulations, and functions for interrogating MCS-96 hardware flags.
- **Interrupt handling.** The INTERRUPT attribute allows you to define interrupt handling procedures. The compiler generates code to save and restore the program status word for INTERRUPT procedures.
- **Compiler controls.** Compile-time options increase the flexibility of the PL/M-96/196 compiler. These controls include: optimization, conditional compilation, the inclusion of common PL/M source files from disk, cross-reference of symbols, and optional assembly language code in the listing file.
- **Data types.** PL/M-96/196 supports seven data types, allowing PL/M-96/196 to perform three different kinds of arithmetic: signed, unsigned, and floating point.
- **Language compatibility.** PL/M-96/196 object modules are compatible with all other object modules generated by Intel MCS-96 translators.

iC-96/196 SOFTWARE PACKAGE

Intel's iC-96/196 is a structured programming language designed to support applications for the 16-bit family of MCS-96 microcontrollers. iC-96/196 implements the C language as described in the Kernighan and Ritchie book, *The C Programming Language*, and includes many of the enhancements as defined by the proposed ANSI C standard. Major features of the iC-96/196 compiler include:

- **Symbolics.** The iC-96/196 compiler boosts programmer productivity by providing extensive debug information, including symbols. The debug information can be used to debug the code using either the VLSiCE™-96 emulator or the ICE™-196PC emulator.
- **Architecture Support.** iC-96/196 generates code which is fully optimized for the MCS-96 architecture. iC-96/196 provides an INTERRUPT attribute, allowing you to define interrupt handling functions in C, and library routines which allow you to enable and disable interrupts directly from C (mid-1989). A REENTRANT/NOREENTRANT control is also included, allowing the compiler to identify non-reentrant procedures. This gives you full access to the large MCS-96 register set.
- **Standard language.** iC-96/196 accepts standard C source code. iC-96/196 code is fully linkable with both PL/M-96/196 and ASM-96/196 modules via an "alien" attribute, allowing programmers to utilize the optimal language for any application. In addition, programmers can quickly begin programming with iC-96/196 because it conforms to accepted C language standards.

RL-96/196 LINKER/RELOCATOR

Intel's RL-196 utility is used to link multiple MCS-96 object modules into a single program and then assign absolute addresses to all relocatable addresses in the new program. Modules can be written in ASM-96/196, PL/M-96/196, or iC-96/196.

The RL-96/196 utility also promotes programmer productivity by encouraging modular programming. Because applications can be broken into separate modules, they're easier to design, test and maintain. Standard modules can be reused in different applications, saving software development time.

FPAL-96/196 FLOATING POINT ARITHMETIC LIBRARY

FPAL-96/196 is a library of single-precision 32-bit floating point arithmetic functions. These functions are compatible with the IEEE floating point standard for accuracy and reliability and include an error-handler library.

LIB-96/196

The Intel LIB-96/196 utility creates and maintains libraries of software object modules. Standard modules can be placed in a library, and linked into your applications programs using RL-96/196.

OH-96/196

The OH-96/196 utility converts Intel OMF-96 object modules into standard hexadecimal format. This allows the code to be loaded directly into a PROM via non-Intel PROM programmers.

SERVICE, SUPPORT, AND TRAINING

Intel augments its MCS-96 architecture family development tools with a full array of seminars, classes, and workshops; on-site consulting services; field application engineering expertise; telephone hot-line support; and software and hardware maintenance contracts. This full line of services will ensure your design success.

ORDERING INFORMATION

D86ASM96NL*	96/196 Assembler for PC XT or AT system (or compatible), running DOS 3.0 or higher
D86PLM96NL*	PL/M-96/196 Software Package for PC XT or AT system (or compatible), running DOS 3.0 or higher
D86C96NL*	iC-96/196 Software Package for PC XT or AT system (or compatible), running DOS 3.0 or higher

*Also Includes: Relocator/Linker, Object-to-hex converter, Floating Point Arithmetic Library, and Librarian.

VAX*/VMS* RESIDENT 86/88/186/188 SOFTWARE DEVELOPMENT PACKAGES

- **Executes on DEC VAX*/MicroVAX Minicomputer under VMS* Operating System to translate PL/M-86, Fortran-86, iC-86, Pascal-86 and ASM-86 Programs for 8086, 88, 186 and 188 Microprocessors.**
- **Specifically Designed for Embedded Microcomputer Applications**
- **Packages include C-86; FORTRAN-86; Pascal-86; PL/M-86; ASM-86; Link and Relocation Utilities; OH-86 Absolute Object Module to Hexadecimal Format Converter; and Library Manager Program.**
- **Output Linkable with Code Generated on PC-DOS Based Systems**

The VAX/VMS Resident Software Development Packages contain software development tools for the 8086, 88, 186, and 188 microprocessors. The tools allow the user to develop, compile, maintain libraries, and link and locate programs on a VAX running the VMS operating system. The compiler or assembler output is object module compatible with programs translated by the corresponding version of the compiler or assembler on a PC-DOS based system.

Four packages are available:

1. An ASM-86 Assembler Package which includes the Assembler, the Link Utility, the Locate Utility, the absolute object to hexadecimal format conversion utility, the Library Manager Program, and Numerics Support Library.
2. A PL/M-86 Compiler Package which contains the PL/M-86 Compiler and Runtime Support Libraries.
3. A Pascal-86 Compiler Package which contains the Pascal-86 Compiler and Runtime Support Libraries.
4. A iC-86 Compiler Package which contains the iC-86 Compiler and Run-Time Libraries.
5. A FORTRAN-86 Compiler Package which contains the FORTRAN-86 Compiler and Run-Time Libraries.

The VAX/VMS resident development packages and the PC-DOS hosted development packages are built from the same technology base. Therefore, the development packages are very similar.

Version numbers can be used to identify features correspondence. The VAX/VMS resident development packages will have the same features as the PC-DOS hosted product with the same version number.

The object modules produced by the translators contain symbol and type information for program debugging using ICE™ translators and/or the PSCOPE debugger. For final production version, the compiler can remove this extra information and code, to reduce final program size.

*VAX, DEC, and VMS are trademarks of Digital Equipment Corporation.

VAX*-PL/M-86/88/186/188 SOFTWARE PACKAGE

- **Executes on VAX*/MicroVAX Minicomputers under the VMS* Operating System**
- **Supports 16-Bit Signed Integer and 32-Bit Floating Point Arithmetic in Accordance with IEEE Standards**
- **Easy-To-Learn Block-Structured Language Encourages Program Modularity**
- **Produces Relocatable Object Code Which is Linkable to All Other Intel 8086 Object Modules, Generated on Either a VAX*, or a PC-DOS Hosted System**
- **Code Optimization Assures Efficient Code Generation and Minimum Application Memory Utilization**
- **Built-In Syntax Checker Doubles Performance for Compiling Programs Containing Errors**
- **ICE™, PSCOPE Symbolic Debugging Fully Supported**

VAX-PL/M-86 is an advanced, structured high-level programming language. The VAX-PL/M-86 compiler was created specifically for embedded software development for the Intel 8086, 88, 186, and 188 microprocessors.

PL/M is a powerful, structured, high-level system implementation language in which program statements can naturally express the program algorithm. This frees the programmer to concentrate on the logic of the program without concern for burdensome details of machine or assembly language programming (such as register allocation, meanings of assembler mnemonics, etc.).

The VAX-PL/M-86 compiler efficiently converts free-form PL/M language statements into equivalent machine instructions. Substantially fewer PL/M statements are necessary for a given application than if it were programmed at the assembly language or machine code level.

The use of PL/M high-level language for system programming, instead of assembly language, results in a high degree of engineering productivity during project development. This translates into significant reductions in initial software development and follow-on maintenance costs for the user.

*VAX, DEC, and VMS are trademarks of Digital Equipment Corporation.

VAX*-PASCAL-86/88/186/188 SOFTWARE PACKAGE

- **Executes on VAX*/MicroVAX Minicomputers under the VMS* Operating System**
- **Produces Relocatable Object Code Which is Linkable to All Other Intel 8086 Object Modules, Generated on either a VAX*, or a PC-DOS Hosted System**
- **ICE™, PSCOPE Symbolic Debugging Fully Supported**
- **Supports Numeric Coprocessors**
- **Conforms to IEEE/ANSI and ISO Standards**
- **Extensions for Embedded Microcomputer Applications**
- **Separate Compilation with Type-Checking Enforced between Pascal Modules**
- **Compiler Option to Support Full Run-Time Range-Checking**
- **Source Input/Object Output Compatible with Pascal-86 Hosted on a PC-DOS or RMX Development Host**

VAX-PASCAL-86 conforms to and implements the ISO Pascal standard, level 0. The language is enhanced to support embedded microcomputer applications with special features, such as separate compilation, interrupt handling and direct port I/O. Other extensions include additional data types not required by the standard and miscellaneous enhancements such as an allowed underscore in names, and an OTHERWISE clause in CASE statements. To assist the development of portable software, the compiler can be directed to flag all non-standard features.

The VAX-PASCAL-86 compiler runs on the Digital Equipment Corporation VAX under the VMS Operating System. A well-defined I/O interface is provided for run-time support. This allows a user-written operating system to support application programs on the target system as an alternate to the development system environment. Program modules compiled under PASCAL-86 are compatible and linkable with modules written in PL/M-86, and ASM-86. With a complete family of compatible programming languages for the 8086, 88, 186, and 188, code can be developed in the language most appropriate to the task at hand.

*VAX, DEC, and VMS are trademarks of Digital Equipment Corporation.

VAX* ASM-86/88/186/188 MACRO ASSEMBLER

- **Executes on VAX*/MicroVAX Minicomputers under The VMS* Operating System**
- **Produces Relocatable Object Code Which is Linkable to Other Intel 8086/88/186 Object Modules, Generated on either a VAX*, or a PC XT/AT running PC-DOS**
- **Powerful and Flexible Text Macro Facility with Three Macro Listing Options to Aid Debugging**
- **Highly Mnemonic and Compact Language, Most Mnemonics Represent Several Distinct Machine Instructions**
- **"Strongly Typed" Assembler Helps Detect Errors at Assembly Time**
- **High-Level Data Structuring Facilities Such as "STRUCTURES" and "RECORDS"**
- **Detailed and Fully Documented Error Messages**
- **Produces Relocatable and Linkable Object Code**
- **Source Input/Object Output Compatible with ASM-86 hosted on a PC-DOS Hosted System**

VAX-ASM-86 is the "high-level" macro assembler for the 8086/88/186/188 assembly language. VAX-ASM-86 translates symbolic assembly language mnemonics into relocatable object code.

VAX-ASM-86 should be used where maximum code efficiency and hardware control is needed. The assembly language includes approximately 100 instruction mnemonics. From these few mnemonics the assembler can generate over 3,800 distinct machine instructions. Therefore, the software development task is simplified, as the programmer need know only 100 mnemonics to generate all possible machine instructions. VAX-ASM-86 will generate the shortest machine instruction possible given no forward referencing or given explicit information as to the characteristics of forward referenced symbols.

VAX-ASM-86 offers many features normally found only in high-level languages. The ASM-86 assembly language is strongly typed. The assembler performs extensive checks on the usage of variable and labels. The assembler uses the attributes which are derived explicity when a variable or label is first defined, then makes sure that each use of the symbol in later instructions conforms to the usage defined for that symbol. This means that many programming errors will be deteced when the program is assembled, long before it is being debugged on hardware.

*VAX, DEC, and VMS are trademarks of Digital Equipment Corporation.

VAX*-LIB-86

- **Executes on VAX*/MicroVAX Minicomputers under the VMS* Operating System**
- **VAX-LIB-86 is a Library Manager Program which Allows You to:**
  Create Specifically Formatted Files to Contain Libraries of Object Modules
  Maintain These Libraries by Adding or Deleting Modules
  Print a Listing of the Modules and Public Symbols in a Library File
- **Libraries Can be Used as Input to VAX-LINK-86 Which Will Automatically Link Modules from the Library that Satisfy External References in the Modules Being Linked**
- **Abbreviated Control Syntax**

Libraries aid in the job of building programs. The library manager program VAX-LIB-86 creates and maintains files containing object modules. The operation of VAX-LIB-86 is controlled by commands to indicate which operation VAX-LIB-86 is to perform. The commands are:

CREATE:	creates an empty library file
ADD:	adds object modules to a library file
DELETE:	deletes modules from a library file
LIST:	lists the module directory of library files
EXIT:	terminates the LIB-86 program and returns control to VMS

When using object libraries, the linker will call only those object modules that are required to satisfy external references, thus saving memory space.

VAX-OH-86

- **Executes on VAX*/MicroVAX Minicomputers under the VMS* Operating System**
- **Converts an 8086/88/186/188 Absolute Object Module to Symbolic Hexademical Format**
- **Facilitates Preparing a file for Loading by Symbolic Hexadecimal Loader (e.g. iSBC® Monitor SDK-86 Loader), or Universal PROM Mapper**
- **Converts an Absolute Module to a More Readable Format that can be Displayed on a CRT or Printed for Debugging**

The VAX-OH-86 utility converts an absolute object module to the hexadecimal format. This conversion may be necessary for later loading by a hexadecimal loader such as the iSBC 86/12 monitor or the Universal PROM Mapper. The conversion may also be made to put the module in a more readable format that can be displayed or printed.

The module to be converted must be in absolute form; the output from VAX-LOC-86 is in absolute format.

*VAX, VMS are trademarks of Digital Equipment Corporation.

VAX*-LINK-86

- **Executes on VAX*/MicroVAX Minicomputers under the VMS* Operating System**
- **Automatic Combination of Separately Compiled or Assembled 86/88/186/188 Programs into a Relocatable Module, Generated on Either a VAX, or a PC XT/AT running PC-DOS**
- **Automatic Selection of Required Modules from Specified Libraries to Satisfy Symbolic References**
- **Extensive Debug Symbol Manipulation, allowing Line Numbers, Local Symbols, and Public Symbols to be Purged and Listed Selectively**
- **Automatic Generation of a Summary Map Giving Results of the LINK-86 Process**
- **Abbreviated Control Syntax**
- **Relocatable modules may be Merged into a Single Module Suitable for Inclusion in a Library**
- **Supports "Incremental" Linking**
- **Supports Type Checking of Public and External Symbols**

VAX-LINK-86 combines object modules specified in the VAX-LINK-86 input list into a single output module. VAX-LINK-86 combines segments from the input modules according to the order in which the modules are listed.

VAX-LINK-86 will accept libraries and object modules built from any Intel translator generating 8086 Relocatable Object Modules.

Support for incremental linking is provided since an output module produced by VAX-LINK-86 can be an input to another link. At each stage in the incremental linking process, unneeded public symbols may be purged.

VAX-LINK-86 supports type checking of PUBLIC and EXTERNAL symbols reporting a warning if their types are not consistent.

VAX-LINK-86 will link any valid set of input modules without any controls. However, controls are available to control the output of diagnostic information in the VAX-LINK-86 process and to control the content of the output module.

VAX-LINK-86 allows the user to create a large program as the combination of several smaller, separately compiled modules. After development and debugging of these component modules the user can link them together, locate them using VAX-LOC-86 and enter final testing with much of the work accomplished.

*VAX, DEC, and VMS are trademarks of Digital Equipment Corporation.

VAX*-LOC-86

- **Executes on the VAX*/MicroVAX Minicomputers under the VMS* Operating System**
- **Automatic Generation of a Summary Map Giving Starting Address, Segment Addresses and Length, and Debug Symbols and their Addresses**
- **Extensive Capability to Manipulate the Order and Placement of Segments in Memory**
- **Abbreviated Control Syntax**
- **Automatic and Independent Relocation of Independent Relocation of Segments. Segments May be Relocated to Best Match Users Memory Configuration**
- **Extensive Debug Symbol Manipulation, Allowing Line Numbers, Local Symbols, and Public Symbols to be Purged and Listed Selectively**

Relocatability allows the programmer to code programs or sections of programs without having to know the final arrangement of the object code in memory.

VAX-LOC-86 converts relative addresses in an input module in iAPX-86/88/186/188 object module format to absolute addresses. VAX-LOC-86 orders the segments in the input module and assigns absolute addresses to the segments. The sequence in which the segments in the input module are assigned absolute addresses is determined by their order in the input module and the controls supplied with the command.

VAX-LOC-86 will relocate any valid input module without any controls. However, controls are available to control the output of diagnostic information to control the content of the output module, or both.

The program you are developing will almost certainly use some mix of random access memory (RAM), read-only memory (ROM), and/or programmable read-only memory (PROM). Therefore, the location of your program affects both cost and performance in your application. The relocation feature allows you to develop your program and then simply relocate the object code to suit your application.

*VAX, DEC, and VMS are trademarks of Digital Equipment Corporation.

SPECIFICATIONS

Operating Environment

Required Hardware

VAX* with 9 Track Magnetic Tape Drive, 1600 BPI

MicroVAX with TK-50 tape drive.

Required Software

VMS Operating System. All of the development packages are delivered as unlinked VAX object code which can be linked to VMS as designed for the system where the development package is to be used. VMS command files to perform the link are provided.

MicroVMS

Documentation Package

iAPX-86, 88 Development Software Installation Manual and User's Guide for VAX/VMS, Order number 121950-001

Shipping Media

9 Track Magnetic Tape 1600 bpi (VAX)

TK-50 Cartridge Tape (MicroVAX)

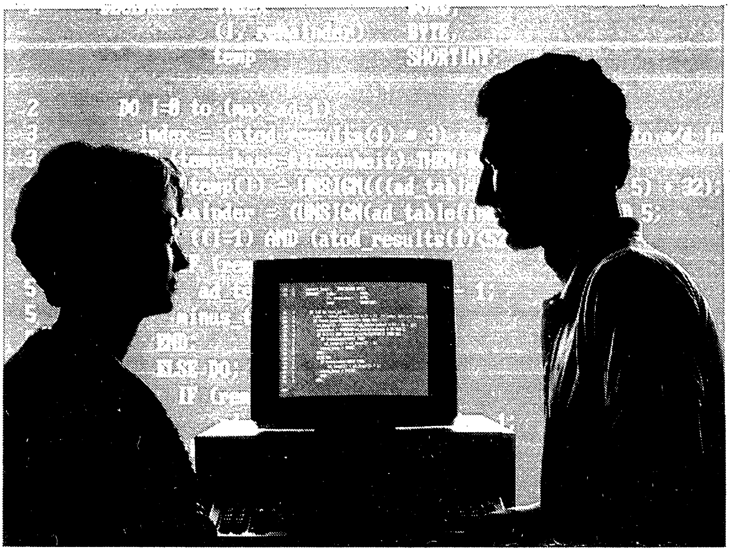
ORDERING INFORMATION

Part Number	Description
VVSASM86	VAX-ASM-86, VAX-LINK-86, VAX-LOC-86, VAX-LIB-86, VAX-OH-86, Package
VVSPLM86	VAX-PLM-86 Package
VVSPAS86	VAX-PASCAL-86 Package
VVSC86	VAX-C-86 Package
VVSFORT86	VAX-FORTRAN-86 Package
MVVSASM86	MICROVAX ASM86 Package
MVVSPLM86	MICROVAX PLM86 Package
MVVSC86	MICROVAX C86 Package
MVVSFORT86	MICROVAX FORTRAN 86 Package
MVVSPAS86	MICROVAX PASCAL-86 Package

REQUIRES SOFTWARE LICENSE

PRELIMINARY

8086/80186 SOFTWARE DEVELOPMENT PACKAGES



COMPLETE SOFTWARE DEVELOPMENT SUPPORT FOR THE 8086/80186 FAMILY OF MICROPROCESSORS

Intel supports application development for the 8086/80186 family of microprocessors (8086, 8088, 80186, 80188 and real mode 80286 and 80386 designs) with a complete set of development languages and utilities. These tools include a macro assembler and compilers for C, PL/M, FORTRAN and Pascal. A linker/relocator program, library manager, numerics support libraries, and object-to-hex utility are also available. Intel software tools generate fast and efficient code. They are designed to give maximum control over the processor. Most importantly, they are designed to get your application up and running in an embedded system fast and with maximum design productivity.

FEATURES

- Macro assembler for speed-critical code
- NEW windowed, interactive source level debugger works with all Intel languages to speed functional debug of code
- ANSI Compatible iC-86 package for structured C programming, with many processor specific extensions
- PL/M compiler for reliable, easy-to-maintain high-level language programs with support for many low-level hardware functions
- FORTRAN for ANSI-compatible programming of numeric intensive applications
- Pascal for developing modular, portable applications that are easy to maintain
- Linker program for linking modules generated by Intel compilers and assemblers into relocatable object modules and direct creation of files that can be executed on the DOS host
- Locator for generating programs with absolute addresses for execution from ROM based systems
- AEDIT Source Code and text editor
- Library manager for creating and maintaining software object module libraries
- Complete numeric support libraries for use with the 8087 or 80C187, including a software emulator for true emulation of the 8087 in applications where the 8087 is not available
- Object-to-hex conversion utility for burning code into (E)PROMS
- Hosted on IBM PC XT/AT* or compatibles running DOS, DEC VAX* or MicroVAX* systems running VMS, and Intel Systems 86/3XX or 286/3XX running iRMX® Operating System

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

October, 1988
Order Number: 280809-001

ASM-86 MACRO ASSEMBLER

ASM-86 is the macro assembler for the 8086/80186 family of components. It is used to translate symbolic assembly language source into relocatable object code where utmost speed, small code size and hardware control are critical. Intel's exclusive macro facility in ASM-86 saves development and maintenance time, since common code sequences need only be developed once. The assembler's simplified instruction set reduces the number of mnemonics that the programmer needs to remember. This assembler also saves development time by performing extensive checks on consistent usage of variables and labels. Inconsistencies are detected when the program is assembled, before linking or debugging is started.

NEW FOR 1989: SOURCE LEVEL DEBUGGER

DB-86 is an on-host software execution envrionment with source level debug capabilities for object modules produced by iC-86, ASM-86, PL/M-86, Pascal-86 and FORTRAN-86. Its powerful, source-oriented interface allows users to focus their efforts on finding bugs rather than spending time learning and manipulating the debug environment.

- **Ease of learning.** Drop-down menus make the tool easy to learn for new or casual users. A command line interface is also provided for more complex problems.
- **Extensive debug modes.** Watch windows, conditional breakpoints (breakpoints triggered by program conditions), trace points, and fixed and temporary breakpoints can be set and modified as needed.
- **See into your program.** You can browse source and call stack, observe processor registers, output screen, and watch window variables accessed by either the pull down menu or by a single keystroke using function keys.
- **Full debug symbolics for maximum productivity.** The user need not know whether a variable is an unsigned integer, a real, or a structure; the debugger utilizes the wealth of variable typing information available in Intel languages to display program variables in their respective type formats.
- **Support for overlaid programs and the numeric coprocessor.**

iC-86 SOFTWARE PACKAGE

Intel's iC-86 brings the full power of the C programming language to 8086, 8088, 80186 and 80188-based microprocessor systems. It can also be used to develop real mode programs for execution on the 80286 or 80386. iC-86 has been developed specifically for embedded microprocessor-based applications. iC-86 meets the draft proposed ANSI C standard. Key features of the iC-86 compiler include:

- **Highly Optimized.** Four levels of optimization are available. Important optimization features include a jump optimizer and improved register manipulation via register history.
- **ROMable Code and Libraries.** The iC-86 compiler produces ROMable code which can be loaded directly into embedded target systems. Libraries are also completely ROMable, retargetable and reentrant.
- **Supports Small, Medium, Compact, and Large memory segmentation models.**
- **Symbolics.** The iC-86 compiler boosts programming productivity by providing extensive debug information, including type information and symbols. The symbolics information can be used to debug using Intel ICE™ emulators and the new DB-86 Source level debugger.
- **Built-in functions.** iC-86 is loaded with built-in functions. The flags register, I/O ports, interrupts, and numerics chip can be controlled directly, without the need for assembly language coding. You spend more of your productive time programming in C and less with Assembler. Built-in functions also improve compile-time and run-time performance since the compiler generates in-line code instructions instead of function calls to assembly instructions.
- **Standard Language.** iC-86 conforms to the 1988 Draft Proposed ANSI standard for the C language. iC-86 code is fully linkable with other modules written in other Intel 8086/186 languages, allowing programmers to use the optimal language for any task.

PL/M-86 SOFTWARE PACKAGE

PL/M-86 is a high-level programming language designed to support the software requirements of advanced 16-bit microprocessors. PL/M-86 provides the productivity advantages of a high-level language while providing the low-level hardware access features of assembly language. Key features of PL/M-86 include:

- **Structured programming.** PL/M-86 supports modular and structured programming, making programs easier to understand, maintain and debug.
- **Built-in functions.** PL/M-86 includes an extensive list of functions, including TYPE CONVERSION functions, STRING manipulations, and functions for interrogating 8086/186 hardware flags.
- **Interrupt handling.** The INTERRUPT attribute allows you to define interrupt handling procedures. The compiler generates code to save and restore all registers for INTERRUPT procedures.
- **Compiler controls.** Compile-time options increase the flexibility of the PL/M-86 compiler. They include: optimization, conditional compilation, the inclusion of common PL/M source files from disk, cross-reference of symbols, and optional assembly language code in the listing file.
- **Data types.** PL/M-86 supports seven data types, allowing the compiler to perform three different kinds of arithmetic: signed, unsigned and floating point.
- **Language compatibility.** PL/M-86 object modules are compatible with all other object modules generated by Intel 8086/186 languages.

FORTRAN-86 SOFTWARE PACKAGE

FORTRAN-86 meets the ANSI FORTRAN 77 Language Subset Specification and includes almost all of the features of the full standard. This compatibility assures portability of existing FORTRAN programs and shortens the development process, since programmers are immediately productive without retraining.

FORTRAN-86 provides extensive support for numeric processing tasks and applications, with features such as:

- Support for single, double, double extended precision, complex, and double complex floating-point data types
- Support for proposed REALMATH IEEE floating point standard
- Full support for all other data types: integer, logical and character
- Optional hardware (8087 numeric data processor) or software (simulator) floating-point support at link time

PASCAL-86 SOFTWARE PACKAGE

Pascal-86 conforms to the ISO Pascal standard, facilitating application portability, training and maintenance. It has also been enhanced with microcomputer support features such as interrupt handling, direct port I/O and separate compilation.

A well-defined and documented run-time operating system interface allows the user to execute applications under user-designed operating systems as an alternate to the development system environment. Program modules compiled under Pascal-86 are compatible and linkable with modules written in other Intel 8086/186 languages, so developers can implement each module in the language most appropriate for the task at hand.

Pascal-86 object modules contain symbol and type information for program debugging using Intel ICE™ emulators and the DB-86 debugger.

LINK-86 LINKER

Intel's LINK-86 utility is used to combine multiple object modules into a single program and resolve references between independently compiled modules. The resulting linked module can be either a bound load-time-locatable module or simply a relocatable module. A .EXE option allows modules to be generated which can be executed directly on a DOS system.

LINK-86 greatly increases productivity by allowing you to use modular programming. The incremental link capability allows new modules to be easily added to existing software. Because applications can be broken into separate modules, they're easier to design, test and maintain. Standard modules can be reused in different applications, saving software development time.

LOC-86 LOCATOR

The LOC-86 utility changes relocatable 8086/186 object modules into absolute object modules. Its default address assignment algorithm will automatically assign absolute addresses to the object modules prior to loading of the code into the target system. This frees you from concern about the final arrangement of the object code in memory. You still have the power to override the control and specify absolute addresses for various Segments, Classes, and Groups in memory. You may also reserve various parts of memory.

LOC-86 is a powerful tool for embedded development because it simplifies set up of the bootstrap loader and initialization code for execution from ROM based systems. The locator will also optionally generate a print file containing diagnostic information to assist in program debugging.

NUMERICS SUPPORT LIBRARY

The 8087 Support Library (80C187 support in June 1989) greatly facilitates the use of floating-point calculations from programs written in Assembler, PL/M, and C. It adds to these languages many of the functions that are built into applications programming languages, such as Pascal and FORTRAN. A full 8087 software emulator and interface libraries are included for precision floating point calculations without the use of the 8087 component. The decimal conversion library aids the translation between decimal and binary formats. A Common Elementary Function library provides support for transcendental, rounding and other common functions, not directly handled by the numeric processor. An Error Handler Module makes it easy to write interrupt routines that recover from floating-point error conditions.

LIB-86 LIBRARIAN

The Intel LIB-86 utility creates and maintains libraries of software object modules. Standard modules can be placed in a library and linked to your application using the LINK-86 utility.

AEDIT SOURCE CODE AND TEXT EDITOR

AEDIT is a full-screen text editing system designed specifically for software engineers and technical writers. With the facilities for automatic program block indentation, HEX display and input, and full macro support, AEDIT is an essential tool for any programming environment. And with AEDIT, the output file is the pure ASCII text (or HEX code) you input—no special characters or proprietary formats.

Dual file editing means you can create source code and its supporting documents at the same time. Keep your program listing with its errors in the background for easy reference while correcting the source in the foreground. Using the split-screen windowing capability, it is easy to compare two files, or copy text from one to the other. The DOS system-escape command eliminates the need to leave the editor to compile a program, get a directory listing, or execute any other program executable at the DOS system level.

OH-86 OBJECT-TO-HEXADECIMAL CONVERTER

The OH-86 utility converts Intel 8086/186 object modules into standard hexadecimal format, allowing the code to be loaded directly into PROM using industry standard PROM programmers.

SERVICE, SUPPORT AND TRAINING

Intel augments its 8086/186 family development tools with a full array of seminars, classes and workshops. In addition, on-site consulting services, field application engineering expertise, telephone hotline support, and software and hardware maintenance contracts are available to help assure your design success.

ORDERING INFORMATION

D86ASM86NL	ASM-86	Assembler for PC XT or AT system (or compatible) running DOS 3.0 or higher
VVSASM86	ASM-86	Assembler for VAX/VMS
MVVSASM86	ASM-86	Assembler for MicroVAX/VMS
R86ASM86SU	ASM-86	Assembler for Intel 86/3XX systems running iRMX 86 operating system
R286ASM86EU	ASM-86	Assembler for Intel 286/3XX systems running iRMX II™ operating system
Note:	ASM-86 includes Macro Assembler, Link-86, Loc-86, Lib-86, Cross-Reference utility, OH-86, Numerics Support, and DB-86 Source Level Debugger. (DB-86 available in DOS version only.)	
D86C86NL	iC-86	Software Package for IBM PC XT/AT running PC DOS 3.0 or higher
VVSC86	iC-86	Software Package for VAX/VMS
MVVSC86	iC-86	Software Package for MicroVAX/VMS
R86C86SU	iC-86	Software Package for Intel System 8086/3XX running iRMX 86 operating system
D86PLM86NL	PL/M-86	Software Package for IBM PC XT/AT running PC DOS 3.0 or higher
VVSPLM86	PL/M-86	Software Package for VAX/VMS
MVVSPLM86	PL/M-86	Software Package for MicroVAX/VMS
R86PLM86SU	PL/M-86	Software Package for Intel System 8086/3XX running iRMX 86 operating system
D86FOR86NL	FORTRAN-86	Software Package for PC XT/AT (or compatible) running PC-DOS 3.0 or higher
VVSFORT86	FORTRAN-86	Software Package for VAX/VMS 4.3 and later
MVVSFORT86	FORTRAN-86	Software Package for MicroVAX/VMS
R86FOR86SU	FORTRAN-86	Software Package for Intel System 86/3XX running iRMX 86 operating system
D86PAS86NL	PASCAL-86	Software Package for IBM PC XT/AT running PC DOS 3.0 or higher
VVSPAS86	PASCAL-86	Software Package for VAX/VMS
MVVPAS86	PASCAL-86	Software Package for MicroVAX/VMS
R86PAS86SU	PASCAL-86	Software Package for Intel System 86/3XX running iRMX 86
D86EDNL		AEDIT Source Code Editor for IBM PC XT/AT running PC DOS 3.0 or higher

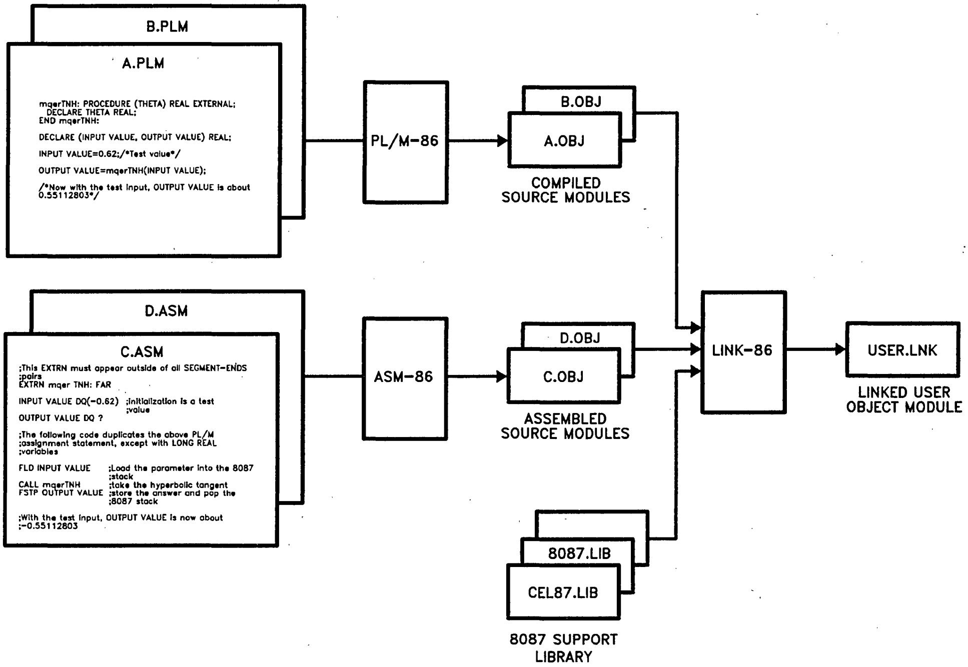
ICE is a trademark and iRMX a registered trademark of Intel Corporation. VAX and VMS are registered trademarks of Digital Equipment Corporation.

8087 SUPPORT LIBRARY

- **Library to Support Floating Point Arithmetic in Pascal-86, PL/M-86, FORTRAN-86, ASM-86, and iC-86**
- **Decimal Conversion Library Supports Binary-Decimal Conversions**
- **Supports Proposed IEEE Floating Point Draft 8.0 Standard for High Accuracy and Software Portability**
- **Common Elementary Function Library Provides Trigonometric, Logarithmic and Other Useful Functions**
- **Error-Handler Module Simplifies Floating Point Error Recovery**

The 8087 Support Library provides iC-86, Pascal-86, FORTRAN-86, PL/M-86 and ASM-86 users with numeric data processing capability. With the Library, it is easy for programs to do floating point arithmetic. Programs can bind in library modules to do trigonometric, logarithmic and other numeric functions, and the user is guaranteed accurate, reliable results for all appropriate inputs. Figure 1 below illustrates how the 8087 Support Library can be bound with PL/M-86 and ASM-86 user code to do this. The 8087 Support Library supports Draft 8.0 of the IEEE Floating Point Standard page 754. Consequently, by using this Library, the user saves software development time and the software investment is maintained.

The 8087 Support Library consists of the common elementary function library (CEL87.LIB), the decimal conversion library (DC87.LIB), the emulator interface library E8087.LIB, the error handler module (EH87.LIB) and interface libraries (8087.LIB, NUL87.LIB).



```
mqerTNH: PROCEDURE (THETA) REAL EXTERNAL;
  DECLARE THETA REAL;
END mqerTNH:

DECLARE (INPUT VALUE, OUTPUT VALUE) REAL;

INPUT VALUE=0.62;/*Test value*/

OUTPUT VALUE=mqerTNH(INPUT VALUE);

/*Now with the test input, OUTPUT VALUE is about
0.55112803*/
```

```
;This EXTRN must appear outside of all SEGMENT-ENDS
;pairs
EXTRN mqer TNH: FAR

INPUT VALUE DQ(-0.62)  ;initialization is a test
                       ;value
OUTPUT VALUE DQ ?

;The following code duplicates the above PL/M
;assignment statement, except with LONG REAL
;variables

FLD INPUT VALUE     ;Load the parameter into the 8087
                    ;stack
CALL mqerTNH        ;take the hyperbolic tangent
FSTP OUTPUT VALUE   ;store the answer and pop the
                    ;8087 stack

;With the test input, OUTPUT VALUE is now about
;-0.55112803
```

231613–1

Figure 1. Use of 8087 Support Library with PL/M-86 and ASM-86

CEL87.LIB
THE COMMON ELEMENTARY FUNCTION LIBRARY

FUNCTIONS

CEL87.LIB contains commonly used floating point functions. It is used along with the 8087 numeric coprocessor. It provides a complete package of elementary functions, giving valid results for all appropriate inputs. Following is a summary of CEL87 functions, grouped by functionality.

Rounding and Truncation Functions:

mqerIEX, mqerIE2, and mqerIE4. Round a real number to the nearest integer; to the even integer if there is a tie. The answer returned is real, a 16-bit integer or a 32-bit integer respectively.

mqerIAX, mqerIA2, mqerIA4. Round a real number to the nearest integer, to the integer away from zero if there is a tie; the answer returned is real, a 16-bit integer or a 32-bit integer, respectively.

mqerICX, mqerIC2, mqerIC4. Truncate the fractional part of a real input; the answer is real, a 16-bit integer or 32-bit integer, repectively.

Logarithmic and Exponential Functions:

mqerLGD computes decimal (base 10) logarithms.

mqerLGE computes natural base (base e) logarithms.

mqerEXP computes exponentials to the base e.

mqerY2X computes exponentials to any base.

mqerY12 raises an input real to a 16-bit integer power.

mqerY14 is as mqerY12, except to a 32-bit integer power.

mqerYIS is as mqerY12, but it accommodates PL/M-286 users.

Trigonometric and Hyperbolic Functions:

mqerSIN, mqerCOS, mqerTAN compute sine, cosine, and tangent.

mqerASN, mqerACS, mqerATN compute the corresponding inverse functions.

mqerSNH, mqerCSH, mqerTNH compute the corresponding hyperbolic functions.

mqerAT2 is a special version of the arc tangent function that accepts rectangular coordinate inputs.

Other Functions (of real variables):

mqerDIM is FORTRAN's positive difference function.

mqerMAX returns the maximum of two real inputs.

mqerMIN returns the minimum of two real inputs.

mqerSGH combines the sign of one input with the magnitude of the other input.

mqerMOD computes a modulus, retaining the sign of the dividend.

mqerRMD computes a modulus, giving the value closest to zero.

Complex Number Functions:

mqercCMUL, and mqercCDIV perform complex multiplication and division of complex numbers.

mqercCPOL converts complex numbers from rectangular to polar form. mqercCREC converts complex numbers from polar to rectangular form.

mqercCSQR, and mqercCABS compute the complex square root and real absolute value (magnitude) of a complex number.

mqercCEXP, and mqercCLGE compute the complex value of e raised to a complex power and the complex natural logarithm (base e) of a complex number.

mqercCSIN, mqercCCOS, and mqercCTAN compute the complex sine, cosine, and tangent of a complex number.

mqercCASN, mqercCACS, and mqercCATN compute the complex inverse sine, cosine, and tangent of a complex number.

mqercCSNH, mqercCCSH, and mqercCTNH compute the complex hyperbolic sine, cosine, and tangent of a complex number.

mqercCACH, mqercCASH, and mqercCATH compute the comples inverse hyperbolic sine, cosine, and tangent of a complex number.

mqercCC2C, mqercCR2C,mqercCC2R,mqercCCI2, mqercCCI4, and mqercCCIS return complex values of complex (or real) values raised to complex (real, short integer, or long integer) values.

DC87.LIB
THE DECIMAL CONVERSION LIBRARY

DC87.LIB is a library of procedures which convert binary representations of floating point numbers and ASCII-encoded string of digits.

The binary-to-decimal procedure mqcBIN_DECLOW accepts a binary number in any of the formats used for the representation of floating point numbers in the 8087. Because there are so many output formats for floating point numbers, mqcBIN_DECLOW does not attempt to provide a finished, formatted text string. Instead, it provides the "building blocks" for you to use to construct the output string which meets your exact format specification.

The decimal-to-binary procedure mqcDEC_BIN accepts a text string which consists of a decimal number with optional sign, decimal point, and/or power-of-ten exponent. It translates the string into the caller's choice of binary formats.

Decimal-to-binary procedure mqcDECLOW_BIN is provided for callers who have already broken the decimal number into its constituent parts.

The procedures mqcLONG_TEMP, mqcSHORT_TEMP, mqcTEMP_LONG, and mqcTEMP_SHORT convert floating point numbers between the longest binary format, TEMP_REAL, and the shorter formats.

EH87.LIB
THE ERROR HANDLER LIBRARY

EH87.LIB is a library of five utility procedures for writing trap handlers. Trap handlers are called when an unmasked 8087 error occurs.

The 8087 error reporting mechanism can be used not only to report error conditions, but also to let software implement IEEE draft standard options not directly supported by the chip. The three such extensions to the 8087 are: normalizing mode, non-trapping not-a-number (NaN), and non-ordered comparison. The utility procedures support these extra features.

DECODE is called near the beginning of the trap handler. It preserves the complete state of the 8087, and also identifies what function called the trap handler, and returns available arguments and/or results. DECODE eliminates much of the effort needed to determine what error caused the trap handler to be called.

NORMAL provides the "normalizing mode" capability for handling the "D" exception. By calling NORMAL in your trap handler, you eliminate the need to write code in your application program which tests for non-normal inputs.

SIEVE provides two capabilities for handling the "I" exception. It implements non-trapping NaN's and non-ordered comparisons. These two IEEE draft standard features are useful for diagnostic work.

ENCODE is called near the end of the trap handler. It restores the state of the 8087 saved by DECODE, and performs a choice of concluding actions, by either retrying the offending function or returning a specified result.

FILTER calls each of the above four procedures. If your error handler does nothing more than detect fatal errors and implement the features supported by SIEVE and NORMAL, then your interface to EH87.LIB can be accomplished with a single call to FILTER.

8087.LIB, NUL87.LIB, E8087.LIB INTERFACE LIBRARIES

E8087.LIB, 8087.LIB and NUL87.LIB libraries configure a user's application program for the run-time environment; running with the 8087 component or without floating point arithmetic, respectively.

FULL 8087 EMULATOR

The Full 8087 Emulator is a 16-kilobyte object module that is linked to the application program for floating-point operations. Its functionality is identical to the 8087 chip, and is ideal for prototyping and debugging floating-point applications. The Emulator is an alternative to the use of the 8087 chip, although the latter executes floating-point applications up to 100 times faster than an 8086 with the 8087 Emulator. Furthermore, since the 8087 is a "coprocessor," use of the chip will allow many operations to be performed in parallel with the 8086.

SPECIFICATIONS

Operating Environment

Intel Microcomputer Development Systems (Series III, Series IV)

Documentation Package

8087 Support Library Reference Manual

ORDERING INFORMATION

8087 Support Library is included in ASM-86 Assembler package on the following hosts.

Part Number	Description
D86ASM86NL	ASM-86 Assembler for PC XT or AT System (or compatible) running DOS 3.0 or higher.
VVSASM86	ASM-86 Assembler for VAX/VMS.
MVVSASM86	ASM-86 Assembler for Micro VAX/VMS.

Requires Software License

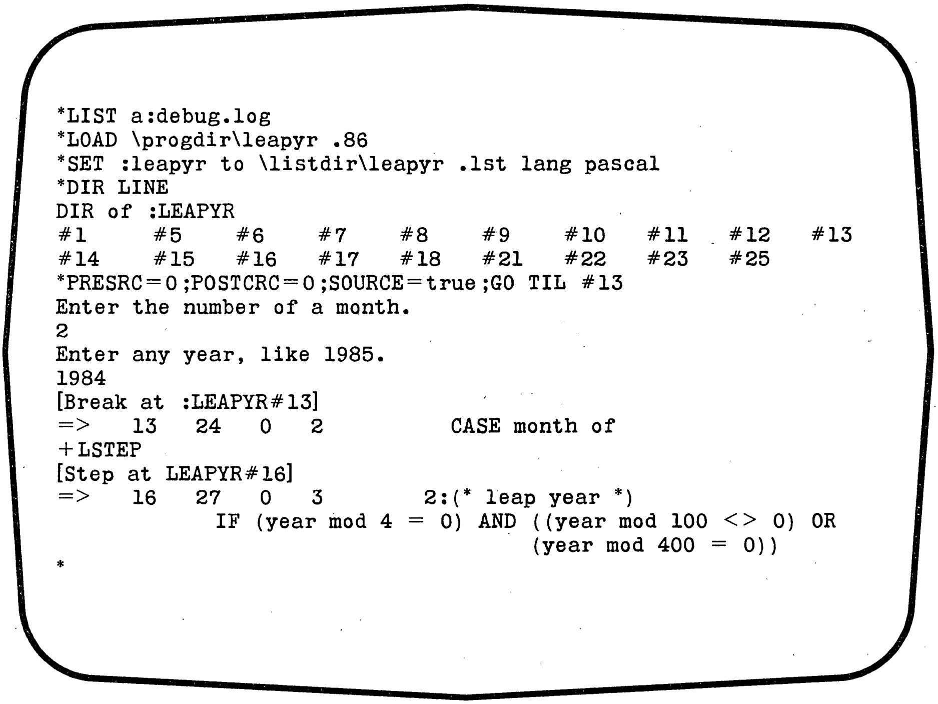
SUPPORT

Intel offers several levels of support for this product which are explained in detail in the price list. Please consult the price list for a description of the support options available.

PSCOPE-86 FOR DOS
HIGH-LEVEL APPLICATION PROGRAM DEBUGGER

- **Debugs PL/M-86, Pascal-86, iC-86, FORTRAN-86, and ASM86 Programs**
- **Displays Program Text on the Screen During Debugging:**
  - **—Uses the Listing File to Display Program Text**
  - **—Displays Source Code on Program Step, at Execution Break Points, or on User Request**
- **Disassembles Memory and Provides an Interactive Assembler**
- **Permits Creation of Program Patches Using High-level Language Constructs**
- **Supports Access to DOS Operating System Commands**
- **Offers Symbolic Debugging Capabilities:**
  - **—Supports Access to Memory by Program Defined Variable and Program Names**
  - **—Maintains Type Information About Variables**
  - **—Allows Definition of User-defined Debugging Variables and Procedures**
- **Single-steps Through Assembly Language Instructions, High-level Language Statements, or Procedures**
- **Sets Break Points and Traces Program Execution**
- **Runs Under the PC-DOS Version 3.0 or Greater**

PSCOPE-86 for DOS is an interactive, symbolic debugger for high-level language programs written in iC-86, PL/M-86, Pascal-86, and FORTRAN-86, and for assembly language programs written in ASM86. PSCOPE-86 for DOS runs under the PC-DOS operating system, version 3.0 or greater.



```
*LIST a:debug.log
*LOAD \progdir\leapyr .86
*SET :leapyr to \listdir\leapyr .lst lang pascal
*DIR LINE
DIR of :LEAPYR
#1     #5     #6     #7     #8     #9     #10    #11    #12    #13
#14    #15    #16    #17    #18    #21    #22    #23    #25
*PRESRC=0;POSTCRC=0;SOURCE=true;GO TIL #13
Enter the number of a month.
2
Enter any year, like 1985.
1984
[Break at :LEAPYR#13]
=>    13   24   0   2         CASE month of
+LSTEP
[Step at LEAPYR#16]
=>    16   27   0   3       2:(* leap year *)
             IF (year mod 4 = 0) AND ((year mod 100 <> 0) OR
                                      (year mod 400 = 0))
*
```

280194–1

MAJOR FEATURES

With PSCOPE-86 for DOS, a user can load an application program, set break points at symbolic or numeric addresses, trace program execution, and view source code text. Program bugs can be patched using high-level PSCOPE commands or assembly code. The corrections can be tested without leaving the PSCOPE software.

Other debugging aids include the ability to single-step a program through assembly language instructions, high-level-language statements, or procedures, to display and modify program variables, to inspect files, and to personalize the debugging environment.

The following sections describe some of the major features of PSCOPE-86 for DOS.

Source Display

With the DOS version of PSCOPE-86, a user can correlate a module under debug to a source code file. Then, when break points are encountered, source text is displayed along with the break message and line number of the break point. The number of source lines displayed before and after a break point can also be defined by the user.

View all or part of the listing file on command. The following example uses the PSCOPE command to list the current module. The asterisk (*) is the PSCOPE prompt, the command follows, and after pressing <Enter>, PSCOPE responds with a list file.

```
*SHOWSRC #1 LENGTH 28
 1   1   0   0  program leapyr (input,output);
                (* Input month and year, receive number of days *)
 2   5   0   0  var year    :integer;
 3   6   0   0      month   :integer;
 4   7   0   0      nrdays  :integer;

 5   9   0   0  begin

 5  11   0   1     month := 0;
 6  12   0   1      year := 0;
 7  13   0   1    nrdays := 0;

 8  15   0   1  writeln('Enter the number of a month.');
 9  16   0   1  readln(month);
10  17   0   1    while month <> 999 do

11  19   0   1    begin

11  21   0   2  writeln('Enter any year, like 1985.');
12  22   0   2  readln(year);

13  24   0   2      CASE month of
14  25   0   3                  4,6,9,11:nrdays := 30;
15  26   0   3            1,3,5,7,8,10,12:nrdays := 31;
16  27   0   3                          2: (* leap year *)
                    IF (year mod 4 = 0) AND ((year mod 100 <> 0) OR
                                             (year mod 400 = 0))
                                   THEN nrdays := 29
17  31   0   3                     ELSE nrdays := 28;

19  33   0   3      end;

21  35   0   2    writeln('Number of days in the month is',nrdays);

22  37   0   2    writeln('Enter the number of a month.');
23  38   0   2    readln(month)
                 end;
25  40   0   1  end.
```

Single-Stepping

PSCOPE has two commands to single-step through high level instructions and display source code. The commands differ in how they handle program calls. The following example illustrates the LSTEP command.

```
*LSTEP
[Step at :LEAPYR#17]
=>   17  31  0  3                     ELSE nrdays := 28;
     19  33  0  3              end;
*LSTEP
[Step at :LEAPYR#21]
=>   21  35  0  2        writeln('The number of days in the month is',nrdays);
```

PSCOPE can single-step through code at assembly level and display assembly mnemonics as in the following example which uses the ISTEP command.

```
*ISTEP
:LEAPYR
512A:00FEH       C70600000000        MOV WORD PTR 0000H,0
```

Symbolic Debugging

With symbolic debugging, a user can examine or modify a memory location by using its symbolic reference. A symbolic reference is a procedure name, variable name, line number, or program label that corresponds to a location in the user program's memory space. For example, to display the value of the program variables, users need only execute the program until the variable is active and type that variable's name.

```
*LSTEP
[Step at :LEAPYR#22]
=>   22  37  0  2         writeln('Enter the number of a month.');
*month
+2
*year
+1900
*nrdays
+28
```

Define the Debug Environment

With the PSCOPE high-level program debugger, a user can define the debugging environment within PSCOPE software. You can define break points and trace points. With PSCOPE, you can write macros that set the debug environment when PSCOPE is invoked, or these macros can be included at any time during the debugging session. Shorten commands with literal definitions, try program bug fixes with patches and procedures, or write procedures to control program execution. All debug variables and procedures can be saved in files and reused.

BREAK REGISTERS AND TRACE REGISTERS

Breaks occur at addresses in the program under execution. The user can enter physical addresses or symbolic addresses to halt program execution. With PSCOPE, you can easily break at executable statement addresses by using line numbers. Simply use the PSCOPE directory command with the line option (DIR LINE) to get a directory of line numbers. Then define a break register or a trace register to stop at these addresses.

A break register (BRKREG) stops program execution and returns a PSCOPE prompt (*). A trace register (TRCREG) displays a message and continues program execution. Following are examples of how to define a break register and a trace register.

```
*DEFINE BRKREG stop = #22
*DEFINE TRCREG stop2 = #17
```

DEBUGGING PROCEDURES

Debugging procedures are groups of PSCOPE commands that have been labeled. Writing procedures with PSCOPE commands is much the same as writing high-level language procedures. A procedure can be used for any definable function during a debugging session, and it can be used with a program under execution.

In the LEAPYR program, the while loop continues until 999 is entered for a month number. The following example of a PSCOPE procedure (PROC) that querys the user about halting execution. If the answer is yes ('Y' or 'y'), the procedure sets *month* to 999.

```
*DEFINE PROC query = DO
.*WRITE USING ('Do you want to quit? Enter Y for yes.')
.*DEFINE CHAR ccc = CI
.*WRITE ccc
.*IF ccc =='Y' or 'y' then
..*month == 999
..*RETURN = true
..*else RETURN = false
.*endif
.*END
*
```

To call this procedure while the program is executing, define a break register and use it with the GO command as follows:

```
*DEFINE BRKREG stop = #22 CALL query
*GO USING stop
```

PSCOPE PATCHES

A PSCOPE patch is used to temporarily correct run-time errors in the program under debug. A patch can be an additional line (or lines) in a program, or can be used to replace lines in a program. PSCOPE enables both high-level patches (the PATCH command) and assembly-level patches (the ASM command).

High-Level Patch

In the LEAPYR program, the way to exit the program is to enter 999 for the month. However, nothing instructs the user to do this. With a high-level patch, it is simple to add a line of code to the program. Following is an example.

```
*DEFINE PATCH #22 = WRITE 'To exit the program, enter 999.'
```

When the program is executed, the patch is used automatically. There is no need for a break register. Program execution stops at line number 22, the patch message is displayed, and program execution continues at line number 22. It is also possible to replace lines by using the TIL option in a high-level patch. Then program execution continues from the line number, or address, defined after the TIL. To simply eliminate lines of code, set the line to NOP as follows:

```
*DEFINE PATCH #18 = NOP
```

Assembly-Level Patch

Assume there is a typo in the LEAPYR program. Instead of the else condition setting nrdays to 28, it sets nrdays to 29, making every year leap year. Use the ASM command first to display assembly code as in the following example.

```
*ASM #17 LENGTH 4
:LEAPYR
521A:01E0H     C70600001D00       MOV  WORD PTR 0000H,001DH   ;+29T
521A:01E6H     EB06               JMP    $+0008H          ; A=01EEH
=>  17   31   0 3                    ELSE nrdays := 29;
521A:01E8H      C70600001D00      MOV  WORD PTR 0000H,001DH   ;+29T
521A:01EEH EB00 JMP $+0002H              ; A=01F0H
```

Notice that source code can be displayed to assist you in finding the ELSE statement. However, source display can be eliminated simply by setting the variable SOURCE to false. After finding the address for the correct line of code, use the ASM command to change the second 29 to 28. Notice in the following example, 'word' is sufficient for the assembly mnemonic. The 'ptr' mnemonic is unnecessary.

```
*ASM 521A:01E8H = 'mov word 0000H,001Ch'
521A:01E8H     C70600001C00       MOV  WORD 0000H,001DH
```

LITERALLY DEFINITIONS

LITERALLY definitions are shortened names for previously defined character strings. LITERALLY definitions save keystrokes or improve clarity. For example, the following LITERALLY definition replaces the command DEFINE with the abbreviation DEF.

```
*DEFINE LITERALLY def = 'DEFINE'
```

Save and Restore the Debug Environment

All debug variables and procedures can be saved in a file for future debug sessions. To save everything in a file, use the PUT command as follows:

```
*PUT a:debug.mac DEBUG
```

The saved file can be used as a macro and invoked automatically with PSCOPE by using the following invocation command to start PSCOPE.

```
C:>PSCOPE MACRO(a:debug.mac)
```

After PSCOPE is loaded, a list of all the commands in the macro will print to the screen and will be included in the debug environment. It is also possible to include a macro after PSCOPE is loaded. The following example uses the NOLIST option to prevent the commands from writing to the screen.

```
*INCLUDE a:debug.mac NOLIST
```

The Internal Editor

PSCOPE has an internal editor that is a version of Intel's Aedit. Use this editor to correct source code as program fixes are confirmed with PSCOPE. The editor can also be used to create macros, procedures, or correct command lines.

Escape to DOS

PSCOPE has an escape function to enable access to the DOS operating system commands. This is very useful to verify a file location or print a file. Any DOS operating system command is accepted after entering the 'bang', explanation point, (!). The following is an example of the ESCAPE command.

```
* !print a:debug.mac
```

The DOS print message will appear on the screen, and then the PSCOPE prompt. Once the printing is complete, you are again in PSCOPE withoutaltering the debug environment.

The PSCOPE Command Language

The syntax of PSCOPE commands resembles that of a high-level language. The PSCOPE command language is versatile and powerful while remaining easy to learn and use because commands are often self explanatory like GO. GO starts execution of the user program.

The PSCOPE command language can be divided into functional categories.

- Emulation commands instruct PSCOPE to execute the user program. They consist of GO and the three stepping commands, ISTEP, LSTEP, and PSTEP.
- Debugging environment commands define PATCHes, debugging PROCedures, debugging *variables*, LITERALLYs, break registers (BRKREG), and trace registers (TRCREG) using the DEFINE command. A user can also delete these definitions with the REMOVE command.
- Block commands consist of DO-END, COUNT-END, REPEAT-END, and IF-THEN-ELSE constructs. They can be used alone or within debugging procedures and patches.
- String functions concatenate strings (CONCAT), return the string length (STRLEN), return a substring (SUBSTR), and accept console input (CI).
- Utility commands are general-purpose commands for use in a debugging environment. They consist of the following:

  ! accesses the DOS operating system commands.

  $ is a pseudo-variable that represents the current execution point.

  ACTIVE is a function that determines whether a specified dynamic variable is currently defined on the stack.

  ASM assembles or disassembles memory.

  BASE sets or displays the current radix.

  CALLSTACK displays the dynamic calling sequence stored on the stack.

  DIR displays all objects of a specified type.

  EDIT invokes the internal, menu-driven text editor.

  EVAL returns the value of a symbol in binary, decimal, hexadecimal, and ASCII.

  EXIT returns control to the host operating system.

  HELP provides on-line help for selected topics and selected error messages.

  NAMESCOPE is pseudo-variable that represents the current scope of a variable. It gives access to variables without requiring a fully qualified symbolic reference.

  OFFSET$OF is a function that returns the offset of a specified address (virtual or symbolic).

  SELECTOR$OF is a function that returns the selector of a specified address (virtual or symbolic).

  WRITE writes variables and strings to the console's screen.
- File handling commands access disk files. The user can load program files to be debugged (LOAD), save patches, debugging procedures, debugging variables, LITERALLYs, and debugging registers in a disk file (PUT and APPEND), read-in these definitions during later debugging sessions (INCLUDE), and record a debugging session in a disk file for later analysis (LIST and NOLIST).
- Register access commands provide access to the 8086/8088 registers and flags.

  The REGS command displays the 8086/8088 registers and flags. Users can also inspect or change an individual register by specifying its mnemonic. For example, CS represents the code segment register.

  The FLAG pseudo-variable represents the 8086/8088 flag word. The user can also inspect or change each flag separately as a Boolean variable. (For example, TFL represents the trap flag).

  PSCOPE provides register access for programs that perform real arithmetic. There is a built-in 8087 math coprocessor emulator, or there is a CH8087 option with the LOAD command to tell PSCOPE to access the hardware (8087 math coprocessor chip) registers. Access or change the 8087 registers by name.
- Source display commands are used to view a specified number of lines of source text at break points or on demand. LPATH or SET directs PSCOPE to the source text file. SOURCE is the pseudo-variable used to determine if source text will be displayed at break points. With PRESRC and POSTSRC, the user can determine how many lines of source code will be displayed before and after the line at the break point. The SHOWSRC command enables the display of source code outside of program execution.

SPECIFICATIONS

Memory Requirements

PSCOPE-86 for DOS requires approximately 300KB of memory for PSCOPE software and buffers.

DOS Version

PSCOPE is designed to run on the DOS operating system version 3.0 or greater.

Language Support

iC-86

PL/M-86

FORTRAN-86

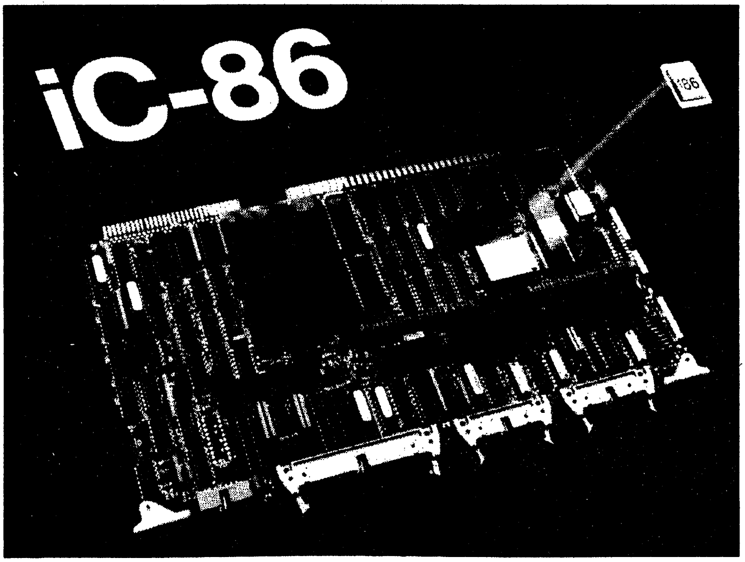
ASM86

PASCAL-86

ORDERING INFORMATION

Order Code	Description
D86PSC86NL	High-Level Software Debugger

INTEL iC-86 C COMPILER

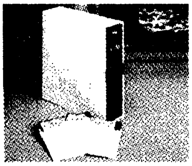


INTEL iC-86 R4.0 COMPILER

Intel's iC-86 R4.0 is a new generation C compiler for the 8086/186 family of microprocessors, providing unparalleled performance for embedded microprocessor designs. In addition to outstanding execution speed, Intel's iC-86 also offers low memory consumption, ROMability, and easy debug.

iC-86 R4.0 COMPILER FEATURES

- State-of-the-art code generation technology
- Built-in functions for automatic machine code generation
- ROMable code and libraries
- Outstanding optimization
- Integrated debugging with Intel ICE™ and I²ICE™
- Compliance with draft ANSI standard
- Supports multiple memory models: Small, Medium, Compact, and Large
- Linkable with other Intel 8086 languages such as ASM-86 and PL/M-86
- ROMable and reentrant libraries
- Ability to mix memory models with "near" and "far" pointers
- Compatible with other C compilers and PL/M providing both standard C and PL/M calling conventions



ICE and I²ICE are trademarks of Intel Corporation.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel.

February, 1988
Order Number: 280787-003

BUILT-IN FUNCTIONS

iC-86 R4.0 is loaded with built-in functions that directly generate machine code within the C language. iC-86's built-ins eliminate the need for in-line assembly language programming by allowing you to program in a high-level language all the time, speeding software development and simplifying software maintenance. The built-ins also allow you to develop highly optimized code by extending the compiler instruction set—with built-ins you can enable or disable interrupts and directly control hardware I/O without having to exit C for assembler. This means you can write high performance software for real time applications without having to keep track of every architectural detail, as you would in assembly language. For example, to generate an INT instruction, you simply type:

```
causeinterrupt (number)
```

Or, the following iC-86 instruction will cause the processor to come to a halt with interrupts enabled:

```
halt( )
```

EMBEDDED COMPONENT SUPPORT

iC-86 was designed specifically for embedded microprocessor applications. iC-86 produces ROMable code which can be loaded directly into target systems via Intel ICE emulators and debugged *without modification* for fast, easy, development and debugging.

HIGHLY OPTIMIZED

iC-86 is based on Intel's latest code generation techniques for developing high-performance applications, and has been optimized for developing embedded applications. Four levels of optimization are available. Important optimization features include a jump optimizer and improved register manipulation via register history. In addition, the PL/M calling convention will improve performance significantly. An example of the optimization in iC-86 R4.0 is its outstanding performance on the Dhrystone benchmark. Using a Compaq 386, iC-86 produced the following results:

	Microsoft 4.0	Microsoft 5.0	Intel iC-86
Execution Speed (dhry/sec)	3333	3369	3571

RUN-TIME SUPPORT

STDIO run-time libraries for iC-86 are targeted to a generic POSIX interface, with documentation provided for interfacing with your embedded target system. This means you can easily retarget the libraries for use in your target application, regardless of operating system. These libraries support the complete draft ANSI standard. For iC-86 versions on DOS, Intel provides the interface between the STDIO libraries and the DOS operating system. This allows you to develop, test, and debug your embedded application code on DOS, or write applications directly for DOS.

INTEGRATED DEBUG TOOLS

iC-86 has been designed to work with Intel's ICE family of in-circuit emulators (I²ICE, ICE-186, ICE-286, and ICE-386) and performance analysis tools (iPAT), Intel software debuggers, linkers, locators, and other software development tools. In addition to the object records required for program execution, iC-86 object code contains detailed debug records that describe the actual symbols and variable names you defined in your source code. A complete listing file can also be produced. Intel's "integration by design" of all development tools, including iC-86, will speed the development of your embedded microprocessor applications. Figure 1 illustrates the steps in going from C source code to PROM- or ICE-loadable object code with iC-86.

SERVICE AND SUPPORT

Intel's development tools are backed by our worldwide service and support organization, which is set up to deal with problems encountered by embedded component designers. Our field application experts get you up and running quickly, and our hands-on training workshops ensure that you have a thorough understanding of how our tools work. Intel compilers come with 90 days of technical support, troubleshooting guides, application newsletters, and optional support contracts.

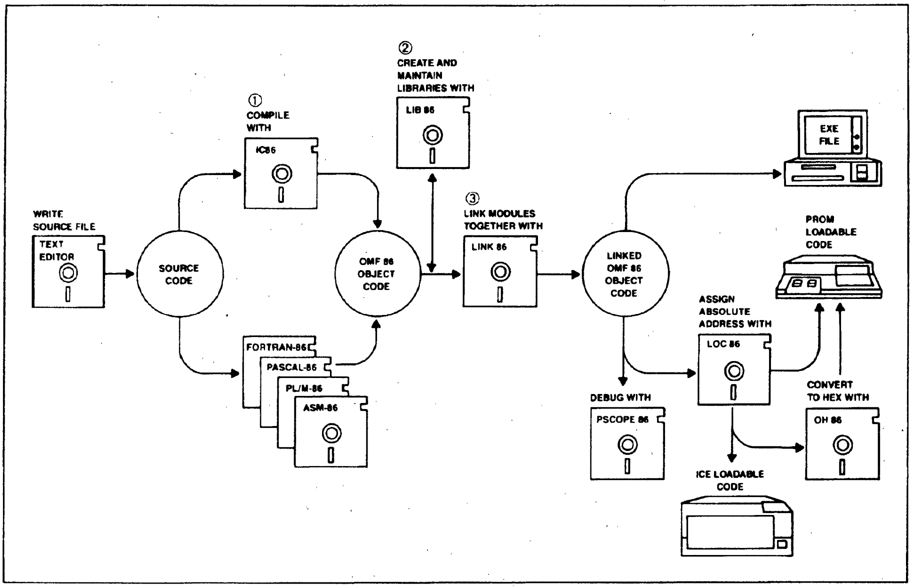


Figure 1. The Application Development Process

SPECIFICATIONS

ENVIRONMENT

Hardware requirements	IBM PC XT or AT (or 100% compatible) running DOS 3.0 or greater
Memory requirements:	512K Bytes of RAM, hard disk strongly recommended
Media:	5¼" DS/DD DOS diskettes

STANDARDS

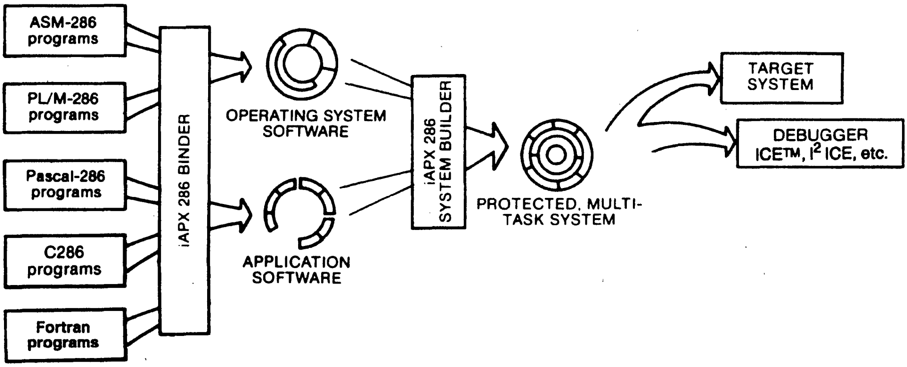
iC-86 complies with the 1986 X3J11 ANSI draft proposal for the C programming language

ORDERING INFORMATION

Order Code	Description
D86C86NL	DOS hosted iC-86

286 SOFTWARE DEVELOPMENT TOOLS AVAILABLE ON CHOICE OF INDUSTRY STANDARD HOSTS INCLUDING PC-DOS AND VAX/VMS*

- **286 Software Development Package**
  - **—Complete System Development Capability for High-Performance 286 Applications**
  - **—Allows Creation of Multi-User, Virtual Memory, and Memory-Protected Systems**
  - **—Macro Assembler for Machine-Level Programming**
- **Pascal-286 Software Package**
  - **—High-Level Programming Language for the Protected Virtual Mode of the 286**
  - **—Implements ISO Standard Pascal**
- **PL/M 286 Software Package**
  - **—Systems Programming Language for the Protected Virtual Address Mode of the 286**
  - **—Advanced Structured System Implementation Language for Algorithm Development**
- **ic-286, C Compiler for the 80286**
  - **—Implements Full C Language**
  - **—High-Level Language for the Protected Virtual Mode of the 286**
- **FORTRAN 286 Software Package**
  - **—Features High-Level Language Support for Floating Point Calculations**
  - **—Meets ANSI FORTRAN 77 Subset Language Specifications**



231665–1

The iAPX 286 Software Development Package keeps the protection mechanism invisible to the application programmer, yet easy to configure for the system programmer.

*VAX/VMS are trademarks of Digital Equipment Corporation.

80286 MACRO ASSEMBLER

- **Instruction Set and Assembler Mnemonics Are Upward Compatible with ASM-86/88**
- **Powerful and Flexible Text Macro Facility**
- **Type-Checking at Assembly Time Helps Reduce Errors at Run-Time**
- **Structures and RECORDS Provide Powerful Data Representation**
- **"High-Level" Assembler Mnemonics Simplify the Language**
- **Supports Full Instruction Set of the 80286/20, Including Memory Protection and Numerics**

ASM-286 is the "high-level" macro assembler for the 80286 assembly language. ASM-286 translates symbolic assembly language mnemonics into relocatable object code. The assembler mnemonics are a superset of ASM-86/88 mnemonics; new ones have also been added to support the new 80286 instructions. The segmentation directives have been greatly simplified.

The 80286 assembly language includes approximately 150 instruction mnemonics. From these few mnemonics the assembler can generate over 4,000 distinct machine instructions. Therefore, the software development task is simplified, as the programmer need know only 150 mnemonics to generate all possible machine instructions. ASM-286 will generate the shortest machine instruction possible (given explicit information as to the characteristics of any forward referenced symbols).

The powerful macro facility in ASM-286 saves development and maintenance time by coding common program sequences only once. A macro substitution is made each time the sequence is to be used. This facility also allows for conditional assembly of certain program sequences.

ASM-286 offers many features normally found only in high-level languages. The assembly language is strongly typed, which means it performs extensive checks on the usage of variables and labels. This allows many programming errors to be detected when the program is asembled, long before it is being debugged.

ASM-286 object modules conform to a thorough, well-defined format used by all 286 high-level languages and utilities. This makes it easy to call (and be called from) HLL object modules.

Key Benefit

For programmers who wish to use assembly language. ASM-286 provides many powerful "high-level" capabilities that simplify program development and maintenance.

80286 BINDER

- **Links Separately Compiled Program Modules Into an Executable Task**
- **Makes the 80286 Protection Mechanism Invisible to Application Programmers**
- **Works with PL/M-286, Pascal-286, FORTRAN-286, ASM-286 Object Modules and IC-286**
- **Performs Incremental Linking with Output of Binder and Builder**
- **Resolves PUBLIC/EXTERNAL Code and Data References, and Performs Intermodule Type-Checking**
- **Provides Print File Showing Segment Map, Errors and Warnings**
- **Assigns Virtual Addresses to Tasks in the 2^{32} Address Space**
- **Generates Linkable or Loadable Module for Debugging**

BND-286 is a utility that combines 80286 object modules into executable tasks. In creating a task, the Binder resolves Public and External symbol references, combines segments, and performs address fix-ups on symbolic code and data.

The Binder takes object modules written in ASM-286, PL/M-286, Pascal-286, FORTRAN-286 or iC-286 and generates a loadable module (for execution or debugging), or a linkable module (to be re-input to the Binder later; this is called incremental binding). The binder accepts library modules as well, linking only those modules required to resolve external references. BND-286 generates a print file displaying a segment map, and error messages.

The Binder is used by system programmers and application programmers. Since application programmers need to develop software independent of any system architecture, the 286 memory protection mechanism is "hidden" from users of the Binder. This allows application tasks to be fully debugged before becoming part of a protected system. (A protected system may be debugged as well.) System protection features are specified later in the development cycle, using the 286 System Builder. It is possible to link operating system services required by a task using either the Binder or the Builder. This flexibility adds to the ease of use of the 286 utilities.

Key Benefits

The Binder is the only utility an application programmer needs to develop and debug an individual task. Users of the Binder need not be concerned with the architecture of the target machine, making application program development for the 286 very simple.

80286 MAPPER

- **Flexible Utility to Display Object File Information**
- **MAP-286 Selectively Purges Symbols from a Load Module**
- **Provides Inter-Module Cross-Referencing for Modules Written in All Languages**
- **Mapper Allows Users to Display:**
  Protection Information:
    - **Segment Tables**
    - **Gate Tables**
    - **Public Addresses**

  Debug Information:
    - **Module Names**
    - **Program Symbols**
    - **Line Numbers**

Key Benefit

A cross-reference map showing references *between* modules simplifies debugging; the map also lists and controls all symbolic information in one easy-to-read place.

80286 LIBRARIAN

- **Fast, Easy Management of 80286 Object Module Libraries**
- **Only Required Modules Are Linked, When Using the Binder or Builder**
- **Librarian Allows User to:**
  Create Libraries
  Add Modules
  Replace Modules
  Delete Modules
  Copy Modules from Another Library
  Save Library Module to Object File
  Create Backup
  Display Module Information (Creation Date, Public, Segments)

Key Benefit

Program libraries improve management of program modules, and reduce software administrative overhead.

80286 SYSTEM BUILDER

- **Supports Complete Creation of Protected, Multi-Task Systems**
- **Resolves PUBLIC/EXTERNAL Definitions (Between Protection Levels)**
- **Supports Memory Protection by Building System Tables, Initializing Tasks, and Assigning Protection Rights to Segments**
- **Creates a Memory Image of a 286 System for Cold-Start Execution**
- **Target System may be Boot-Loadable, Programmed into ROM, or Loaded From Mass-Store**
- **Generates Print File with Command Listing and System Map**

BLD-286 is the utility that lets system programmers configure mutli-tasking, protected systems from an operating system and discrete tasks. The Builder generates a cold-start execution module, suitable for ROM-based or disk-based systems.

The Builder accepts input modules from 80286 translators or the 80286 Binder. It also accepts a "Build File" containing definitions and initial values for the 286 protection mechanism—descriptor tables, gates, segments, and tasks. BLD-286 generates a Loadable or bootloadable output module, as well as a print file with a detailed map of the memory-protected system.

Using the Builder command Language, system programmers may perfrom the following functions:

— Assign physical addresses to segments; also set segment access rights and limits.

— Create Call, Trap, and Interrupt "Gates" (entry-points) for inter-level program transfers.

— Make gates available to tasks; this is an easier way to define program interfaces than using interface libraries.

— Create Global (GDT), Interrupt (IDT), and any Local (LDT) Descriptor Tables.

— Create Task State Segments and Task Gates for multi-task applications.

— Resolve inter-module and inter-level references, and perform type-checking.

— Automatically select required modules from libraries.

— Configure the memory image into partitions in the address space.

— Selectively generate an object file and various sections of the print file.

Key Benefit

Allows a system programmer to define the configuration of a protected system in *one* place, with one easy-to-use Utility. This specification may then be adopted by all project members, using either the Builder *or just the Binder*. The flexibility simplifies program development for all users.

SPECIFICATIONS

Documentation

ASM 286 Language Reference Manual
ASM 286 Macro Assembler Operating Instructions
80286 Utilities User's Guide
80286 System Builder User's Guide
Pocket Reference for all the above:
ASM 286
Utilties

SUPPORT AVAILABLE

Hotline Telephone Support, Software Updates, Technical Reports

ORDERING INFORMATION

Product Code	Operating Environment
D86 ASM 286	IBM PC XT/AT running PCDOS 3.0 or later
VVS ASM 286	VAX/VMS
MVVS ASM 286	MicroVAX/VMS
X286 ASM 286	Xenix for Intel 286/3XX Systems
R286 ASM 286	RMX 286 for Intel 286/3XX Systems

PASCAL-286 SOFTWARE PACKAGE

- **High-Level Programming Language for the Protected Virtual Mode iAPX 286**
- **Implements ISO Standard Pascal Many Useful Extensions may be Enabled via a Compiler Switch**
- **Choice of Industry Standard Hosts**
- **Supports Full Symbolic Debugging with iAPX 286 Software and ICE™ Debuggers**
- **Upward Compatible with Pascal-86 for Software Portability**
- **Produces Relocatable Object Code Which is Linkable to Object Modules Generated by Other iAPX 286 Translators**
- **Fully Supports the 80287 Numeric Processor using the IEEE Floating Point Standard**

Pascal-286 is a powerful, structured, applications programming language for the protected virtual address mode of the iAPX 286. Pascal-286 is upward compatible with Pascal-86 so that 8086 Pascal source code can be ported to the iAPX 286 in protected mode.

Pascal-286 implements strict ISO standard Pascal, but with many useful extensions. These include separate compilation of modules, interrupt handling, port I/O, and 80287 numerics support. A control is provided in the compiler to flag all non-ISO features used.

Pascal-286 produces relocatable object code which can be linked with object code produced by other iAPX 286 translators such as ASM-286 and PL/M-286. Thus, a combination of translators can be used to provide great programming flexibility.

Type and symbol information needed by software and in-circuit debuggers is added to the object code by the Pascal-286 compiler. This information can be stripped off by the compiler or linker for the final production version.

The Pascal-286 compiler runs on the IBM PC XT/AT running PCDOS version 3.0 or later and Digital Equipment VAX/VMS systems.

231665-2

FEATURES

Conforms to ISO Standard Pascal

Pascal has gained wide acceptance as a portable language for microcomputer applications. However, portability can result only if standards are adhered to. Pascal-286 is a strict implementation of ISO standard Pascal. Extensions are provided to make the language more powerful for microprocessor applications. All extensions are clearly highlighted in the documentation. In addition, the compiler provides a control to flag any non ISO feature used. Pascal-286 will evolve to track future enhancements to standard Pascal.

Upward Compatible with Pascal-86

The Pascal-286 compiler produces object code for the protected virtual address mode of the iAPX 286 language. However, no 286 architecture specific features have been added to the Pascal-286 language. This makes Pascal-286 source code upward compatible with Pascal-86, which allows for porting of 8086 software to the protected 286 with relative ease.

Compatible With Other iAPX 286 Translators

All Intel iAPX 286 translators output object code in a standardized format. This allows 286 programs to be written in a mixture of languages. Systems routines which need access to architectural features can be coded in PL/M-286 or ASM-286. Pascal-286 may be better suited for the applications routines. The systems and application routines can then be combined using the 286 linker (BIND-286).

Standardized Run Time Support

Programs compiled with Pascal-286 can be moved from the development host environment to the target environment with ease. This is the result of standardizing run-time operating system interfaces required by the compiled program into a well defined and well documented set of routines. After programs are developed on a development host, they can then be executed in the target using the same set of system interfaces.

Extensions for Microprocessor Programming

Pascal-286 provides extensions that make it powerful for microprocessor applications. Built-in procedures allow I/O directly from the ports of the iAPX 286. This speeds up I/O as it is done by direct communication with the microprocessor. Interrupt processing is also supported by built in procedures. Examples are: ENABLEINTERRUPTS, DISABLEINTERRUPTS, CAUSEINTERRUPT. Many built in procedures and variables are provided for communicating with the 80287 for numeric computations.

Compiler Controls

The Pascal-286 compiler provides many controls which can be used at invocation time to enhance programming flexibility. Examples are: CODE/NOCODE, DEBUG/NODEBUG, INCLUDE (file), LIST/NOLIST, OPTIMIZE (n), EXTENSIONS/NOEXTENSIONS. All controls have default values that are active unless the opposite is specified during invocation. Thus, for most compiles, no controls need be specified.

Support for IEEE Standard Numerics

Pascal-286 provides full support for the 80287 numerics co-processor. All floating point operations are done according to the IEEE floating point standard. The benefits are predictable, accurate and consistent results. Built-in procedures to support the 80287 include GET8087ERRORS and MASK 8087ERRORS. A full set of 80287 library routines are supplied with the compiler.

Optimizations

The Pascal-286 compiler produces highly optimized code, both in size and execution time. This is achieved by:

- Use of powerful iAPX 286 instructions, in particular, for string handling, 80287 numerics and subroutine linkage
- Short circuit evaluation of boolean expressions, constant folding and strength reduction of multiplications and additions
- Elimination of superfluous branches, optimization of span dependent jumps

Support Available

Hotline service, Software Updates and technical newsletters.

ORDERING INFORMATION

Product Code	Operating Environment	Documentation Package
D86 PAS 286	IBM PC XT/AT running PCDOS version 3.0 or later	Pascal-286 Pocket Reference
VVSPAS286	VAX/VMS	
MVVSPAS286	MicroVAX	

PL/M 286 SOFTWARE PACKAGE

- System Programming Language for the Protected Virtual Address Mode 80286
- Upward Compatible with PL/M 86 and PL/M 80 Assuring Software Portability
- Enhanced to Support Design of Protected, Multi-User, Multi-Tasking, Virtual Memory Operating System Software
- Multiple Levels of Optimization
- Advanced, Structured System Implementation Language for Algorithm Development
- Produces Relocatable Object Code Which is Linkable to Object Modules Generated by all Other 80286 Language Translators
- Wide Choice of Industry Standard Hosts

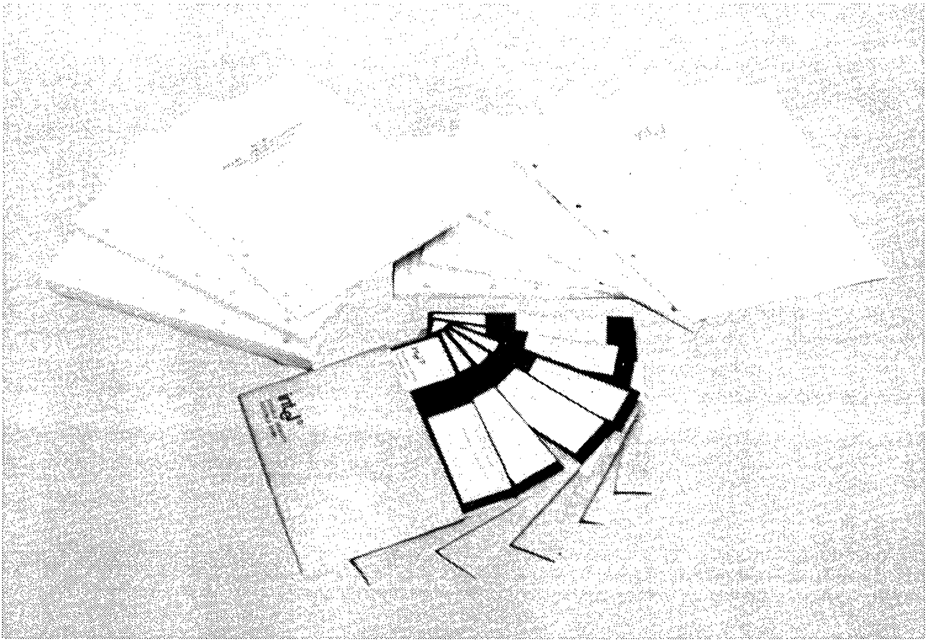
PL/M 286 is a powerful, structured, high-level system implementation language for the development of system software for the protected virtual address mode 80286. PL/M 286 has been enhanced to utilize 80286 features—memory management and protection—for the implementation of multi-user, multi-tasking virtual memory operating systems.

PL/M 286 is upward compatible with PL/M 86 and PL/M 80. Existing systems software can be recompiled with PL/M 286 to execute in protected virtual address mode on the 80286.

PL/M 286 is the high-level alternative to assembly language programming on the 80286. For the majority of 80286 systems programs, PL/M 286 provides the features needed to access and to control efficiently the underlying 80286 hardware and consequently it is the cost-effective approach to develop reliable, maintainable system software.

The PL/M 286 compiler has been designed to efficiently support all phases of software development features such as a built-in syntax checker, multiple levels of optimization, virtual symbol table and four models of program size and memory usage for efficient code generation provide the total program development support needed.

The PL/M 286 compiler runs on the IBM PC XT/AT running PC DOS version 3.0 or later, Digital Equipment VAX/VMS† Systems, and Intel XENIX** 286 and RMX 286 based systems.



231665-3

†VAX, VMS are trademarks of Digital Equipment Corporation.
**XENIX is a trademark of Microsoft Corporation.

FEATURES

Major features of the Intel PL/M 286 compiler and programming language include:

Structured Programming

PL/M source code is developed in a series of modules, procedures, and blocks. Encouraging program modularity in this manner makes programs more readable, and easier to maintain and debug. The language becomes more flexible by clearly defining the scope of user variables (local to a private procedure, for example).

The use of modules and procedures to break down a large problem leads to productive software development. The PL/M 286 implementation of block structure allows the use of REENTRANT procedures, which are especially useful in system design.

Language Compatibility

PL/M 286 object modules are compatible with object modules generated by all other 286 translators. This means that PL/M programs may be linked to programs written in any other 286 language.

Object modules are compatible with In-Circuit Emulators; DEBUG compiler control provides the In-Circuit Emulators with full symbolic debugging capabilities.

PL/M 286 language is upward compatible with PL/M 86 and PL/M 80 so that application programs may be easily ported to run on the protected mode 80286.

Supports Seven Data Types

PL/M makes use of seven data types for various applications. These data types range from one to four bytes and facilitate various arithmetic, logic, and addressing functions:

— Byte: 8-bit unsigned number

— Word: 16-bit unsigned number

— Dword: 32-bit unsigned number

— Integer: 16-bit signed number

— Real: 32-bit floating-point number

— Pointer: 16-bit or 32-bit memory address indicator

— Selector: 16-bit pointer base

Another powerful facility allows the use of BASED variable which permit run-time mapping of variables to memory locations. This is especially useful for passing parameters, relative and absolute addressing, and dynamic memory allocation.

Two Data Structuring Facilities

In addition to the seven data types and based variables, PL/M supports two powerful data structuring facilities. These help the user to organize data into logical groups.

— Array: Indexed list of same type data elements

— Structure: Named collection of same or different type data elements

— Combinations of both: Arrays of structures or structures of arrays

Numerics Support

PL/M programs that use 32-bit REAL data are executed using the 80287 Numeric Data Processor for high performance. All floating-point operations supported by PL/M are executed on the 80287 according to the IEEE floating-point standard. PL/M 286 programs can use built-in functions and predefined procedures—INIT$REAL$MATH$UNIT, SET$REAL$MODE, GET$REAL$ERROR, SAVE$REAL$STATUS, RESTORE$REAL$STATUS,—to control the operation of the 80287 within the scope of the language.

Built-In String Handling Facilities

The PL/M 286 language contains built-in functions for string manipulation. These byte and word functions perform the following operations on character strings: MOVE, COMPARE, TRANSLATE, SEARCH, SKIP, and SET.

Built-In Port I/O

PL/M 286 directly supports input and output from the 80286 ports for single BYTE and WORD transfers. For BLOCK transfers, PL/M 286 programs can make calls to predefined procedures.

Interrupt Handling

PL/M 286 has the facility for generating and handling interrupts on the 80286. A procedure may be defined as an interrupt handler through use of the INTERRUPT attribute. The compiler will then generate code to save and restore the processor status on each execution of the user-defined interrupt handler routine. The PL/M statement CAUSE$INTERRUPT allows the user to trigger a software interrupt from within the program.

Protection Model

PL/M 286 supports the implementation of protected operating system software by providing built-in procedures and variables to access the protection mechanism of the 80286. Predefined variables—TASK$REGISTER, LOCAL$TABLE, MACHINE$STATUS, etc.—allow direct access and modification of the protection system. Untyped procedures and functions—SAVE$GLOBAL$TABLE, RESTORE$GLOBAL$TABLE, SAVE$INTERRUPT$TABLE, RESTORE$INTERRUPT$TABLE, CLEAR$TASK$SWITCHED$FLAG, GET$ACCESS$RIGHTS, GET$SEGMENT$LIMIT, SEGMENT$READABLE, SEGMENT$WRITEABLE, ADJUST$RPL—provide all the facilities needed to implement efficient operating system software.

Compiler Controls

The PL/M 286 compiler offers controls that facilitate such features as:

- Optimization
- Conditional compilation
- The inclusion of additional PL/M source files from disk
- Cross-reference of symbols
- Optional assembly language code in the listing file
- The setting of overflow conditions for run-time handling

Addressing Control

The PL/M 286 compiler uses the SMALL, COMPACT, MEDIUM, and LARGE control to generate optimum addressing instructions for programs. Programs of any size can be easily modularized into "subsystems" to exploit the most efficient memory addressing schemes. This lowers total memory requirements and improves run-time execution of programs.

Code Optimization

The PL/M 286 compiler offers four levels of optimization for significantly reducing overall program size.

- Combination or "folding" of constant expressions; and short-circuit evaluation of Boolean expressions
- "Strength reductions": a shift left rather than multiply by 2; and elimination of common sub-expressions within the same block
- Machine code optimizations; elimination of superfluous branches; reuse of duplicate code; removal of unreachable code
- Optimization of based-variable operations and cross-statement load/store

Error Checking

The PL/M 286 compiler has a very powerful feature to speed up compilations. If a syntax or program error is detected, the compiler will skip the code generation and optimization passes. This usually yields a 2X performance increase for compilation of programs with errors.

A fully detailed and helpful set of programming and compilation error messages is provided by the compiler and user's guide.

BENEFITS

PL/M 286 is designed to be an efficient, cost-effective solution to the special requirements of protected mode 80286 Microsystem Software Development, as illustrated by the following benefits of PL/M use:

Low Learning Effort

PL/M 286 is easy to learn and use, even for the novice programmer.

Earlier Project Completion

Critical projects are completed much earlier than otherwise possible because PL/M 286, a structured high-level language, increases programmer productivity.

Lower Development Cost

Increases in programmer productivity translate immediately into lower software development costs because less programming resources are required for a given programmed function.

Increased Reliability

PL/M 286 is designed to aid in the development of reliable software (PL/M 286 programs are simple statements of the program algorithm). This substantially reduces the risk of costly correction of errors in systems that have already reached full production status, as the more simply stated the program is, the more likely it is to perform its intended function.

Easier Enhancements and Maintenance

Programs written in PL/M tend to be self-documenting, thus easier to read and understand. This means it is easier to enhance and maintain PL/M programs as the system capabilities expand and future products are developed.

Cost-Effective Alternative to Assembly Language

PL/M 286 programs are code efficient. PL/M 286 combines all of the benefits of a high-level language (ease of use, high productivity) with the ability to access the 80286 architecture. This includes language features for control of the 80286 protection mechanism. Consequently, for the development of systems software, PL/M 286 is the cost-effective alternative to assembly language programming.

SPECIFICATIONS

Support Available

90 Days:
Hotline Telephone Support, Software Updates, Subscription Service

Documentation Package

PL/M 286 User's Guide
PL/M 286 Pocket Reference

ORDERING INFORMATION

Ordering Code	Operating Environment
D86PLM286	IBM PC XT/AT running PCDOS version 3.0 or later
VVSASM286	VAX, VMS
X286PLM286	Xenix for Intel Systems 286/3XX
R286PLM286	iRMX™ 286 for Intel Systems 286/3XX

iC-286
C COMPILER FOR THE 80286

- **Implements Full C Language as Defined by the Draft ANSI Standard**
- **Produces High Density Code Rivaling Assembler**
- **Supports Intel Object Module Format (OMF)**
- **Available for VAX/VMS* PCDOS and iRMXII Operating Systems**
- **Supports Both Small, Medium, Compact and Large Models of Computation**
- **Supports ICE286 and I^{2}ICE™**
- **Supports IEEE Floating Point Math with Intel Math Coprocessor**
- **Supports I/O and Hardware Interrupts Directly in C**
- **Supports Full Standard I/O Library (STDIO)**
- **Written in C**
- **Supports Both Standard Intel (PL/M like) and Standard C Calling Conventions**

The C Programming Language was originally designed in 1972 and has become increasingly popular as a systems development language. C combines the flexibility and programming speed of a higher level language with the efficiency and control of assembly language.

Intel iC-286 brings the full power of the C programming language to 80286 based microprocessor systems.

Intel iC-286 supports the full C language as described in the Kernighan and Ritchie book, "The C Programming Language", (Prentice-Hall, 1978). iC286 implements the complete C language specification as defined in the ANSI Draft proposed X3JII Standard.

C is rapidly becoming the standard microprocessor system implementation language because it provides:

1. the ability to manipulate the fundamental objects of the machine (including machine addresses) as easily as assembly language.
2. the power and speed of a structured language supporting a large number of data types, storage classes, expressions and statements.
3. processor independence (most programs developed for other processors can be easily transported to the 80286), and
4. code that rivals assembly language in efficiency.

INTEL iC-286 COMPILER DESCRIPTION

The iC-286 compiler operates in four phases; preprocessor, parser, code generator, and optimizer. The preprocessor phase interprets directives in C source code, including conditional compilations (# define). The parser phase converts the C program into an intermediate free form and does all syntactic and semantic error checking. The code generator phase converts the parser's output into an efficient intermediate binary code, performs constant folding, and features an extremely efficient register allocator, ensuring high quality code. The optimizer phase converts the output of the code generator into relocatable Intel Object Module Format (OMF) code, without creating an intermediate assembly file. Optionally, the iC-286 compiler can produce a symbolic assembly like file. The iC-286 optimizer eliminates common code, eliminates redundant loads and stores, and resolves span dependencies (shortens branches) within a program.

The iC-286 runtime library consists of a number of functions which the C programmer can call. The runtime system includes the standard I/O library (STDIO), conversion routines, routines for manipulating strings, special routines to perform functions not available on the 80286 (32-bit arithmetic and emulated floating point), and (where appropriate) routines for interfacing with the operating system.

iC-286 uses Intel's linker and locator and generates debug records for symbols and lines on request, permitting access to Intel's ICE and I^{2}ICE to aid in program testing.

FEATURES

Support for Small and Large Models

Intel iC-286 supports both the SMALL and LARGE modes of segmentation. A SMALL model program can have up to 64K bytes of code and 64K bytes of data, with all pointers occupying two bytes. Because two byte pointers permit the generation of highly compact and efficient code, this model is recommended for programs that can meet the size restrictions. The LARGE segmentation model is used by programs that require access to the full addressing space of the 80286 processors. In this model, each source file generates a distinct pair of code and data segments of up to 64K bytes in length. All pointers are four bytes long.

Preprocessor Directives

#define—defines a macro
#include—includes code outside of the program source file
#if—conditionally includes or excludes code
Other preprocessor directives include #undef, #ifdef, #ifndef, #else, #endif, and #line.

Statements

The C language Supports a variety of statements:

Conditionals; IF, IF-ELSE
Loops: WHILE, DO-WHILE, FOR
Selection of cases: SWITCH, CASE DEFAULT
Exit from a function: RETURN
Loop control: CONTINUE, BREAK
Branching: GOTO

Expressions and Operators

The C language includes a rich set of expressions and operators.

Primary expression: invoke functions, select elements from arrays, and extract fields from structures or unions

Arithmetic operators: add, subtract, multiply, divide, modulus

Relational operators: greater than, greater than or equal, less than, less than or equal, not equal

Unary operators: indirect through a pointer, compute an address, logical negation, ones complement, provide the size in bytes of an operand.

Logical operators: AND, OR

Bitwise operators: AND, exclusive OR, inclusive OR, bitwise complement

Data Types and Storage Classes

Data in C is described by its type and storage class. The type determines its representation and use, and the storage class determines its lifetime, scope, and storage allocation. The following data types are fully supported by iC-286.

char
an 8-bit signed integer

int
a 16-bit signed integer

short
same as int (on the 80286)

long
a 32-bit integer

unsigned
a modifier for integer data types (char, int, short, and long) which doubles the positive range of values

float
a 32-bit floating point number which utilizes the 80287 or a software floating point library

double
a 64-bit floating point number

void
a special type that cannot be used as an operand in expressions; normally used for functions called only for effect (to prevent their use in contexts where a value is required).

enum
an enumerated data type
These fundamental data types may be used to create other data types including: arrays, functions, structures, pointers, and unions.

The storage classes available in iC-286 include:

register
suggests that a variable be kept in a machine register, often enhancing code density and speed

extern
a variable defined outside of the function where it is declared; retaining its value throughout the entire program and accessible to other modules

auto
a local variable, created when a block of code is entered and discarded when the block is exited

static
a local variable that retains its value until the termination of the entire program

typedef
defines a new data type name from existing data types

BENEFITS

Faster Compilation

Intel iC-286 compiles C programs substantially faster than standard C compilers because it produces Intel OMF code directly, eliminating the traditional intermediate process of generating an assembly file.

Portability of Code

Because Intel iC-286 supports the STDIO and produces Intel OMF code, programs developed on a variety of machines can easily be transported to the 80286.

Rapid Program Development

Intel iC-286 provides the programmer with detailed error messages and access to PSCOPE and I^{2}ICE to speed program development.

Full Manipulation of the 80286

Intel iC-286 enables the programmer to utilize features of the C language to control bit fields, pointers, addresses and register allocation, taking full advantage of the fundamental concepts of the 80286.

SPECIFICATIONS

Operating Environment

The iC-286 compiler runs host resident on the System 286/310 under the iRMX™II 286 operating system iC-286 can also run as a cross compiler on a VAX 11/780 computer under the VMS operating system 128K bytes of User Memory is required on all versions. The PCDOS system is also a supported environment. Specify desired version when ordering.

Required Hardware

iRMX II version:

— Any iAPX 286, iSBC® 286 or based system capable of running the iRMX II Operating System

VAX version:

— Digital Equipment Corporation VAX 11/780 or compatible computer

PCDOS version:

— PC AT using PCDOS V3.0 or later

Optional Hardware

iRMX-II version:

— Numeric Data Processors for support of the REALMATH standard

VAX version:

— None

PC DOS:

— ICE286, I^{2}ICE286

Required Software

iRMX 286 version:

— iRMX 286 Realtime Multiprogramming Operation

— iRMX 286 Utilities Package

VAX version:

— VMS Operating System

Shipping Media

iRMX II version:

— Double Density iRMX II format $5\frac{1}{4}''$ diskette

VAX version:

— 1600 bpi, 9 track Magnetic tape

DOS version:

— Double Density PC-DOS format $5\frac{1}{4}''$ diskette

ORDERING INFORMATION

Order Code	Description
R286C286	iC-286 Compiler for iRMX 86
D86C286	iC-286 Cross Compiler for PCDOS
VVSC286	iC-286 Cross Compiler for VAX/VMS
MVVSC286	iC-286 Cross Compiler for MicroVAX/VMS

Intel Software License required.

Documentation Package

The C Programming Language by Kernighan and Ritchie (1978 Prentice-Hall)

iC-286 User Manual

SUPPORT

Intel offers several levels of support for this product which are explained in detail in the price list. Please consult the price list for a description of the support options available.

*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. VAX, VMS are trademarks of Digital Equipment Corporation.

FORTRAN 80286 SOFTWARE PACKAGE

- **Features High-Level Language Support for Floating-Point Calculations, Transcendentals, Interrupt Procedures, and Run-Time Exception Handling**
- **Meets ANSI FORTRAN 77 Subset Language Specifications**
- **Supports 8087 and 80287 Numeric Data Processors for Fast and Efficient Execution of Numeric Instructions**
- **Uses REALMATH Floating-Point Standard for Consistent and Reliable Results**
- **Supports Arrays Larger Than 64K**
- **Offers Upward Compatibility with FORTRAN 86**
- **Provides FORTRAN Run-Time Support for 80286 Based Design**
- **Provides Users Ability to do Formatted and Unformatted I/O with Sequential or Direct Access Methods**
- **I²ICE™/ICE286 Symbolic Debugging Fully Supported**
- **Supports Complex Data Types**
- **Choice of Industry Standard Hosts**
- **Full Compatibility with Microsoft™ C Modules under XENIX**

FORTRAN 286 meets the ANSI FORTRAN 77 Language Subset Specification and includes many features of the full standard. Therefore, the user is assured of portability of most existing ANS FORTRAN programs and of full portability from other computer systems with an ANS FORTRAN 77 Compiler.

FORTRAN 286 is available to run on the IBM PC AT running PC DOS Version 3.0 or later, Digital Equipment VAX†/VMS† and Intel System 86/3XX running iRMX™ II and XENIX operating systems.

FORTRAN 286 is one of a complete family of compatible programming languages for 80286 development. Therefore, users may choose the language best suited for a specific problem solution.

†VAX, VMS are trademarks of Digital Equipment Corporation.

*IBM, AT are registered trademarks of International Business Machines Corporation.

Microsoft™ C is a trademark of Microsoft Corporation.

FEATURES

Extensive High-Level Language Numeric Processing Support

Single (32-bit), double (64-bit), and double extended precision (80-bit) complex (two 32-bit), and double complex (two 64-bit) floating-point data types

REALMATH Proposed IEEE Floating-Point Standard) for consistent and reliable results

Full support for all other data types: integer, logical, character

Ability to use hardware or software (simulator) floating-point support chosen at link time

ANSI FORTRAN 77 Standard

Intel® Microprocessor Support

FORTRAN 286 language features support of 8087 and 80237 Numeric Data Processors

Intrinsics allow user to control iAPX 8087 and 80287 Numeric Data processor

80286 architectural advantages used for indexing and character-string handling

Symbolic debugging of application using ICE emulators

Microprocessor Application Support

— Direct byte- or word-oriented port I/O

— Reentrant procedures

— Interrupt procedures

BENEFITS

FORTRAN 286 provides a means of developing application software for the Intel 80286 products lines in a familiar, widely accepted, and industry-standard programming language. FORTRAN 286 will greatly enhance the user's ability to provide cost-effective software development for Intel microprocessors as illustrated by the following:

Early Project Completion

FORTRAN is an industry-standard, high-level numerics processing language. FORTRAN programmers can use FORTRAN 286 on microprocessor projects with little retraining. Existing FORTRAN software can be compiled with FORTRAN 286 and programs developed in FORTRAN 286 can run on other computers with ANSI FORTRAN 77 with little or no change. Libraries of mathematical programs using ANSI 77 standards may be compiled with FORTRAN 286.

Application Object Code Portability for a Processor Family

FORTRAN 286 modules "talk" to the resident PCDOS operating system using Intel's standard interface for all development-system software. This allows an application developed under DOS operating system to execute on iRMX II, or a user-supplied operating system by linking in the iRMX II or other appropriate interface library. A standard logical-record interface enables communication with non-standard I/O devices.

Comprehensive, Reliable and Efficient Numeric Processing

The unique combination of FORTRAN 80286, 8087, 80287 Numeric Data processor, and REALMATH (Proposed IEEE Floating-Point Standard) provide universal consistency in results of numeric computations and efficient object code generation.

SPECIFICATIONS

Documentation Package

FORTRAN 80286 User's Guide

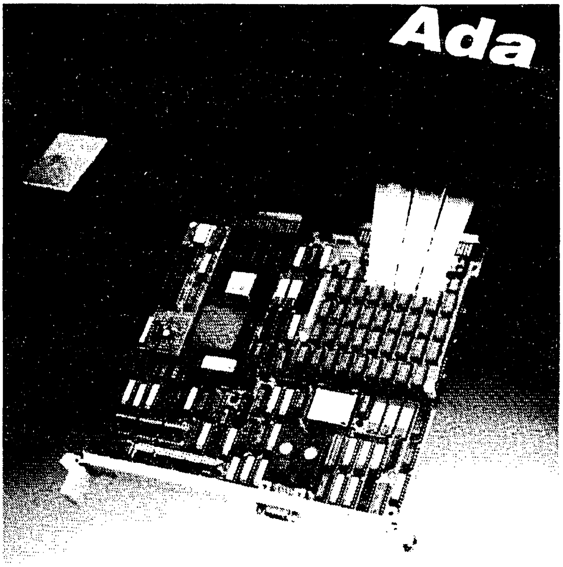
ORDERING INFORMATION

Order Code	Operating Environment
D86FOR286NL	IBM PC AT running PC DOS Version 3.0 or later
R286FOR286	Intel System 86/3XX running iRMX II
X286FOR286	FORTRAN 286 running under Xenix 286

SUPPORT AVAILABLE

Software updates, Subscription Service, Hotline Support.

Ada: CROSS-DEVELOPMENT FOR THE 80386 MICROPROCESSOR



Intel's Ada -386 Cross-Compilation Package comprises a rich set of Ada language tools for the programmer wanting to develop Ada applications targeted to the industry's leading 32-bit architecture, the 80386 microprocessor. This tool set includes an Ada cross compiler which generates compact, highly optimized, code for embedded real-time 80386 applications. The cross compiler and other tools making up the Cross-Compilation Package form a complete development environment designed specifically to support large scale, mission critical Ada programming development projects.

Intel's Ada-386 Cross-Compilation Package runs under VAX/VMS, and features, in addition to the cross compiler, a number of tools which make the programmer's job more efficient. These tools include a VMS hosted and targeted compiler to enable the programmer to do unit testing on the VAX early in the development cycle. Also included with each of the compilers is an Ada symbolic debugger to facilitate the debug process, and sophisticated code generation tools, such as the Linker and Global Optimizer, to help make the target code smaller and more efficient. An object module Importer is included with the cross compiler to allow the programmer to save and make use of program modules written in other Intel 80386 languages.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

September, 1988

Order Number: 280784-002

KEY COMPILATION PACKAGE FEATURES

- Tight, efficient, 32-bit 80386 code
  —Generated code designed and optimized for the 80386 architecture
- Built-in support for the 80386
  —Full representation specifications, including address clauses
  —Machine code insertion
- 80387 coprocessor support
  —Full IEEE numerics support
- pragma INTERFACE
  —Call modules written in other Intel languages: ASM-386, PL/M-386 & C-386
- Highly optimized interrupt handling
  —Fast execution of interrupt handlers without requiring a context switch
- Pre-emptive delay
  —Force synchronization at the end of programmed "delays"
- Optional download and debug paths using the VAX-hosted Ada debugger
  —With a ROM-resident target debug monitor (included), or
  —ICE-386 (80386 In-Circuit Emulator) for less intrusive debugging
- Modular, configurable runtime system
  —Linker excludes routines not required by the embedded application (no overhead penalty)
  —Easily configured for different hardware environments

REAL TIME Ada FROM INTEL

Intel's Ada development environment makes developing real-time embedded applications convenient and easy. All steps in the development process can proceed, start to finish, from a VAX terminal—from initial unit testing with the VMS-targeted compiler to compiling and linking using the 80386-targeted cross compiler. Downloading to the 80386 target and debugging can also be accomplished from the VAX terminal.

COMPILATION PACKAGE COMPONENTS

- **Compilers & Library Tools**

  Both compilers, the VMS targeted version and the 80386 cross targeted version, use the same user interface, commands and library management tools so the programmer learns them only once. The cross compiler has an optional switch which directs the compiler to produce assembly language text interspersed with Ada source text as comments. This feature gives the programmer a convenient way to hand-inspect the code. The assembly language text can also be assembled using the Intel 80386 Assembler.

  The cross compiler has important optimizations to help meet real-time needs. For example, when response times to interrupts are critical, the programmer can speed up response times by invoking the "function mapped" optimization via a special compiler directive, "pragma INTERRUPT." This function mapping enables an interrupt handler to execute without first requiring a task switch, i.e., within the context of the interrupted task.

- **Global Optimizer**

  The Global Optimizer is used to reduce the size and increase the speed of embedded application code. This tool is invoked at the user's option, but usually after most of the coding and debugging is complete. Some key functions performed by the Global Optimizer include:

Feature	**Benefit**
dead code elimination	removes all unnecessary code from the application
unused subprogram deletion	
common subexpression elimination	
constant folding and propagation	combines and substitutes constants where appropriate
static evaluation of conditional expressions	reduces conditional expressions to boolean equivalents where possible

- **Linker**

  The Linker combines separately compiled Ada modules, imported non-Ada modules (see Importer below), and the Ada runtime system into one executable image. To reduce target code size, the Linker also eliminates subprograms in the application code and in the runtime system that are not actually required by the application. The programmer may also use the Linker to produce output in a format suitable for burning PROMs.

- **Importer**

  The Importer can help preserve previous software investments. The Importer converts object modules from Intel's OMF-386 format to a format suitable for linking with Ada modules. An Ada application can call these imported non-Ada modules through "pragma INTERFACE." Pragma INTERFACE is supported for Intel's ASM-386, PL/M-386 and C-386 languages.

- **Ada Runtime System**

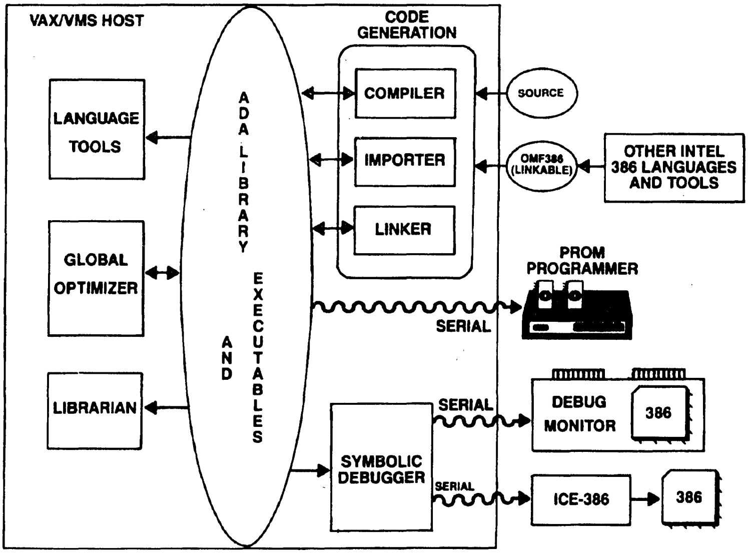
  All the necessary low-level support routines for executing programs on a bare 80386 microprocessor are provided in the Ada Runtime System. These routines are responsible for managing tasking, interrupts, the real-time clock and memory. Also included are predefined Ada packages, such as Text_I/O, IO_Exceptions, Unchecked_Conversion and Calendar. The Ada Runtime System is written almost entirely in Ada, with a small number of packages written in 80386 assembly language to support key low-level functions. The Runtime System is easily configured for different 80386 hardware environments, and source code is provided for this purpose.

- **Symbolic Debugger**
  A VAX-resident symbolic debugger is supplied for each compiler. The debugger allows the programmer to debug at the source level while the code is executing on the target. Log and script files may be used to automate repetitive debug sessions. Important debugger features include:

Feature	**Benefit**
machine level interface	step through machine instructions; read/write to registers, memory, and I/O ports
single/multiple step	step through source code by single or multiple statements
breakpoints	halt execution at specified points
call chain display	display the dynamic nesting of a program at a particular point in time
task status display	determine the status of a task at a particular point in time
variable display	examine values of program variables
trap unhandled exceptions	examine state of the target program when an unhandled exception occurs

A small debug monitor, supplied in PROM, is used with the Symbolic Debugger. The code can also be downloaded and debugged using Intel's ICE-386 in-circuit emulator.

- **Language Tools**
  Intel's Ada development environment includes tools that help development projects run more smoothly. A *Cross Referencer* provides a cross-reference listing of all source file locations where a user-defined symbol in an Ada compilation unit is defined. A *Source Dependency Lister* produces a valid compilation order list for compilation units in an Ada program and lists dependencies among units. A *Source Formatter* (or "pretty printer") takes as input a non-formatted Ada source file and outputs a formatted version of the same text that adheres to standardized language conventions.



WORLDWIDE SERVICE AND SUPPORT

Complete hardware and software support is provided. The Ada-386 Compilation Package comes with Intel's standard 90-day warranty plus an extended one-year maintenance agreement, ensuring uninterrupted support for a full 15 months. One-year maintenance contract renewals are available from Intel annually.

ORDERING INFORMATION

Order Code	Host Configuration	Product
VVSAda386-75	VAX 730, 750	Ada-386 Cross-Compilation Package. Included are the following tools hosted on VAX/VMS: • 80386-targeted Ada cross compiler & library tools • VMS self-targeted Ada compiler & library tools • For each compiler a Symbolic Debugger, a Global Optimizer and Language Tools • Ada-386 Linker and Importer • Ada-386 Runtime System The Compilation Package also includes full documentation, Intel's standard 90-day warranty, and an extended one-year maintenance contract.
-82	VAX 78X, 8200	
-83	VAX 8300	
-85	VAX 85XX, 86XX, 8700	
-88	VAX 8800	
MVVSAda386-VS	VAXStationII	
-02	MicroVAXII	
ICE386HW	N/A	80386 In-Circuit Emulator (hardware only). Used with the VAX-hosted Symbolic Debugger, the ICE-386 offers the programmer an alternative download method to using the ROM-resident debug kernel supplied with the Compilation Package.

Note: Ada-386 software license required for each host CPU. Multiple copies require multiple licenses.

*VAX/VMS is a registered trademark of Digital Equipment Corporation.

INTEL386™ FAMILY DEVELOPMENT SUPPORT

COMPREHENSIVE DEVELOPMENT SUPPORT FOR THE INTEL386™ FAMILY OF MICROPROCESSORS

The perfect complement to the Intel386™ Family of microprocessors is the optimum development solution. From a single source, Intel, comes a complete, synergistic hardware and software development toolset, delivering full access to the power of the Intel386 architecture in a way that only Intel can.

Intel development tools are easy to use, yet powerful, with contemporary user interface techniques and productivity boosting features such as symbolic debugging. And you'll find Intel first to market with the tools needed to start development, and with lasting product quality and comprehensive support to keep development on-track.

If what interests you is getting the best product to market in as little time as possible, Intel is the choice.

FEATURES

- Comprehensive support for the full 32 bit Intel386 architecture, including protected mode and 4 gigabyte physical memory addressing
- Source line display and symbolics allow debugging in the context of the original program
- Architectural extensions in Intel high-level languages provides for manipulating hardware directly without assembly language routines
- A common object code format (OMF-386) supports the intermixing of modules written in various languages
- ROM-able code is output directly from the language tools, significantly reducing the effort necessary to integrate software into the final target system
- Support for the 80387 numeric coprocessor

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

September, 1988
Order Number: 280808-001

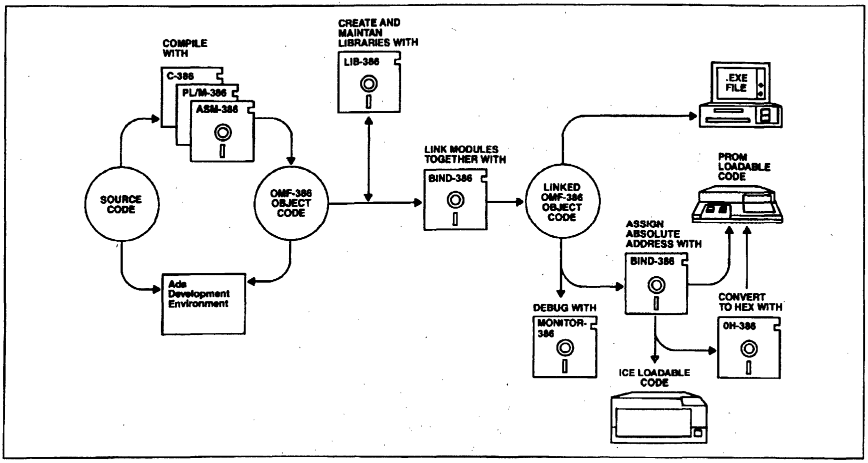


Figure 1. Intel Microprocessor Development Environment

ASM-386 MACRO ASSEMBLER

ASM-386 is a "high-level" macro assembler for the Intel386 Family. ASM-386 offers many features normally found only in high-level languages. The macro facility in ASM-386 saves development time by allowing common program sequences to be coded only once. The assembly language is strongly typed, performing extensive checks on the usage of variables and labels.

Other ASM-386 features include:

- "High-level" assembler mnemonics to simplify the language
- Structures and records for data representation
- Upward compatibility with ASM-286

PL/M-386 COMPILER

PL/M-386 is a structured high-level system implementation language for the Intel386 Family. PL/M-386 supports the implementation of protected operating system software by providing built-in procedures and variables to access the Intel386 architecture.

For efficient code generation, PL/M-386 features four levels of optimization, a virtual symbol table, and four models of program size and memory usage.

Other PL/M-386 features include:

- The ability to define a procedure as an interrupt handler as well as facilities for generating interrupts
- Direct support of byte, half-word, and word input and output from microprocessor ports
- Upward compatibility with PL/M-286 and PL/M-86 source code

PL/M-386 combines the benefits of a high-level language with the ability to access the Intel386 architecture. For the development of systems software, PL/M-386 is a cost-effective alternative to assembly language programming.

C-386 COMPILER

C-386 brings the C language to the Intel386 Family. For code efficiency, C-386 features two levels of optimization, three models of program size and memory usage, and an extremely efficient register allocator. The C-386 compiler eliminates common code, eliminates redundant loads and stores, and resolves span dependencies (shortens branches) within a program.

C-386 allows full access to the Intel386 architecture through control of bit fields, pointers, addresses, and register allocations.

Other C-386 features include:

- An interrupt directive defining a function as an interrupt function
- Built-in functions allow direct access to the microprocessor through the inline insertion of machine code
- Structure assignments, functions taking structure arguments, and returning structures, and the void and enum data types

The C-386 runtime library is implemented in layers. The upper layers include the standard I/O library (STDIO), memory management routines, conversion routines, and string manipulation routines. The lowest layer, operating system interface routines, is documented for adaptation to the target environment.

RLL-386™ RELOCATION, LINKAGE, AND LIBRARY TOOLS

The RLL-386™ relocation, linkage, and library tools are a cohesive set of utilities featuring comprehensive support of the full Intel386 architecture. RLL-386 provides for a variety of functions—from linking separate modules, building an object library, or linking in 80387 support, to building a task to execute under protected mode or the multi-tasking, memory protected system software itself.

The RLL-386 relocation, linkage, and library tools package includes a program binder for linking ASM, PL/M, and C modules together, a system builder for configuring protected, multi-task systems, a cross reference mapper, a program librarian, an 80387 numeric coprocessor support library, and a conversion utility for outputing hex format code for PROM programming.

Ada-386™ CROSS COMPILATION PACKAGE

The Ada-386™ Cross-Compilation Package is a complete development environment for embedded real-time Ada applications for the 386 microprocessor. The Ada cross compiler, which runs under VAX/VMS, generates code highly optimized for the 80386. The Ada-386 Cross-Compilation Package also features a VMS hosted and targeted compiler and tools to support software debugging before the target system is available. Sophisticated code generation tools, such as the Global Optimizer, help make the target code smaller and more efficient.

Ada-386 includes a source level symbolic debugger working in unison with a small debug monitor supplied in PROM. Code can also be downloaded and debugged using Intel's ICE™-386 In-Circuit Emulator.

Other Ada-386 features include:

- The ability to directly call Intel's iRMK real-time kernel
- An object module importer allows program modules written in other Intel386 Family languages to be linked with Ada modules
- Built in support for the 386, including machine code insertion and full representation specifications
- Highly optimized interrupt handling—fast execution of interrupt handlers without requiring a context switch

INTEL386™ FAMILY IN-CIRCUIT EMULATORS

Intel386 Family in-circuit emulators embody exclusive technology that gives the emulator access to internal processor states that are accessible in no other way. Intel386 microprocessors fetch and execute instructions in parallel; fetched instructions are not necessarily executed. Because of this, an emulator without this access to internal processor states is prone to error in determining what actually occurred inside the microprocessor. With Intel's exclusive technology, Intel386 Family emulators are one hundred percent accurate.

Other features of Intel386 Family in-circuit emulators include:

- Unparalleled support of the Intel386 architecture, notably the native protected mode
- Emulation at clock speeds to 25MHz and full-featured trigger and trace capabilities
- Non-intrusive operation
- Convertible to support any Intel386 microprocessor

With symbolic debugging, memory locations can be examined or modified using symbolic references to the original program, such as a procedure or a variable name, line number, or program label. Source code associated with a given line number can be displayed, as can the type information of variables, such as byte, word, record, or array. Microprocessor data structures, such as registers, descriptor tables, and page tables, can also be examined and modified using symbolic names. The symbolic debugging information for use with Intel development tools is produced only by Intel languages.

MONITOR-386™ SOFTWARE DEBUGGER

MONITOR-386™ is a software debugger for 386 and 386SX-based systems. MONITOR-386 provides program execution control and symbolic processor and memory interrogation and modification. Hardware and software breakpoints can be set at symbolic addresses and program execution can be single-stepped through assembly or high-level language instructions.

Other MONITOR-386 features include:

- Debug procedures, named user-definable sequences of MONITOR-386 commands, enable users to define macro commands that would otherwise take several lines of command entries to perform the same function
- A disassembler/single line assembler allows users to display memory as and patch memory with 80386/80387 mnemonics

MONITOR-386, used in conjunction with Intel single board computers iSBC® 386/22 and iSBC 386/116, can debug software before a functional prototype of the target system is available.

iPAT-386™ PERFORMANCE ANALYSIS TOOL

iPAT-386™ performance analysis tool provides analysis of real-time software executing on a 386-based target system. With iPAT-386, it is possible to speed-tune applications, optimize use of operating systems, determine response characteristics, and identify code execution coverage.

By examining iPAT-386 histogram and tabular information about procedure usage (with the option of including interaction with other procedures, hardware, the operating system, or interrupt service routines) for critical functions, performance bottlenecks can be identified. With iPAT-386 code execution coverage information, the completeness of testing can be confirmed. iPAT-386 can be used in conjunction with Intel's ICE-386™ in-circuit emulator to control test conditions.

iPAT-386 provides real-time analysis up to 20MHz, performance profiles of up to 125 partitions, and code execution coverage analysis over 252K.

SERVICE, SUPPORT, AND TRAINING

To augment its development tools, Intel offers a full array of seminars, classes, and workshops, field application engineering expertise, hotline technical support, and on-site service.

PRODUCT SUPPORT MATRIX

	Component			Host	
Product	386	386sx	376	DOS	VAX/VMS
ASM-386 Macro Assembler	✓	✓	✓	✓	✓
PL/M-386 Compiler	✓	✓	✓	✓	✓
C-386 Compiler	✓	✓	✓	✓	✓
RLL-386 Relocation, Linkage, and Library tools	✓	✓	✓	✓	✓
Ada-386 Cross Compilation Package	✓				✓
Intel386 Family In-Circuit Emulators	✓	✓	✓	✓	
Monitor-386 Software Debugger	✓	✓		✓	
iPAT-386 Performance Analysis Tool	✓			✓	

Intel386, 386, 386SX, 376, ICE, and iSBC are trademarks of Intel Corporation.
VAX and VMS are registered trademarks of Digital Equipment Corporation.

ORDERING INFORMATION

For direct information on Intel's Development Tools, or for the number of your nearest sales office or distributor, call 800-874-6835 (U.S.). For information or literature on additional Intel products, call 800-548-4725 (U.S. and Canada).

AEDIT SOURCE CODE AND TEXT EDITOR

PROGRAMMER SUPPORT

AEDIT is a full-screen text editing system designed specifically for software engineers and technical writers. With the facilities for automatic program block indentation, HEX display and input, and full macro support, AEDIT is an essential tool for any programming environment. And with AEDIT, the output file is the pure ASCII text (or HEX code) you input—no special characters or proprietary formats.

Dual file editing means you can create source code and its supporting documents at the same time. Keep your program listing with its errors in the background for easy reference while correcting the source in the foreground. Using the split-screen windowing capability, it is easy to compare two files, or copy text from one to the other. The DOS system-escape command eliminates the need to leave the editor to compile a program, get a directory listing, or execute any other program executable at the DOS system level.

There are no limits placed on the size of the file or the length of the lines processed with AEDIT. It even has a batch mode for those times when you need to make automatic string substitutions or insertions in a number of separate text files.

AEDIT FEATURES

- Complete range of editing support—from document processing to HEX code entry and modification
- Supports system escape for quick execution of PC-DOS System level commands
- Full macro support for complex or repetitive editing tasks
- Hosted on PC-DOS and RMX operating systems
- Dual file support with optional split-screen windowing
- No limit to file size or line length
- Quick response with an easy to use menu driven interface
- Configurable and extensible for complete control of the editing process

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

September, 1988
Order Number: 280804-002

FEATURES

POWERFUL TEXT EDITOR

As a text editor, AEDIT is versatile and complete. In addition to simple character insertion and cursor positioning commands, AEDIT supports a number of text block processing commands. Using these commands you can easily move, copy, or delete both small and large blocks of text. AEDIT also provides facilities for forward or reverse string searches, string replacement and query replace.

AEDIT removes the restriction of only inserting characters when adding or modifying text. When adding text with AEDIT you may choose to either insert characters at the current cursor location, or over-write the existing text as you type. This flexibility simplifies the creation and editing of tables and charts.

USER INTERFACE

The menu-driven interface AEDIT provides makes it unnecessary to memorize long lists of commands and their syntax. Instead, a complete list of the commands or options available at any point is always displayed at the bottom of the screen. This makes AEDIT both easy to learn and easy to use.

FULL FLEXIBILITY

In addition to the standard PC terminal support provided with AEDIT, you are able to configure AEDIT to work with almost any terminal. This along with user-definable macros and full adjustable tabs, margins, and case sensitivity combine to make AEDIT one of the most flexible editors available today.

MACRO SUPPORT

AEDIT will create macros by simply keeping track of the command and text that you type, "learning" the function the macro is to perform. The editor remembers your actions for later execution, or you may store them in a file to use in a later editing session.

Alternatively, you can design a macro using AEDIT's powerful macro language. Included with the editor is an extensive library of useful macros which you may use or modify to meet your individual editing needs.

TEXT PROCESSING

For your documentation needs, paragraph filling or justification simplifies the chore of document formatting. Automatic carriage return insertion means you can focus on the content of what you are typing instead of how close you are to the edge of the screen.

SERVICE, SUPPORT, AND TRAINING

Intel augments its development tools with a full array of seminars, classes, and workshops; on-site consulting services; field application engineering expertise; telephone hot-line support; and software and hardware maintenance contracts. This full line of services will ensure your design success.

SPECIFICATIONS

HOST SYSTEM

AEDIT for PC-DOS has been designed to run on the IBM* PC XT, IBM PC AT, and compatibles. It has been tested and evaluated for the PC-DOS 3.0 or greater operating system.

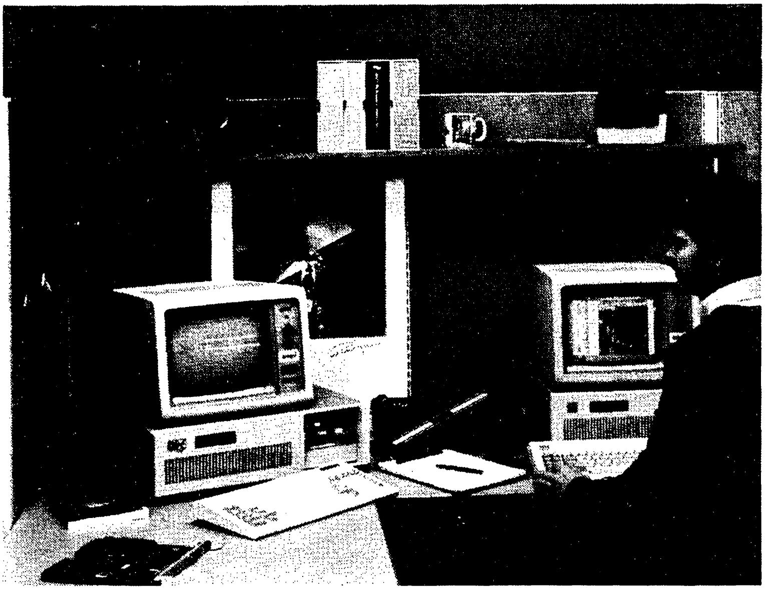
Versions of AEDIT are available for the iRMX™-86 and RMX II Operating System.

ORDERING INFORMATION

D86EDINL	AEDIT Source Code Editor Release 2.2 for PC-DOS with supporting documentation
122716	AEDIT-DOS Users Guide
122721	AEDIT-DOS Pocket Reference
RMX864WSU	AEDIT for iRMX-86 Operating System
R286EDI286EU	AEDIT for iRMX II Operating System

For direct information on Intel's Development Tools, or for the number of your nearest sales office or distributor, call 800-874-6835 (U.S.). For information or literature on additional Intel products, call 800-548-4725 (U.S. and Canada).

iPAT™ PERFORMANCE ANALYSIS TOOL



REAL-TIME SOFTWARE ANALYSIS FOR THE 8086/88, 80186/188, 80286, AND 80386

Intel's iPAT™ Performance Analysis Tool enables OEMs developing applications based on the 8086/88, 80186/188, 80286, or 80386 microprocessors to analyze real-time software execution in their prototype systems at speeds up to 20 MHz. Through such analysis, it is possible to speed-tune applications with real-time data, optimize use of operating systems (such as Intel's iRMX® II Real-Time Multitasking Executive for the 80286 and 80386, and iRMK™ Real-Time Multitasking Kernel for the 80386), characterize response characteristics, and determine code execution coverage by real-time test suites. Analysis is performed symbolically, non-intrusively, and in real-time with 100% sampling in the microprocessor prototype environment. iPAT supports analysis of OEM-developed software built using 8086, 80286, and 80386 assemblers and compilers supplied by Intel and other vendors.

All iPAT Performance Analysis Tool products are serially linked to DOS computer systems (such as IBM* PC AT, PC XT, and PS/2* Model 80) to host iPAT control and graphic display software. Several means of access to the user's prototype microprocessor system are supported. For the 80286 (real and protected mode), a 12.5 MHz iPAT-286 probe can be used with the iPATCORE system. For the 8086/88 (MAX MODE designs only), a 10 MHz iPAT-88 probe can be used with the iPATCORE system. iPATCORE systems also can be connected to sockets provided on the ICE™-286 and ICE-186 in-circuit emulators, or interfaced to I^{2}ICE™ in-circuit emulators with probes supporting the 8086/88, 80186/188, or 80286. The 20 MHz iPAT™-386™ probe, also supported by the common iPATCORE system, can be operated either in "piggyback" fashion connected to an Intel ICE in-circuit emulator for the Intel386™, or directly connected to a prototype system independent of an ICE. iPAT-386 supports all models of 80386 applications anywhere in the lowest 16 Megabytes of the 80386 linear address space.

iPAT FEATURES

- Up to 20 MHz real-time analysis
- Histograms and analysis tables
- Performance profiles of up to 125 partitions
- Code execution coverage over up to 252K
- Hardware or software interrupt analysis
- Simple use with function keys and graphics
- Use with or without Intel ICEs

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel.

June, 1988
Order Number: 280786-002

FEATURES

MOST COMPLETE REAL-TIME ANALYSIS AVAILABLE TODAY

iPAT Performance Analysis Tools use in-circuit probes containing proprietary chip technology to achieve full sampling in real-time non-intrusively.

MEETS THE REAL-TIME DESIGNER'S NEEDS

The iPAT products include support for interactions between real-time software and hardware interrupts, real-time operating systems, "idle" time, and full analysis of real-time process control systems.

SPEED-TUNING YOUR SOFTWARE

By examining iPAT histogram and tabular information about procedure usage (including or not including their interaction with other procedures, hardware, operating systems, or interrupt service routines) for critical functions, the software engineer can quickly pinpoint trouble spots. Armed with this information, bottlenecks can be eliminated by means such as changes to algorithms, recoding in assembler, or adjusting system interrupt priorities. Finally, iPAT can be used to prove the acceptability of the developer's results.

EFFICIENCY AND EFFECTIVENESS IN TESTING

With iPAT code execution coverage information, product evaluation with test suites can be performed more effectively and in less time. The evaluation team can quickly pinpoint areas of code that are executed or not executed under real-time conditions. By this means, the evaluation team can substantially remove the "black box" aspect of testing and assure 100% hits on the software under test. Coverage information can be used to document testing at the module, procedure, and line level. iPAT utilities also support generation of instruction-level code coverage information.

ANALYSIS WITH OR WITHOUT SYMBOLICS

If your application is developed with "debug" symbolics generated by Intel 8086, 80286, or 80386 assemblers and compilers, iPAT can use them—automatically. Symbolic names also can be defined within the iPAT environment, or conversion tools supplied with the iPAT products can be used to create symbolic information from virtually any vendor's map files for 8086, 80286, and 80386 software tools.

REAL OR PROTECTED MODE

iPAT supports 80286 and 80386 protected mode symbolic information generated by Intel 80286 and 80386 software tools. It can work with absolute addresses, as well as base:offset or selector:offset references to partitions in the prototype system's execution address space.

FROM ROM-LOADED TO OPERATING SYSTEM LOADED APPLICATIONS

The software analysis provided by iPAT watches absolute execution addresses in-circuit in real time, but also supports use of various iPAT utilities to determine the load locations for load-time located software, such as applications running under iRMXII, DOS, Microsoft Windows*, or MS*-OS/2.

USE STANDALONE OR WITH ICE

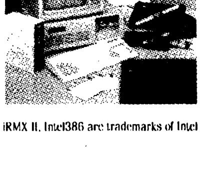
The iPAT-386, iPAT-286, and iPAT-86/88 probes, together with an iPATCORE system, provide standalone software analysis independent of an ICE (in-circuit emulator) system. The iPATCORE system and DOS-hosted software also can be used together with ICE-386, ICE-286, ICE-186, and I^{2}ICE-86/88, 186/188, or 286 in-circuit emulators and DOS-hosted software. Under the latter scenario, the user can examine prototype software characteristics in real-time on one DOS host while another DOS host is used to supply input or test conditions to the prototype through an ICE. It also is possible to use an iPATCORE and I^{2}ICE system with integrated host software on a single Intel Series III or Series IV development system, or on a DOS computer.

UTILITIES FOR YOUR NEEDS

Various utilities supplied with iPAT products support generation of symbolic information from map files associated with 3rd-party software tools, extended analysis of iPAT code execution coverage analysis data, and convenience in the working environment. For example, symbolics can be generated for maps produced by most software tools, instruction-level code execution information can be produced, and iRMXII-format disks can be read/written in DOS floppy drives to facilitate file transfer.

WORLDWIDE SERVICE AND SUPPORT

All iPAT Performance Analysis Tool products are supported by Intel's worldwide service and support. Total hardware and software support is available, including a hotline number when the need is there.



*Microsoft Windows and MS are trademarks of Microsoft Corporation. IBM and PS/2 are trademarks of International Business Machines. IPAT, ICE, iRMK, I^{2}ICE, iRMX II, Intel386 are trademarks of Intel Corporation.

FEATURES

CONFIGURATION GUIDE

For all of the following application requirements, the iPAT system is supported with iPAT 2.0 (or greater) or iPAT/I^{2}ICE 1.2 (or greater) host software, as footnoted.

Application Software	Option	iPAT Order Codes	Host System
80386 Embedded	#1	iPAT386DOS[1], iPATCORE	DOS
iRMK on 80386	#1	iPAT386DOS, iPATCORE	DOS
iRMXII OS-Loaded or Embedded on 386	#1	iPAT386DOS, iPATCORE	DOS
OS/2-Loaded on 386	#1	iPAT386DOS, iPATCORE	DOS
iRMXII OS-Loaded or Embedded	#1	iPAT286DOS, iPATCORE	DOS
80286 Embedded	#1	iPAT286DOS, iPATCORE	DOS
	#2	ICEPATKIT[2]	DOS
	#3	I^{2}ICEPATKIT[3]	DOS
	#4	IIIPATD, iPATCORE[3]	DOS[4]
	#5	IIIPATB, iPATCORE[3]	Series III[4]
	#6	IIIPATC, iPATCORE[3]	Series IV[4]
DOS OS-Loaded 80286	#1	iPAT286DOS, iPATCORE	DOS
OS/2 OS-Loaded 80286	#1	iPAT286DOS, iPATCORE	DOS
80186/188 Embedded	#1	ICEPATKIT[2]	DOS
	#2	I^{2}ICEPATKIT[3]	DOS
	#3	IIIPATD, iPATCORE[3]	DOS[4]
	#4	IIIPATB, iPATCORE[3]	Series III[4]
	#5	IIIPATC, iPATCORE[3]	Series IV[4]
DOS OS-Loaded 8086/88	#1	iPAT88DOS, iPATCORE	DOS
8086/88 Embedded	#1	iPAT88DOS, iPATCORE	DOS
	#2	I^{2}ICEPATKIT[3]	DOS
	#3	IIIPATD, iPATCORE[3]	DOS[4]
	#4	IIIPATB, iPATCORE[3]	Series III[4]
	#5	IIIPATC, iPATCORE[3]	Series IV[4]

Notes:

1. Operable standalone or with ICE-386 (separate product; separate host). iPAT-386 probe connects directly to prototype system socket, or to optional 4" probe-to-socket hinge cable (order code TA386A), or to ICE-386 probe socket.
2. Requires ICE-186 or ICE-286 in-circuit emulator system.
3. Requires I^{2}ICE in-circuit emulator system.
4. Includes iPAT/I^{2}ICE integrated software (iPAT/I^{2}ICE 1.2 or greater), which only supports sequential iPAT and ICE operation on one host, rather than in parallel on two hosts (iPAT 2.0 or greater).

SPECIFICATIONS

HOST COMPUTER REQUIREMENTS

All iPAT Performance Analysis Tool products are hosted on IBM PC AT, PC XT, or PS/2 Model 80 personal computers, or 100% compatibles, and use a serial link for host-to-iPAT communications. At least a PC AT class system is recommended. The DOS host system must meet the following minimum requirements:

- 640K Bytes of Memory
- 360K Byte or 1.2M Byte floppy disk drive
- Fixed disk drive
- A serial port (COM1 or COM2) supporting 9600 baud data transfer
- DOS 3.0 or later
- IBM or 100% compatible BIOS

PHYSICAL DESCRIPTIONS

	Width		Height		Length	
Unit	Inches	Cm.	Inches	Cm.	Inches	Cm.
iPATCORE	8.25	21.0	1.75	4.5	13.75	35.0
Power Supply	7.75	20.0	4.25	11.0	11.0	28.0
iPAT-386 probe	3.0	7.6	0.50	1.3	4.0	10.1
iPAT-286 probe	4.0	10.2	1.12	2.8	6.0	15.3
iPAT-86 probe	4.0	10.2	1.12	2.8	6.0	15.3
iPATCABLE (to ICE-186/286)	4.0	10.2	.25	.6	36.0	91.4
IIPATB,C,D (I^{2}ICE board)	12.0	30.5	12.0	30.5	.5	1.3
Serial cables PC AT/XT PS/2					144.0	370.0

ELECTRICAL CONSIDERATIONS

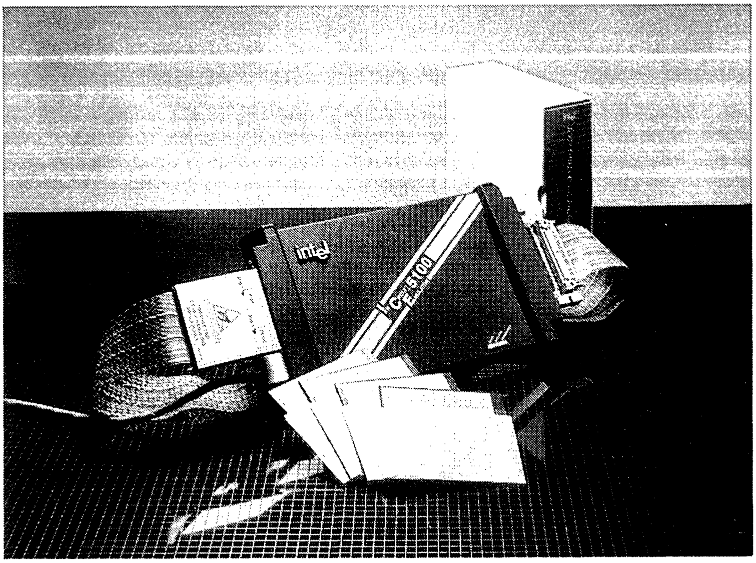
The iPATCORE system power supply uses an AC power source at 100V, 120V, 220V, or 240V over 47Hz to 63Hz. 2 amps (AC) at 100V or 120V; 1 amp at 220V or 240V.

iPAT-386, iPAT-286 and iPAT-86/88 probes are externally powered, impose no power demands on the user's prototype, and can thus be used to analyze software activity through power down and power up of a prototype system. For ICE-386, ICE-286, ICE-186, and I^{2}ICE microprocessor probes, see the appropriate in-circuit emulator factsheets.

ENVIRONMENTAL SPECIFICATIONS

Operating Temperature:	10°C to 40°C (50°F to 104°F) ambient
Operating Humidity:	Maximum of 85% relative humidity, non-condensing

ICE-5100/452 In-Circuit Emulator



IN-CIRCUIT EMULATOR FOR THE UPI™-452 FAMILY OF PROGRAMMABLE I/O PROCESSORS

The ICE-5100/452 In-Circuit Emulator is a complete hardware/software debug environment for developing embedded control applications based on the Intel UPI™-452 family of I/O peripherals. With high-performance full-speed emulation, symbolic debugging, and flexible memory mapping, the ICE-5100/452 emulator expedites all stages of development: hardware development, software development, system integration, and system test; shortening your project's time to market.

FEATURES

- Full speed to the speed of the component.
- 64KB of emulation mapped memory.
- 254 frames of execution trace.
- Symbolic debug.
- Serial link to an IBM PC XT, AT, 100% compatible.
- Four address breakpoints with in-range, out-of-range, and page breaks.
- On-line disassembler and single line assembler.
- Full emulation and debug support for the FIFO Buffer.
- Source code display.
- ASM-51 and PL/M-51 language support.
- Pop-up help.
- DOS shell escape.
- On-line tutorial.
- Built-in CRT based editor.
- System self-test diagnostics.
- Worldwide service and support.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

September, 1988
Order Number: 280817-001

ONE TOOL FOR ENTIRE DEVELOPMENT CYCLE

The ICE-5100/452 emulator speeds target system development by allowing hardware and software design to proceed simultaneously. You can develop software even before prototype hardware is finished. And because the ICE-5100/452 emulator precisely matches the component's electrical and timing characteristics, it's a valuable tool for hardware development and debug. Thus, the ICE-5100/452 emulator can debug a prototype or production system at any stage in its development, without introducing extraneous hardware or software test tools.

HIGH-SPEED, REAL-TIME EMULATION

The ICE-5100/452 emulator provides full-speed, real-time emulation up to the speed of the component. Because the emulator is fully transparent to the target system, you have complete control over hardware and software debug and system integration.

64KB of zero wait-state emulation memory is available to replace target system code memory, allowing software debug to begin even before prototype hardware is finished.

FLEXIBLE BREAKPOINTING FOR QUICK PROBLEM ISOLATION

The ICE-5100/452 emulator supports three different types of break specifications: specific address breaks on up to 64,000 possible addresses; range breaks, both within and outside a user-defined range; and page breaks, up to 256 pages on 256-byte boundaries. 254 frames of execution trace memory provide ample debug information, with each frame divided into 16 bits of program execution address and 8 bits of external event information. A maximum of four tracepoints allows qualified trace for a variety of debug conditions.

SYMBOLIC DEBUGGING FOR FAST DEVELOPMENT

Design team productivity is enhanced by the use of symbolic debug references to program line, high-level statements, and module and variable names. The terms used to develop programs are the same used for system debugging.

PATCH CODE WITHOUT RECOMPILING

Code-patching is easy with the ICE-5100/452 emulator's single-line assembler. Machine code can be disassembled to mnemonics for significantly easier debugging and project development.

EASY TO LEARN AND USE

The ICE-5100/452 is accompanied by a full tutorial that explains all system functions and provides many examples. Additional features such as on-line help, a built-in CRT-based editor, and DOS shell escape make the emulator fast and easy to use for both novice and experienced users. You can develop your own test suites or save frequently-used debug routines as debug procedures (PROCs) that can be invoked with a single command.

WORLDWIDE SERVICE AND SUPPORT

The ICE-5100/452 emulator is supported by Intel's worldwide service and support organization. In addition to an extended warranty, you can choose from hotline support, on-site system engineering assistance, and a variety of hands-on training workshops.

ELECTRICAL CONSIDERATIONS

The emulation processor's user-pin timings and loadings are identical to the 452 component except as follows:

- Up to 25 pF of additional pin capacitance is contributed by the processor module and target adaptor assemblies.

PROCESSOR MODULE DIMENSIONS

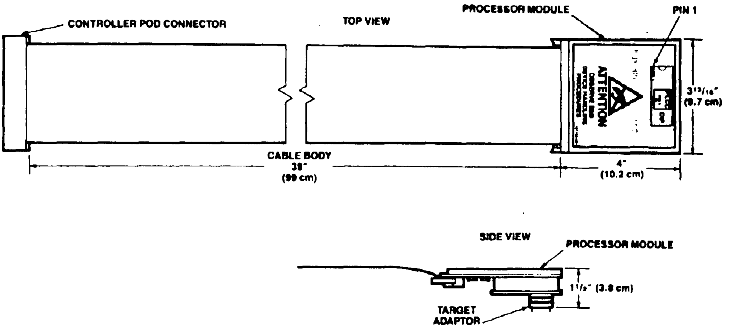


Figure 1: Processor Module Dimensions

SPECIFICATIONS

Host Requirements:

IBM PC-XT, AT or compatible
PC-DOS 3.0 or later
512K RAM
One floppy drive and hard disk

Physical Characteristics:

The ICE-5100/452 emulator consists of the following components:

Unit	Width		Height		Length	
	Inch	Cm	Inch	Cm	Inch	Cm
Controller Pod	8.25	21.0	1.5	3.8	13.5	34.3
User Cable					39.0	99.0
Processor Module*	3.8	9.7	1.5	3.8	4.0	10.2
Power Supply	7.6	18.1	4.0	10.2	11.0	28.0
Serial Cable					144.0	360.0

*with supplied target adapter.

Electrical Characteristics:

Power supply
100-120V or 220-240V selectable
50-60 Hz
2 amps (AC max) @ 120V
1 amp (AC max) @ 240V

Environmental Characteristics:

Operating temperature: + 10°C to + 40°C (50°F to 104°F)
Operating humidity: Maximum of 85% relative humidity, non-condensing

ORDERING INFORMATION

Order Code	Description
pI452KITAD	Kit contains ICE-5100/452 user probe assembly, power supply and cables, serial cables, target adapter, crystal power accessory, emulator controller pod, emulator software, DOS host communication cables, ASM-51 and AEDIT text editor (requires software license).
pI452KITD	Kit contains the same components as pI452KITAD, excluding ASM-51 and the AEDIT text editor (requires software license).
pC452KITD	Conversion kit for ICE-5100/451, ICE-5100/252, or ICE-5100/044 running PC-DOS 3.0 or later, to provide emulation support for 80C452 components (requires software license).
TA452E	Target adapter for 68-pin PLCC package support.
D86ASM51	ASM/RL 51 package for PC-DOS (requires software license).
D86PLM51	PL/M/RL 51 package for PC-DOS (requires software license).
D86EDINL	AEDIT text editor for PC-DOS.

MCS is a registered trademark and ICE is a trademark of Intel Corporation.

IBM and PC/AT are registered trademarks and PC/XT a trademark of International Business Machines Corporation.

For direct information on Intel's Development Tools, or for the number of your nearest sales office or distributor, call 800-874-6835 (U.S.). For information or literature on additional Intel products, call 800-548-4725 (U.S. and Canada).

ICE™-5100/044 In-Circuit Emulator

IN-CIRCUIT EMULATOR FOR THE RUPI™-44 FAMILY OF PERIPHERALS

The ICE-5100/044 In-Circuit Emulator is a complete hardware/software debug environment for developing embedded control applications based on the Intel RUPI™-44 family of peripherals, including the 8044-based BITBUS™ board products. With high-performance 12 MHz emulation, symbolic debugging, and flexible memory mapping, the ICE-5100/044 emulator expedites all stages of development: hardware development, software development, system integration, and system test; shortening your project's time to market.

FEATURES

- Full speed to 12 MHz.
- 64KB of emulation mapped memory.
- 254 frames of execution trace.
- Symbolic debug.
- Serial link to an IBM PC XT, AT, 100% compatible.
- Four address breakpoints with in-range, out-of-range, and page breaks.
- On-line disassembler and single line assembler.
- Source code display.
- ASM-51 and PL/M-51 language support.
- Pop-up help.
- DOS shell escape.
- On-line tutorial.
- Built-in CRT based editor.
- System self-test diagnostics.
- Worldwide service and support.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

September, 1988
Order Number: 280818-001

ONE TOOL FOR ENTIRE DEVELOPMENT CYCLE

The ICE-5100/044 emulator speeds target system development by allowing hardware and software design to proceed simultaneously. You can develop software even before prototype hardware is finished. And because the ICE-5100/044 emulator precisely matches the component's electrical and timing characteristics, it's a valuable tool for hardware development and debug. Thus, the ICE-5100/044 emulator can debug a prototype or production system at any stage in its development, without introducing extraneous hardware or software test tools.

HIGH-SPEED, REAL-TIME EMULATION

The ICE-5100/044 emulator provides full-speed, real-time emulation up to 12 MHz. Because the emulator is fully transparent to the target system, you have complete control over hardware and software debug and system integration.

64KB of zero wait-state emulation memory is available to replace target system code memory, allowing software debug to begin even before prototype hardware is finished.

FLEXIBLE BREAKPOINTING FOR QUICK PROBLEM ISOLATION

The ICE-5100/044 emulator supports three different types of break specifications: specific address breaks on up to 64,000 possible addresses; range breaks, both within and outside a user-defined range; and page breaks, up to 256 pages on 256-byte boundaries. 254 frames of execution trace memory provide ample debug information, with each frame divided into 16 bits of program execution address and 8 bits of external event information. A maximum of four tracepoints allows qualified trace for a variety of debug conditions.

SYMBOLIC DEBUGGING FOR FAST DEVELOPMENT

Design team productivity is enhanced by the use of symbolic debug references to program line, high-level statements, and module and variable names. The terms used to develop programs are the same used for system debugging.

PATCH CODE WITHOUT RECOMPILING

Code-patching is easy with the ICE-5100/044 emulator's single-line assembler. Machine code can be disassembled to mnemonics for significantly easier debugging and project development.

EASY TO LEARN AND USE

The ICE-5100/044 is accompanied by a full tutorial that explains all system functions and provides many examples. Additional features such as on-line help, a built-in CRT-based editor, and DOS shell escape make the emulator fast and easy to use for both novice and experienced users. You can develop your own test suites or save frequently-used debug routines as debug procedures (PROCs) that can be invoked with a single command.

WORLDWIDE SERVICE AND SUPPORT

The ICE-5100/044 emulator is supported by Intel's worldwide service and support organization. In addition to an extended warranty, you can choose from hotline support, on-site system engineering assistance, and a variety of hands-on training workshops.

ELECTRICAL CONSIDERATIONS

The emulation processor's user-pin timings and loadings are identical to the 8044 component except as follows.

- Up to 25 pf of additional pin capacitance is contributed by the processor module and target adaptor assemblies.
- Pin 31, $\overline{EA}$, has approximately 32 pf of additional capacitance loading due to sensing circuitry.
- Pins 18 and 19, XTAL1 and XTAL2, respectively, have approximately 15 to 16 pf of additional capacitance when configured for crystal operation.

DESIGN CONSIDERATIONS

Execution of user programs that contain interrupt routines causes incorrect data to be stored in the trace buffer. When an interrupt occurs, the next instruction to be executed is placed into the trace buffer before it is actually executed. Following completion of the interrupt routine, the instruction is executed and again placed into the trace buffer.

PROCESSOR MODULE DIMENSIONS

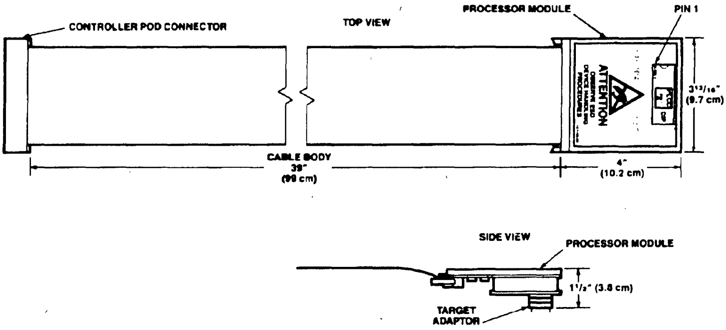


Figure 1. Processor Module Dimensions

SPECIFICATIONS

Host Requirements:

IBM PC-XT, AT or compatible
PC-DOS 3.0 or later
512K RAM
One floppy drive and hard disk

Physical Characteristics:

The ICE-5100/044 emulator consists of the following components:

Unit	Width		Height		Length	
	Inch	Cm	Inch	Cm	Inch	Cm
Controller Pod	8.25	21.0	1.5	3.8	13.5	34.3
User Cable					39.0	99.0
Processor Module*	3.8	9.7	1.5	3.8	4.0	10.2
Power Supply	7.6	18.1	4.0	10.2	11.0	28.0
Serial Cable					144.0	360.0

*with supplied target adaptor.

Electrical Characteristics:

Power supply
100-120V or 220-240V selectable
50-60 Hz
2 amps (AC max) @ 120V
1 amp (AC max) @ 240V

Environmental Characteristics:

Operating temperature: + 10°C to + 40°C (50°F to 104°F)
Operating humidity: Maximum of 85% relative humidity, non-condensing

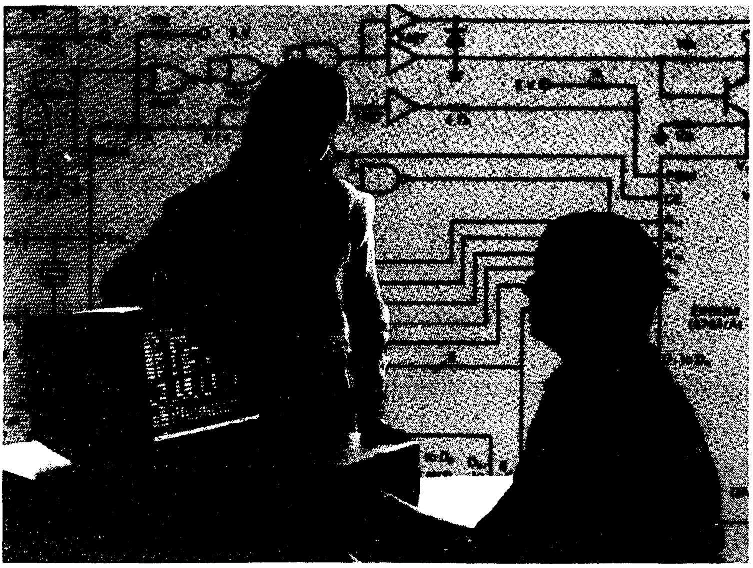
ORDERING INFORMATION

Order Code	Description
pI044KITAD	Kit contains ICE-5100/044 user probe assembly, power supply and cables, serial cables, target adapter, crystal power accessory, emulator controller pod, emulator software, DOS host communication, ASM-51 and AEDIT text editor (requires software license).
pI044KITD	Kit contains the same components as pI044KITAD, excluding ASM-51 and the AEDIT text editor (requires software license).
pC044KITD	Conversion kit for ICE-5100/452, ICE-5100/451, or ICE-5100/252 running PC-DOS 3.0 or later, to provide emulation support for RVPI-44 family of peripherals (requires software license).
D86ASM51	ASM/RL 51 package for PC-DOS (requires software license).
D86PLM51	PL/M/RL 51 package for PC-DOS (requires software license).
D86EDINL	AEDIT text editor for PC-DOS.

MCS is a registered trademark and ICE is a trademark of Intel Corporation.

IBM and PC/AT are registered trademarks and PC/XT a trademark of International Business Machines Corporation.

ICE™-5100/252 In-Circuit Emulator



IN-CIRCUIT EMULATOR FOR THE MCS® 51 FAMILY OF MICROCONTROLLERS

The ICE-5100/252 In-Circuit Emulator is a complete hardware/software debug environment for developing embedded control applications based on the Intel MCS-51 family of microcontrollers. With high-performance 16 MHz emulation, symbolic debugging, and flexible memory mapping, the ICE-5100/252 emulator expedites all stages of development: hardware development, software development, system integration, and system test; shortening your project's time to market.

FEATURES

- Full speed to 16 MHz.
- 64KB of emulation mapped memory.
- 254 frames of execution trace.
- Symbolic debug.
- Serial link to an IBM PC XT, AT, 100% compatible.
- Four address breakpoints with in-range, out-of-range, and page breaks.
- On-line disassembler and single line assembler.
- Source code display.
- ASM-51 and PL/M-51 language support.
- Pop-up help.
- DOS shell escape.
- On-line tutorial.
- Built-in CRT based editor.
- System self-test diagnostics.
- Worldwide service and support.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

September, 1988
Order Number: 280798-002

ONE TOOL FOR ENTIRE DEVELOPMENT CYCLE

The ICE-5100/252 emulator speeds target system development by allowing hardware and software design to proceed simultaneously. You can develop software even before prototype hardware is finished. And because the ICE-5100/252 emulator precisely matches the component's electrical and timing characteristics, it's a valuable tool for hardware development and debug. Thus, the ICE-5100/252 emulator can debug a prototype or production system at any stage in its development, without introducing extraneous hardware or software test tools.

HIGH-SPEED, REAL-TIME EMULATION

The ICE-5100/252 emulator provides full-speed, real-time emulation up to 16 MHz. Because the emulator is fully transparent to the target system, you have complete control over hardware and software debug and system integration.

64KB of zero wait-state emulation memory is available to replace target system code memory, allowing software debug to begin even before prototype hardware is finished.

FLEXIBLE BREAKPOINTING FOR QUICK PROBLEM ISOLATION

The ICE-5100/252 emulator supports three different types of break specifications: specific address breaks on up to 64,000 possible addresses; range breaks, both within and outside a user-defined range; and page breaks, up to 256 pages on 256-byte boundaries. 254 frames of execution trace memory provide ample debug information, with each frame divided into 16 bits of program execution address and 8 bits of external event information. A maximum of four tracepoints allows qualified trace for a variety of debug conditions.

SYMBOLIC DEBUGGING FOR FAST DEVELOPMENT

Design team productivity is enhanced by the use of symbolic debug references to program line, high-level statements, and module and variable names. The terms used to develop programs are the same used for system debugging.

PATCH CODE WITHOUT RECOMPILING

Code-patching is easy with the ICE-5100/252 emulator's single-line assembler. Machine code can be disassembled to mnemonics for significantly easier debugging and project development.

EASY TO LEARN AND USE

The ICE-5100/252 is accompanied by a full tutorial that explains all system functions and provides many examples. Additional features such as on-line help, a built-in CRT-based editor, and DOS shell escape make the emulator fast and easy to use for both novice and experienced users. You can develop your own test suites or save frequently-used debug routines as debug procedures (PROCs) that can be invoked with a single command.

WORLDWIDE SERVICE AND SUPPORT

The ICE-5100/252 emulator is supported by Intel's worldwide service and support organization. In addition to an extended warranty, you can choose from hotline support, on-site system engineering assistance, and a variety of hands-on training workshops.

ICE™-5100/252 Emulator Supported Components

Part	On-Chip Program Memory	On-Chip Data Memory
8031	None	128 bytes
8051	4K ROM	128 bytes
8751	4K EPROM	128 bytes
80C31	None	128 bytes
80C51	4K ROM	128 bytes
87C51	4K EPROM	128 bytes
8032	None	256 bytes
8052	8K ROM	256 bytes
8752	8K EPROM	256 bytes
80C51FA	None	256 bytes
83C51FA	8K ROM	256 bytes
87C51FA	8K EPROM	256 bytes
80C51FB	None	256 bytes
83C51FB	16K ROM	256 bytes
87C51FB	16K EPROM	256 bytes

ELECTRICAL CONSIDERATIONS

The emulation processor's user-pin timings and loadings are identical to the 80C51FA component except as follows.

Maximum Operating ICC and Idle ICC (ma)*

V_{cc}	Maximum Operating ICC (ma)*			Maximum Idle ICC (ma)**		
	4V	5V	6V	4V	5V	6V
Frequency						
0.5 MHz	0.87	1.62	3.0	0.58	1.21	2.5
3.5 MHz	4.8	6.82	9.76	2.2	4.97	6.33
8.0 MHz	10.5	15.0	20.5	6.0	8.98	11.76
12.0 MHz	15.2	22.2	30.2	9.2	13.34	17.46
16.0 MHz	19.4	28.6	38.7	11.8	17.4	23.4

*ICC is measured with all output pins disconnected
XTAL1 driven with TCLCH,TCHCL = 10ns, $V_{il} = V_{ss} + .5V$, $V_{ih} = V_{cc} - .5V$. XTAL2 not connected.
For maximum operating ICC
$\overline{EA}$ = RST = Port0 = V_{cc}.
**For maximum idle ICC
$\overline{EA}$ = Port0 = V_{cc}, RST = V_{cc}, internal clock to PCA gated off.

- Up to 25 pf of additional pin capacitance is contributed by the processor module and target adaptor assemblies.
- Pin 31, $\overline{EA}$, has approximately 32 pf of additional capacitance loading due to sensing circuitry.
- Pins 18 and 19, XTAL1 and XTAL2, respectively, have approximately 15 to 16 pf of additional capacitance when configured for crystal operation.

PROCESSOR MODULE DIMENSIONS

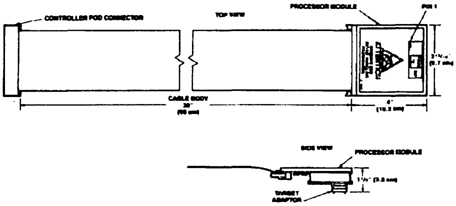


Figure 1. Processor Module Dimensions

CHMOS AND HMOS DESIGN DIFFERENCES

Chip Function	HMOS Component 8031	CHMOS Component 80C31
RST trigger threshold	2.5V	70% V_{cc}(3.5V @ V_{cc} = 5V)
RST input impedance	4K-10K ohms	50K-150K ohms
Port 1_{il}	−800μA	−50μA
Clock threshold	2.5V	70% V_{cc}(3.5V @ V_{cc} = 5V)

SPECIFICATIONS

Host Requirements:

IBM PC-XT, AT or compatible
PC-DOS 3.0 or later
512K RAM
One floppy drive and hard disk

Physical Characteristics:

The ICE-5100/252 emulator consists of the following components:

Unit	Width		Height		Length	
	Inch	Cm	Inch	Cm	Inch	Cm
Controller Pod	8.25	21.0	1.5	3.8	13.5	34.3
User Cable					39.0	99.0
Processor Module*	3.8	9.7	1.5	3.8	4.0	10.2
Power Supply	7.6	18.1	4.0	10.2	11.0	28.0
Serial Cable					144.0	360.0

*with supplied target adaptor.

Electrical Characteristics:

Power supply
100-120V or 220-240V selectable
50-60 Hz
2 amps (AC max) @ 120V
1 amp (AC max) @ 240V

Environmental Characteristics:

Operating temperature: + 10°C to + 40°C (50°F to 104°F)
Operating humidity: Maximum of 85% relative humidity, non-condensing

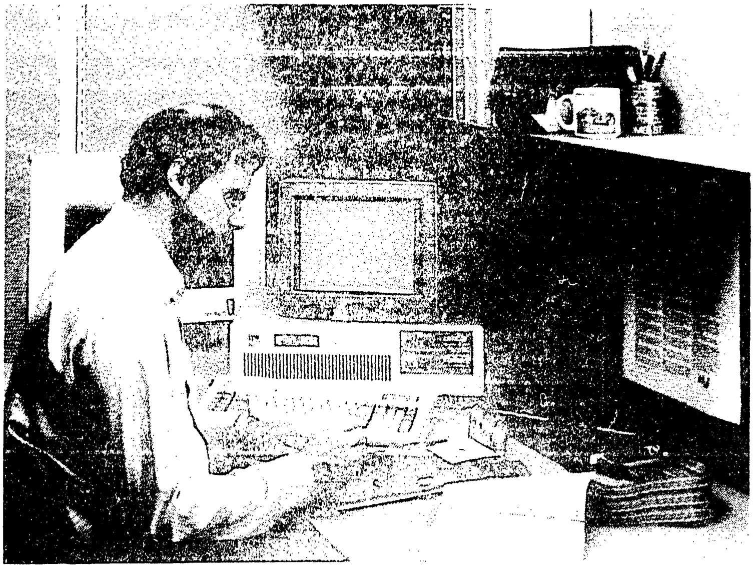
ORDERING INFORMATION

Order Code	Description
pI252KITAD	Kit contains ICE-5100/252 user probe assembly, power supply and cables, serial cables, target adapter, crystal power accessory, emulator controller pod, emulator software, DOS host communication, ASM-51 and AEDIT text editor (requires software license).
pI252KITD	Kit contains the same components as pI252KITAD, excluding ASM-51 and the AEDIT text editor (requires software license).
pC252KITD	Conversion kit for ICE-5100/452, ICE-5100/451, or ICE-5100/044 running PC-DOS 3.0 or later, to provide emulation support for MCS-51 components (requires software license).
TA252D	Target adapter converting 48-pin DIP to 44-pin PLCC package.
D86ASM51	ASM/RL 51 package for PC-DOS (requires software license).
D86PLM51	PL/M/RL 51 package for PC-DOS (requires software license).
D86EDINL	AEDIT text editor for PC-DOS.

MCS is a registered trademark and ICE is a trademark of Intel Corporation.

IBM and PC/AT are registered trademarks and PC/XT a trademark of International Business Machines Corporation.

ICE™-5100/451 In-Circuit Emulator



IN-CIRCUIT EMULATOR FOR THE MCS™-51 FAMILY OF MICROCONTROLLERS

The ICE-5100/451 In-Circuit Emulator is a complete hardware/software debug environment for developing embedded control applications based on the Intel MCS 51 family of microcontrollers. With high-performance 12 MHz emulation, symbolic debugging, and flexible memory mapping, the ICE-5100/451 emulator expedites all stages of development: hardware development, software development, system integration, and system test; shortening your project's time to market.

FEATURES

- Full speed to 12 MHz.
- 64KB of emulation mapped memory.
- 254 frames of execution trace.
- Symbolic debug.
- Serial link to an IBM PC XT, AT, 100% compatible.
- Four address breakpoints with in-range, out-of-range, and page breaks.
- On-line disassembler and single line assembler.
- Source code display.
- ASM-51 and PL/M-51 language support.
- Pop-up help.
- DOS shell escape.
- On-line tutorial.
- Built-in CRT based editor.
- System self-test diagnostics.
- Worldwide service and support.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

September, 1988
Order Number: 280802-002

ONE TOOL FOR ENTIRE DEVELOPMENT CYCLE

The ICE-5100/451 emulator speeds target system development by allowing hardware and software design to proceed simultaneously. You can develop software even before prototype hardware is finished. And because the ICE-5100/451 emulator precisely matches the component's electrical and timing characteristics, it's a valuable tool for hardware development and debug. Thus, the ICE-5100/451 emulator can debug a prototype or production system at any stage in its development, without introducing extraneous hardware or software test tools.

HIGH-SPEED, REAL-TIME EMULATION

The ICE-5100/451 emulator provides full-speed, real-time emulation up to 12 MHz. Because the emulator is fully transparent to the target system, you have complete control over hardware and software debug and system integration.

64KB of zero wait-state emulation memory is available to replace target system code memory, allowing software debug to begin even before prototype hardware is finished.

FLEXIBLE BREAKPOINTING FOR QUICK PROBLEM ISOLATION

The ICE-5100/451 emulator supports three different types of break specifications: specific address breaks on up to 64,000 possible addresses; range breaks, both within and outside a user-defined range; and page breaks, up to 256 pages on 256-byte boundaries. 254 frames of execution trace memory provide ample debug information, with each frame divided into 16 bits of program execution address and 8 bits of external event information. A maximum of four tracepoints allows qualified trace for a variety of debug conditions.

SYMBOLIC DEBUGGING FOR FAST DEVELOPMENT

Design team productivity is enhanced by the use of symbolic debug references to program line, high-level statements, and module and variable names. The terms used to develop programs are the same used for system debugging.

PATCH CODE WITHOUT RECOMPILING

Code-patching is easy with the ICE-5100/451 emulator's single-line assembler. Machine code can be disassembled to mnemonics for significantly easier debugging and project development.

EASY TO LEARN AND USE

The ICE-5100/451 is accompanied by a full tutorial that explains all system functions and provides many examples. Additional features such as on-line help, a built-in CRT-based editor, and DOS shell escape make the emulator fast and easy to use for both novice and experienced users. You can develop your own test suites or save frequently-used debug routines as debug procedures (PROCs) that can be invoked with a single command.

WORLDWIDE SERVICE AND SUPPORT

The ICE-5100/451 emulator is supported by Intel's worldwide service and support organization. In addition to an extended warranty, you can choose from hotline support, on-site system engineering assistance, and a variety of hands-on training workshops.

ELECTRICAL CONSIDERATIONS

The emulation processor's user-pin timings and loadings are identical to the 80C451 component except as follows.

Maximum Operating ICC and Idle ICC (ma)*

V_{cc}	Maximum Operating ICC (ma)*	Maximum Idle ICC (ma)**
	5V	5V
0.5 MHz	4.97	4.93
3.5 MHz	9.53	10.39
8.0 MHz	13.0	12.75
12.0 MHz	22.2	18.19

*ICC is measured with all output pins disconnected
XTAL1 driven with TCLCH,TCHCL = 5ns, $V_{il} = V_{ss} + 0.5V$,
$V_{ih} = V_{cc} - 0.5V$. XTAL2 not connected.
For maximum operating ICC
$\overline{EA}$ = RST = Port0 = V_{cc}.

**Maximum idle ICC is measured with all output pins disconnected.
XTAL1 driven with TCLCH, TCHCL = 5ns
$V_{il} = V_{ss} + 0.5V$,
$V_{il} = V_{cc} - 0.5V$,
XTAL2 not connected,
Port 0 = V_{cc}.
$\overline{EA}$ = RST = V_{cc}.
Up to 25 pf of additional pin capacitance is contributed by the processor module and target adapter assemblies. Pin 1 (EA) has approximately 32 pf of additional capacitance loading due to sensing circuitry. Pins 53 and 52, XTAL1 and XTAL2, respectively, have approximately 15 to 16 pf of additional capacitance when configured for crystal operation.

PROCESSOR MODULE DIMENSIONS

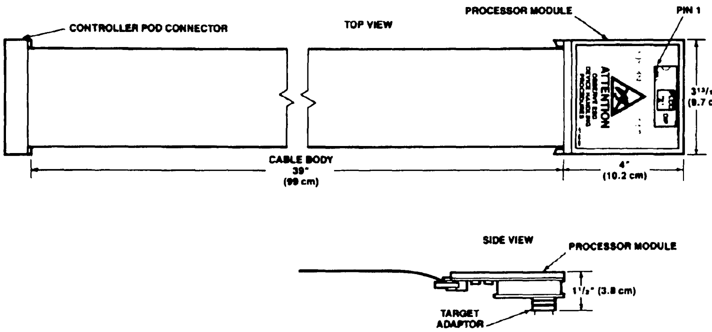


Figure 1: Processor Module Dimensions

CHMOS DESIGN DIFFERENCES

Chip Function	CHMOS Component 80C31
RST trigger threshold	70% V_{cc}(3.5V @ V_{cc} = 5V)
RST input impedance	50K-150K ohms
Port 1_{il}	$-50\mu A$
Clock threshold	70% V_{cc}(3.5V @ V_{cc} = 5V)

SPECIFICATIONS

Host Requirements:

IBM PC-XT, AT or compatible
PC-DOS 3.0 or later
512K RAM
One floppy drive and hard disk

Physical Characteristics:

The ICE-5100/451 emulator consists of the following components:

Unit	Width		Height		Length	
	Inch	Cm	Inch	Cm	Inch	Cm
Controller Pod	8.25	21.0	1.5	3.8	13.5	34.3
User Cable					39.0	99.0
Processor Module*	3.8	9.7	1.5	3.8	4.0	10.2
Power Supply	7.6	18.1	4.0	10.2	11.0	28.0
Serial Cable					144.0	360.0

*with supplied target adapter.

Electrical Characteristics:

Power supply
100-120V or 220-240V selectable
50-60 Hz
2 amps (AC max) @ 120V
1 amp (AC max) @ 240V

Environmental Characteristics:

Operating temperature: + 10°C to + 40°C (50°F to 104°F)
Operating humidity: Maximum of 85% relative humidity, non-condensing

ORDERING INFORMATION

Order Code	Description
pl451KITAD	Kit contains ICE-5100/451 user probe assembly, power supply and cables, serial cables, target adapter, crystal power accessory, emulator controller pod, emulator software, DOS host communication cables, ASM-51 and AEDIT text editor (requires software license).
pl451KITD	Kit contains the same components as pl451KITAD, excluding ASM-51 and the AEDIT text editor (requires software license).
pC451KITD	Conversion kit for ICE-5100/452, ICE-5100/252, or ICE-5100/044 running PC-DOS 3.0 or later, to provide emulation support for 80C451 components (requires software license).
TA451E	Target adapter for 68-pin PLCC package support.
D86ASM51	ASM/RL 51 package for PC-DOS (requires software license).
D86PLM51	PL/M/RL 51 package for PC-DOS (requires software license).
D86EDINL	AEDIT text editor for PC-DOS.

MCS is a registered trademark and ICE is a trademark of Intel Corporation.

IBM and PC/AT are registered trademarks and PC/XT a trademark of International Business Machines Corporation.

VLSiCE™-96 IN-CIRCUIT EMULATOR

IN-CIRCUIT EMULATOR FOR THE 8X9X FAMILY OF MICROCONTROLLERS

The VLSiCE™-96 emulator is a complete hardware/software debug environment for developing systems based on the Intel 8x9x family of microcontrollers. The VLSiCE-96 emulator supports all NMOS members of Intel's MCS-96 microcontrollers, including the 8096BH, the 8098, the 8095, the 8097, and the 8096-90. With high performance 12 MHz emulation, symbolic debugging, and flexible memory mapping, the VLSiCE-96 emulator expedites all stages of development: software development, hardware development, system integration and system test.

FEATURES

- Real-time transparent emulation, up to 12 MHz
- 64K of mappable memory to allow early software debug and (EP)ROM simulation, even before any target hardware is available.
- Trace contains execution address, opcode, symbolics, and bus information
- 4K frame trace buffer for storing real-time execution history
- Ability to break or trace on execution addresses, opcodes, data values, or flags values
- Symbolic debugging for faster and easier access to memory location and program variables.
- Fast breaks and dynamic trace to allow the user to modify and interrogate memory, and access the trace buffer without stopping emulation.
- On-line Help file to speed development
- Shadow Registers can read many write-only registers and write to many read-only registers, allowing enhanced debugging over component features
- Includes 68-pin PGA adaptor; optional 68-pin PLCC and 48-pin DIP adaptors are also available
- Serially hosted on IBM PC AT/XT or compatibles with DOS 3.0 or greater.

intel

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

September, 1988

Order Number: 280794-002

ONE TOOL FOR ENTIRE DEVELOPMENT CYCLE

The VLSiCE-96 emulator speeds target system development by allowing hardware and software design to proceed simultaneously. You can develop software even before prototype hardware is finished. And because the VLSiCE-96 emulator precisely matches the component's electrical and timing characteristics, it's a valuable tool for hardware development and debug.

The VLSiCE-96 emulator also simplifies and expedites system integration and test. As each section of the hardware is completed, it is simply added to the prototype and tested in real-time. When the prototype is complete, it is tested with the final version of the system software. The VLSiCE-96 emulator can then be used to verify or debug the target system as a completed unit.

Because it supports the ROMless, ROM and EPROM versions of Intel's microcontrollers, the VLSiCE-96 emulator can debug a prototype or production product at any stage in its development without introducing extraneous hardware or software test tools.

SPECIFICATIONS

HOST REQUIREMENTS

An IBM PC AT/XT or compatible with 512K bytes RAM and hard disk. Intel recommends an IBM PC AT or compatible with 640K bytes of RAM, one floppy drive and one hard disk running PC-DOS 3.1 or later.

System Performance

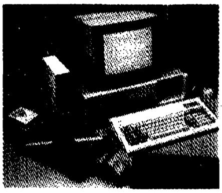
Mappable zero wait state (up to 12 MHz) Min 0K bytes, Max 64K bytes	Mappable to user memory or ICE memory in 1K blocks on 1K boundaries
Trace Buffer	4K bytes × 48 bits
Virtual Symbol Table	A maximum of 61K bytes of host memory space is available for the virtual symbol table (VST). The rest of the VST resides on disk and is paged in and out of host memory as needed.

Electrical Characteristics

Power Supply
100V-120V or 200V-240V (selectable)
50 Hz-60 Hz
2 amps (AC max) @ 120V
1 amp (AC max) @ 240V

Physical Characteristics

Controller Pod	
Width:	8¼" (21 cm)
Height:	1½" (4 cm)
Depth:	13½" (34 cm)
Weight:	4 lbs (2 kg)
Power Supply	
Width:	7⅝" (18 cm)
Height:	4" (10 cm)
Depth:	11" (28 cm)
Weight:	15 lbs (7 kg)
User Cable:	3' (1 m)



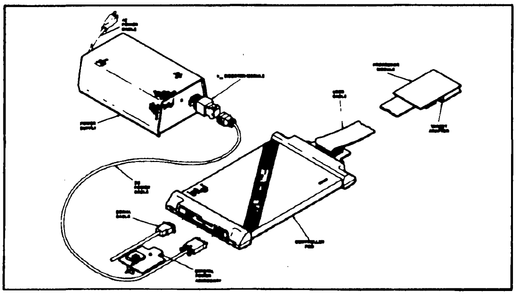


FIGURE 1. The VLSiCE-96™ Emulator

Environmental Characteristics

Operating Temperature: 0°C to +40°C (−32°F to +104°F)
Operating Humidity: Maximum to 85% relative humidity, non condensing

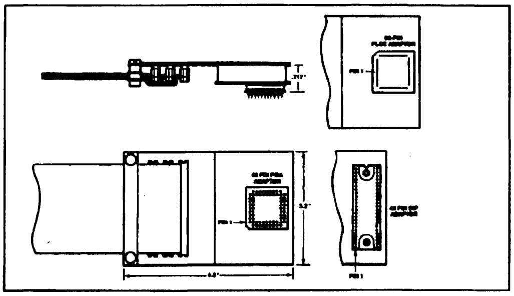


FIGURE 2. Dimensions for the Emulator Processor Board and Adaptors

SERVICE SUPPORT AND TRAINING

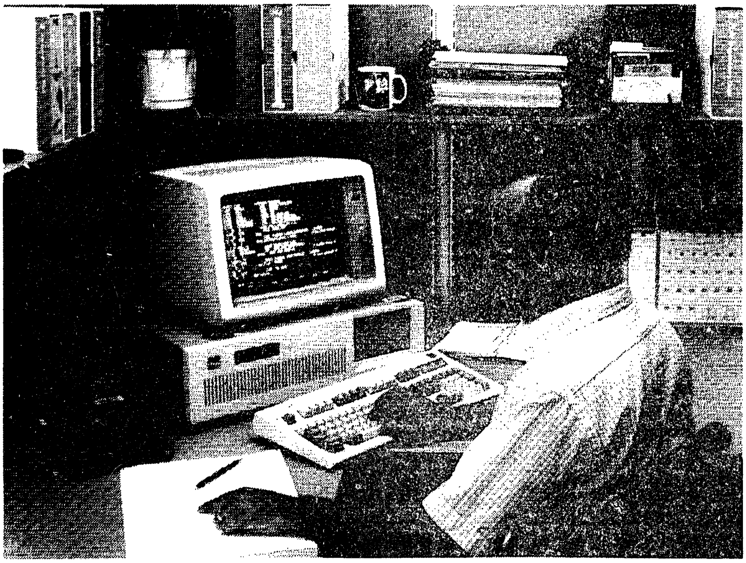
Intel augments its MCS-96 architecture family development tools with a full array of seminars, classes and workshops; on-site consulting services; field application engineering expertise; telephone hot-line support; and software and hardware maintenance contracts. This full line of services will ensure your design success.

ORDERING INFORMATION

V096KITA	VLSiCE-96 Power supply cable, emulation base, user cable, Crystal Power Accessory (CPA), serial cables for PC AT/XT, a 68-pin PGA target adaptor, ASM-96, AEDIT Text Editor. Host, probe, diagnostic and tutorial software on 5¼″ media for DOS hosts running DOS V3.0 or later. (Requires software license.)
V096KITD	Same as V096KITA without ASM-96 and AEDIT text editor.
TA096E	Optional 68-pin PLCC Target Adaptor Board
TA096B	Optional 48-pin DIP Target Adaptor Board
MSA96	Optional Multi-Synchronous Accessory for multi-ICE capability
SA096D	Software for host, probe, diagnostic and tutorial on 5¼″ media for use with the PC AT/XT under PC-DOS V3.0 or later. (Requires software license.) (Included with V096KITA and V096KITD.)
D86C96NL	C-96 Compiler*
D86PLM96NL	PL/M-96 Compiler*
D86ASM96NL	ASM-96 Macroassembler*

*Also Includes: Relocator/Linker, object-to-hex converter, librarian, and Floating Point Arithmetic Library.

REAL-TIME TRANSPARENT 80C196 IN-CIRCUIT EMULATOR



REAL-TIME TRANSPARENT 80C196 IN-CIRCUIT EMULATOR

The ICE™-196KB/PC in-circuit emulator delivers real-time high-level debugging capabilities for developing, integrating and testing 80C196-based designs. Operating at the full speed of the 80C196KB microcontroller, the ICE-196KB/PC provides precise I/O pin timings and functionality. The ICE-196KB/PC also allows you to develop code before prototype hardware is available. The in-circuit emulator represents a low-cost development environment for designing real-time microcontroller-based applications with minimal investment in time and resources.

ICE™-196KB/PC IN CIRCUIT EMULATOR FEATURES

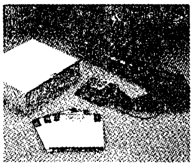
- Real-Time Emulation of the 80C196KB Microcontroller
- 64K Bytes of Mappable Memory
- 2K-entry Trace Buffer
- 3 Breakpoints or 1 Range Break
- Symbolic Support and Source Code Display
- Standalone Operation
- Versatile and Powerful Host Software
- Hosted On IBM PC XT, AT* or Compatibles With DOS 3.0 or Later

REAL-TIME EMULATION

The ICE-196KB/PC provides real-time emulation with the precise input/output pin timings and functions across the full operating frequencies of the 80C196KB microcontroller. The ICE-196KB/PC connects to the intended 80C196KB microcontroller socket via a 16" flex cable, which terminates in a 68-pin PLCC probe. An optional 68-pin PGA adapter is also available.

MAPPABLE MEMORY

The ICE-196KB/PC has 64K bytes (65,536) of zero wait-state memory that can be enabled or mapped as read-only, write-only or read/write in 4K byte increments to simulate the internal (EP)ROM of the 80C196KB or external program memory.



*PC XT, AT are trademarks of IBM.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

January, 1989
Order Number 280727-004

TRACE BUFFER

The ICE-196KB/PC contains a 2K (2048) entry trace buffer for keeping a history of actual instruction execution. The trace buffer can be conditionally turned off to collect a user-specified number of trace frames. Trace information can be displayed as disassembled instructions or, optionally, disassembled instructions and the original C-96 and PL/M-96 source code.

BREAK SPECIFICATION

Three execution address breakpoints or one range of addresses can be active at any time. The ICE-196PC allows any number of breakpoints to be defined and activated when needed.

SYMBOLIC SUPPORT AND SOURCE CODE DISPLAY

Full ASM-96, PL/M-96 and C-96 language symbolics, including variable typing and scope, are supported by the ICE-196KB/PC memory accesses, trace buffer display, breakpoint specification, and assembler/disassembler. Additionally, C-96 and PL/M-96 source code can be displayed to make development and debug easier.

STANDALONE OPERATION

Product software can be developed prior to hardware availability with the optional Crystal Power Accessory (CPA) and the ICE-196KB/PC mappable memory. The CPA also provides diagnostic testing to assure full functionality of the ICE-196KB/PC.

VERSATILE AND POWERFUL HOST SOFTWARE

The ICE-196KB/PC comes equipped with an on-line help facility, a dynamic command entry and syntax guide, built-in editor, assembler and disassembler, and the ability to customize the command set via literal definitions and debug procedures.

HOSTING

The ICE-196KB/PC is hosted on the IBM PC XT, AT or compatibles with PC-DOS 3.0 or later.

SPECIFICATIONS

REQUIREMENTS

Host

IBM PC XT, AT (or compatible)
512K bytes RAM, Hard Disk
PC-DOS 3.0 or Later
One Unused Peripheral Slot
DC Current 2.5A
ICE-196KB/PC 2 Bytes of User Stack Space

TARGET INTERFACE BOARD

Length	2.0" (5.1cm)
Height	1.2" (3.0cm)
Width	2.3" (5.8cm)

USER CABLE

Length	15.6" (39.6cm)

PROBE ELECTRICAL

80C196KB plus per pin	50pf loading 5ns propagation delay
Icc (from target system)	50mA @ 12 MHz
Operating Frequency	3.5 to 12 MHz, 12 MHz only with CPA

ENVIRONMENTAL CHARACTERISTICS

Operating Temperature	10°C to 40°C 37.5°F to 104°F
Operating Humidity	Maximum 55% Relative Humidity, non-condensing

Note: ICE-196KB/PC uses two bytes of the user stack.

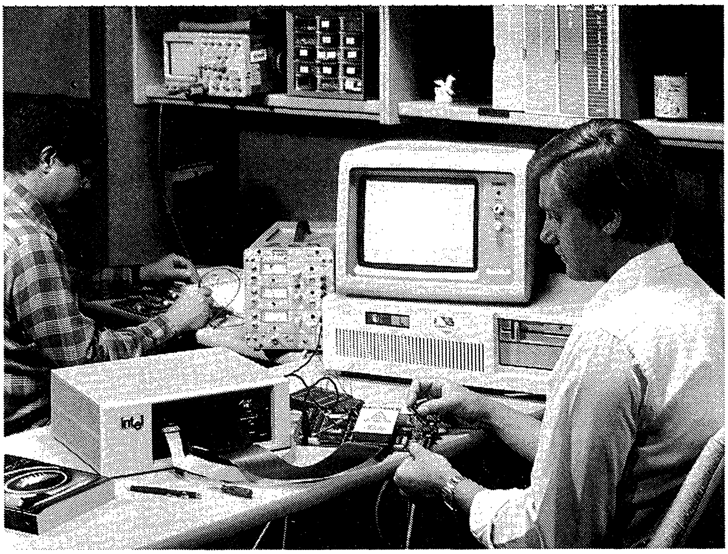
ORDERING INFORMATION

Order Code	Description
ICE196KBPC	Emulation Board, user cable, target interface board (PLCC), host, diagnostic, and tutorial software on 5¼" DOS diskette, and Crystal Power Accessory with power cable
ICE196KBPCB	Same as above except does not include Crystal Power Accessory
TA196PLCC68PGA	68 pin PGA target adapter
CPA196KAKB	Crystal Power Accessory and power cable only
D86C96NL	C-96 Compiler*
D86PLM96NL	PL/M-96 Compiler*
D86ASM96NL	ASM-96 Assembler*

*Includes: Relocator/Linker, Object-to-hex Converter, Floating Point Arithmetic Library, Librarian

PRELIMINARY

ICE™-196KB/xX IN-CIRCUIT EMULATORS

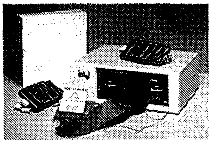


MODULAR IN-CIRCUIT EMULATORS FOR THE 8xC196KB FAMILY OF MICROCONTROLLERS

The ICE™-196KB/MX and ICE-196KB/HX in-circuit emulators deliver a complete, real-time, hardware/software debug environment for developing, integrating, and testing 8xC196KB-based designs. The ICE-196KB/MX emulator is a mid-range modular debugging system featuring high performance 12 MHz emulation, high-level symbolic debugging, 64k bytes zero-waitstate mappable memory, and emulation trace. ICE-196KB/HX emulator is a high-end system with all the functionality of ICE-196KB/MX plus additional break/trace capabilities and expanded mappable memory. The ICE-196KB/MX emulator can be upgraded to an ICE-196KB/HX emulator with optional add-in boards. Both systems feature an identical human interface, utilize the same base chassis, and are serially hosted on IBM* PC XTs and AT's, and 100% compatibles.

ICE-196KB/xX IN-CIRCUIT EMULATORS CORE FEATURES

- **Precisely** matches the component's electrical and timing characteristics
- Supports the ROMless and (EP)ROM versions of the 8xC196KB
- Does not introduce extraneous hardware or software overhead
- Modular base for future growth and migration



*IBM is a registered trademark of International Business Machines.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

October, 1988

Order Number: 280847-001

ICE™-196KB/MX IN-CIRCUIT EMULATOR FEATURES

- Real-time transparent emulation of the 8xC196KB microcontroller family up to 12 MHz, including ROM and EPROM versions
- 64k bytes of zero-waitstate mappable memory to allow early software debug and (EP)ROM simulation, expandable to 128k bytes
- 64k hardware execution breakpoints
- Symbolic debugging and source code display for faster and easier access to memory location and program variables
- 2k frame trace buffer displaying execution address
- Multi-ICE synchronization to start and stop multiple emulators in multi-processor designs
- Run-time viewable execution trace
- Watch window feature automatically displays variables when breaking emulation
- Serially-hosted (RS232C) with very high-speed download capability
- ONCE™ support for on-circuit emulation of surface-mount target systems
- Trigger out for synchronization with external logic analyzer or scope
- Capable of suspension mounting for remote debug
- Full language support with ASM-96, PL/M-96, and C-96
- On-line disassembler and single-line assembler
- Context-sensitive drop-down "help" window to speed development
- On-line tutorial
- Self-test diagnostics to ensure system integrity
- World-wide service and support

ICE™-196KB/HX IN-CIRCUIT EMULATOR FEATURES

Includes all features in ICE™-196KB/MX emulator *plus the following:*

- Additional complex event recognizers for bus break/trace to allow debugging on data values, events, or addresses
- Dynamic trace allows user to view trace buffer without stopping emulation
- Fastbreaks to allow the user to access program variables and SFRs during emulation
- Additional 64k bytes of zero-waitstate mapped memory (128k bytes total)
- Emulation timer and event timer for debugging speed-critical applications and to allow performance analysis capabilities
- Conditional trace to allow tracing under user-specified conditions
- Asynchronous external break capability
- Full multi-ICE communication for enhanced debugging in multi-processor designs
- Input and output logic clips for external logic analysis and control
- 20k bytes additional trace buffer displaying execution address, bus address, bus data, bus status, and clips in
- Run-time reprogrammable break/trace points

COMPLETE FAMILY OF 8x196 DEVELOPMENT TOOLS

ICE-196KB/MX and ICE-196KB/HX emulators are complemented by Intel's low-cost ICE-196KB/PC emulator. All three emulators utilize an upward-compatible human interface to preserve your learning investment and to allow multiple emulators for large design teams. Each emulator has been designed to work in conjunction with Intel's MCS-96 software tools, including a macroassembler, a PL/M-96 compiler, a C-96 compiler, and various utilities.

Optional boards are available to upgrade an ICE-196KB/MX emulator with some or all of the functionality of an ICE-196KB/HX emulator. In addition, the ICE-196KB/MX and ICE-196KB/HX emulators have been designed to support future proliferations within the 8xC196 family of microcontrollers.

WORLDWIDE SERVICE AND SUPPORT

Intel augments its MCS-96 architecture family development tools with a full array of seminars, classes, and workshops; on-site consulting services; field application engineering expertise; telephone hot-line support; and software and hardware maintenance contracts. This full line of services will ensure your design success.

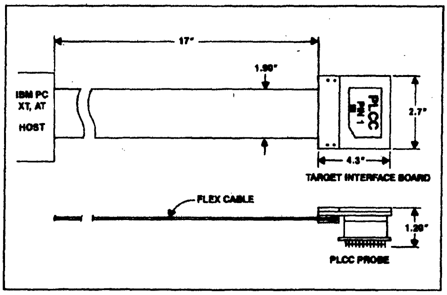
SPECIFICATIONS

HOST REQUIREMENTS

Emulators require an IBM PC AT/XT (or 100% compatible) with 512k bytes RAM and hard disk running DOS 3.1 or higher. Intel recommends 640k bytes of RAM.

ELECTRICAL CHARACTERISTICS

Power Supply: 100V-120V or 200V-240V
50 Hz–60 Hz
5 amps (AC max) @ 120V
2 amps (AC max) @ 240V



Dimensions for the Emulator Processor Board and Adapters

PHYSICAL CHARACTERISTICS

Target Probe

Width: 6.9 cm (2.7")
Height: 3.0 cm (1.2")
Length: 11.0 cm (4.3")
Package: 68-pin PLCC (optional 68-pin PGA flexible adapter available)

Emulator Chassis		***Power Supply***	
Width:	34 cm (13³/₈")	Width:	18 cm (7¹/₂")
Height:	12 cm (4¹/₂")	Height:	10 cm (4")
Depth:	25 cm (9⁷/₈")	Depth:	28 cm (11")
Weight:	3.2 kg (7 lb)	Weight:	7 kg (15 lb)

Probe Cable Length: 40 cm (17")
Serial Cable Length: 3.65 m (12')

ENVIRONMENTAL CHARACTERISTICS

Operating Temperature: 0°C to 40°C
Operating Humidity: Maximum 85% relative humidity, non-condensing

ORDERING INFORMATION

ICE196KBHX ICE in-circuit emulator base chassis, 196 emulation control board (ECB), 196KB target probe, 196KB crystal power accessory (CPA), enhanced break/trace board (BTB), 64k optional memory board (OMB), clips in/out, power supply and cable, serial cables for PC XT/AT, 68-pin PLCC target adapter. Host, 196KB probe, diagnostic, and tutorial software on 5¼" media for DOS hosts running DOS 3.1 or later. (Requires software license.)

ICE196KBMX Same as ICE196KBHX except without enhanced break/trace board (BTB), without 64k optional memory board (OMB), and without clips in/out

ICEBTB Enhanced break/trace board (BTB) for upgrading an ICE-196KB/MX system

ICEOMB Optional memory board with 64k zero-waitstate mapped memory for upgrading an ICE-196KB/MX system

ICECLIPS Clips in/out for upgrading an ICE-196KB/MX system (requires an enhanced break/trace board)

TA196PLCC68PGA 68-pin PGA target adapter

ICEXONCE Target adapter for ONCE (on-circuit) emulation

ICE196KBPC ICE-196KB/PC—PC form-factor in-circuit emulator

D86ASM96NL ASM-96 macroassembler*

D86C96NL C-96 compiler*

D86PLM96NL PL/M-96 compiler*

*Also includes: Relocator/Linker, object-to-hex converter, librarian, and floating point arithmetic library

For direct information on Intel's Development Tools, or for the number of your nearest sales office or distributor, call 800-874-6835 (U.S.). For information or literature on additional Intel products, call 800-548-4725 (U.S. and Canada).

ICE™-186 IN-CIRCUIT EMULATOR

HIGH PERFORMANCE REAL-TIME EMULATION

Intel's ICE-186 emulator delivers real-time emulation for the 80C186 microprocessor at speeds up to 12.5 MHz. The in-circuit emulator is a versatile and efficient tool for developing, debugging and testing products designed with the Intel 80C186 microprocessor. The ICE-186 emulator provides real time, full speed emulation in a user's system. Popular features such as symbolic debug, 2K bytes trace memory, and single-step program execution are standard on the ICE-186 emulator. Intel provides a complete development environment using assembler (ASM86) as well as high-level languages such as Intel's iC86, PL/M86, Pascal 86 and Fortran 86 to accelerate development schedules.

The ICE-186 emulator supports a subset of the 80C186 features at 12.5 MHz and at the TTL level characteristics of the component. The emulator is hosted on IBM's Personal Computer AT, already available as a standard development solution in most of today's engineering environments. The ICE-186 emulator operates in prototype or standalone mode, allowing software development and debug before a prototype system is available. The ICE-186 emulator is ideally suited for developing real-time applications such as industrial automation, computer peripherals, communications, office automation, or other applications requiring the full power of the 12.5 MHz 80C186 microprocessor.

ICE™-186 FEATURES

- Full 12.5 MHz Emulation Speed
- 2K Frames Deep Trace Memory
- Two-Level Breakpoints with Occurrence Counters
- Single-Step Capability
- 128K Bytes Zero Wait-State Mapped Memory
- Supports DRAM Refresh
- High-Level Language Support
- Symbolic Debug
- RS-232-C and GPIB Communication Links
- Crystal Power Accessory
- Interface for Intel Performance Analysis Tool (iPAT)
- Interface for Optional General Purpose Logic Analyzer
- Tutorial Software
- Complete Intel Service and Support

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

September, 1988

HIGHEST EMULATION SPEED AVAILABLE TODAY

The ICE-186 emulator supports development and debug of time-critical hardware and software using Intel's 12.5 MHz 80C186 microprocessor.

RETRACE SOFTWARE TRACKS

This emulator captures up to 2,048 frames of processor activity, including both execution and data bus activity. With this trace memory, large blocks of program code can be traced in real time and viewed for program flow and behavior characteristics.

HARDWARE BREAKPOINTS FOR COMPLEX DEBUG

User-defined "TIL-THEN" breakpoint statements stop emulation at specific execution addresses or bus events. During the hardware and software integration phase, breakpoint statements can be defined as execution addresses and/or bus addresses and/or bus access types such as memory and I/O reads or writes. Additionally, event counters provide another level of breakpoint control for sophisticated state machine constructs used to specify emulation breakpoints/tracepoints.

SMALL OR LARGE STEPS

A stepping command can be used to view program execution one instruction at a time or in preset instruction blocks. When used in conjunction with symbolic debug, code execution can be monitored quickly and precisely.

DEBUG CODE WITHOUT A PROTOTYPE

Even before prototype hardware is available, the ICE-186 emulator working in conjunction with the Crystal Power Accessory (CPA) creates a "virtual" application environment. 128K bytes of zero wait-state memory is available for mapped memory and I/O resource addressing in 4K increments. The CPA provides emulator diagnostics as well as the ability to use the emulator without a prototype.

DON'T LOSE MEMORY

The ICE-186 emulator continues DRAM refresh signals even when emulation has been halted, thus ensuring DRAM memory will not be lost. During interrogation mode the ICE-186 emulator will keep the timers functioning and correctly respond to interrupts in real-time.

HIGH LEVEL LANGUAGE SUPPORT OPTIMIZED FOR INTEL TOOLS

The ICE-186 supports emulation for programs written in Intel's ASM86 or any of Intel's high-level languages:

PL/M-86	Fortran-86
Pascal-86	C-86

These languages are optimized for the Intel 80186/80188 component architectures to deliver a tightly integrated, high performance development environment.

USER-FRIENDLY SYMBOLICS AID IN DEBUG

Symbolics allow access to program symbols by name rather than cumbersome physical addresses. Symbolic debug speeds the debugging process by reducing reliance on memory maps. In a dynamic development process, user variables can be used as parameters for ICE-186 commands resulting in a consistent debug environment.

SUPPORTS FAST BREAKS

"Fastbreaks" is a feature which allows the emulation processor to halt, access memory, and return to emulation as quickly as possible. A fastbreak never takes more than 5625 clock cycles (most types of fastbreaks are considerably less). This feature is particularly useful in embedded applications.

MULTIPLE HIGH-SPEED COMMUNICATION LINKS

Two communication links are available for use in conjunction with the host IBM PC AT. The ICE-186 emulator uses either serial (RS-232-C) or a parallel (GPIB) link. A user supplied National Instruments (IEEE-488) GPIB communication board provides parallel transfers at rates up to 300K bytes per second.

SOFTWARE ANALYSIS (iPAT)

Intel's Performance Analysis Tool (iPAT) is designed to increase team productivity with features like interrupt latency measurement, code coverage analysis and software module performance analysis. These features enable the user to design reliable, high performance embedded control products. The ICE-186 emulator has an external 60 pin connector for iPAT.

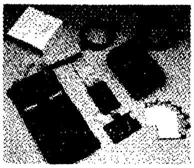
BUILT-IN SUPPORT FOR LOGIC ANALYSIS

General-purpose logic analyzers can be used in conjunction with the ICE-186 to provide detailed timing of specific events. The ICE-186 emulator provides an external sync signal for triggering logic analysis, making complex trigger sequence programming easy. An additional 60 pin connector is included for the logic analyzer.

WORLDWIDE SERVICE AND SUPPORT

The ICE-186 emulator is supported by Intel's worldwide service and support organization. Total hardware and software support is available including a hotline number when the need is there.

Note: This emulator does not support use of the 8087.



SPECIFICATIONS

PERSONAL COMPUTER REQUIREMENTS

The ICE-186 emulator is hosted on an IBM PC AT. The emulator has been tested and evaluated on an IBM PC AT. The PC AT must meet the following minimum requirements:

- 640K Bytes of Memory
- Intel Above Board with at Least 1M Byte of Expansion Memory
- One 360K Bytes or One 1.2M Bytes floppy Disk Drive
- One 20M Bytes Fixed-Disk Drive
- PC DOS 3.2 or Later
- A serial Port (COM1 or COM2) Supporting Minimally at 9600 Baud Data Transfers, or a National Instruments GPIB-PC2A board.
- IBM PC AT BIOS

PHYSICAL DESCRIPTION AND CHARACTERISTICS

The ICE-186 Emulator consists of the following components:

Unit	Width		Height		Length	
	Inches	Cm.	Inches	Cm.	Inches	Cm.
Emulator Control Unit	10.40	26.40	1.70	4.30	20.70	52.60
Power Supply	7.60	19.00	4.15	10.70	11.00	27.90
User Probe	3.70	9.40	.65	1.60	7.00	17.80
User Cable/ Plcc					22.00	55.90
Hinge Cable					3.40	8.60
Crystal Power Accessory	4.30	10.90	.60	1.50	6.70	17.00
CPA Power Cable					9.00	22.90

ELECTRICAL CONSIDERATIONS

Icc 1050mA

I_{IH} 70μA Max.

I_{IL} − 1.5mA Max

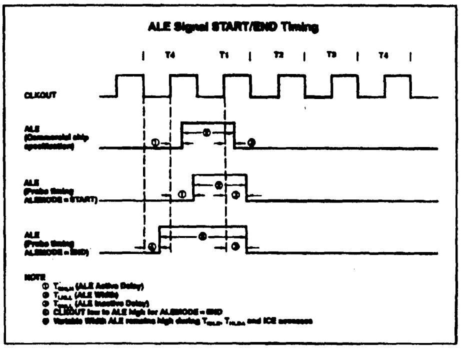
I_{OH} − 1.0mA Max.

TIMING CONSIDERATIONS

ICE-186 User AC Differences					
		COMPONENT SPEC		ICE-186 SPEC	
Symbol	Parameter	Min.	Max.	Min.	Max.
T_{DVCL}	Data in Setup (A/D)	15		24	
T_{ARYCH}	Async Ready (ARDY) Resolution Transition Setup Time	15		23	
T_{SRYCL}	Synchronous Ready (SRDY) Transition Setup Time	15		25	
T_{HVCL}	HOLD Setup	15		32	
T_{INVCH}	NMI	15		32	
	/TEST	15		31	
	INTR,TIMERIN Setup Time	15		17	
T_{INVCL}	DRQ0, DRQ1, Setup Time	15		19	
T_{CLAZ}	Address Float Delay				
	READ Cycles	T_{CLAX}	25	5	36
	INTA cycles	T_{CLAX}	25	0	25
	HLDA	T_{CLAX}	25	10	50
T_{LHLL}	ALE Width (min)	T_{CLCL}-30		T_{CLCL}-32	
T_{CHLH}	ALE Active Delay*		25		42
(N/A)	CLKOUT Low to ALE Active**		(N/A)		19
T_{CHLL}	ALE Inactive Delay		25		40
T_{LLAX}	Address Hold to ALE Inactive (min)	T_{CHCL}-15		T_{CHCL}-28	
T_{CVCTV}	Control Inactive Delay	5	37	1	40
T_{AZRL}	Address Float to /RD Active	0		−30	
T_{AVLL}	Address Valid to ALE Low (min)	T_{CLCH}-15		T_{CLCH}-19	
T_{CHQSV}	Que Status Delay		28		35
T_{DXDL}	/DEN Inactive to DT/R Low	0		−7	
T_{CICO}	CLKIN to CLKOUT Skew		21		37
Consult User Guide for Additional Specifications.					

* Applies only when the ALEMODE variable is set to START.

** Applies only when the ALEMODE variable is set to END.


ALE Signal START/END Timing
T4
T1
T2
T3
T4
CLKOUT
ALE (Commercial chip specification)
ALE (Probe timing ALEMODE = START)
ALE (Probe timing ALEMODE = END)
NOTE
① T(ALE Active Delay)
② T(ALE Width)
③ T(ALE Inactive Delay)
④ CLKOUT low to ALE high for ALEMODE = END
⑤ Variable Width ALE remains high during T, T and ICE accesses

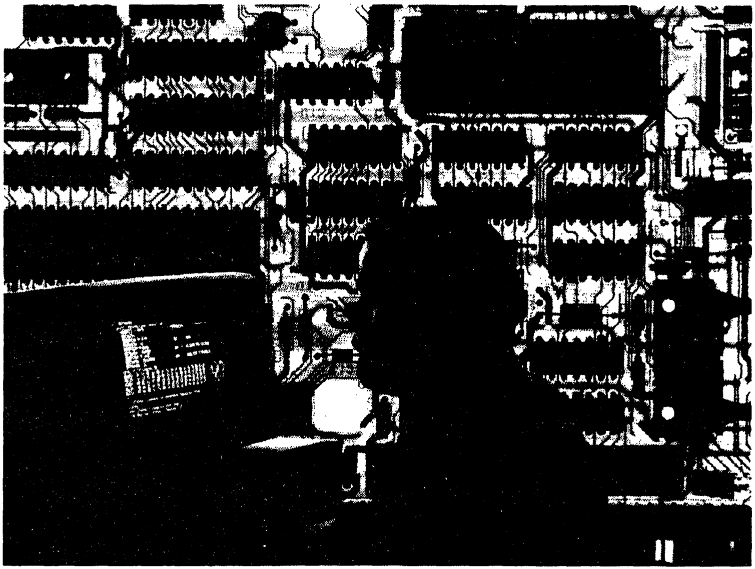
ENVIRONMENTAL SPECIFICATIONS

Operating Temperature 10°C to 40°C Ambient
Storage Temperature −40°C to 70°C

ORDERING INFORMATION

ICE186	ICE-186 System including ICE software (Requires DOS 3.XX PC AT with Above Board)
ICE 186AB	ICE 186 with Above Board included
ICE186IPAT	ICE-186 System including ICE S/W packages and the iPAT system (Requires DOS 3.XX PC AT with Above Board)
D86ASM86NL	86 macro assembler 86 builder/binder/ mapper utilities for DOS 3.XX.
D86C86NL	86 C compiler and run time libraries for DOS 3.XX.
D86PAS86NL	86 Pascal Compiler for DOS 3.XX.
D86PLM86NL	86 PL/M compiler for DOS 3.XX.
D86FOR86NL	86 Fortran compiler for DOS 3.XX.
ICEPAT KIT	iPAT Kit (Performance Analysis Tool) for ICE 186
ICEXONCE	Adapter for on-circuit emulation
ICEXLCC	Adapter for LCC component
ICEXPGA	Adapter for PGA component

ICE™-188 IN-CIRCUIT EMULATOR



HIGH PERFORMANCE REAL-TIME EMULATION

Intel's ICE-188 emulator delivers real-time emulation for the 80C188 microprocessor at speeds up to 12.5 MHz. The in-circuit emulator is a versatile and efficient tool for developing, debugging and testing products designed with the Intel 80C188 microprocessor. The ICE-188 emulator provides real time, full speed emulation in a user's system. Popular features such as symbolic debug, 2K bytes trace memory, and single-step program execution are standard on the ICE-188 emulator. Intel provides a complete development environment using assembler (ASM86) as well as high-level languages such as Intel's iC86, PL/M86, Pascal 86 and Fortran 86 to accelerate development schedules.

The ICE-188 emulator supports a subset of the 80C188 features at 12.5 MHz and at the TTL level characteristics of the component. The emulator is hosted on IBM's Personal Computer AT, already available as a standard development solution in most of today's engineering environments. The ICE-188 emulator operates in prototype or standalone mode, allowing software development and debug before a prototype system is available. The ICE-188 emulator is ideally suited for developing real-time applications such as industrial automation, computer peripherals, communications, office automation, or other applications requiring the full power of the 12.5 MHz 80C188 microprocessor.

ICE™-188 FEATURES

- Full 12.5 MHz Emulation Speed
- 2K Frames Deep Trace Memory
- Two-Level Breakpoints with Occurrence Counters
- Single-Step Capability
- 128K Bytes Zero Wait-State Mapped Memory
- Supports DRAM Refresh
- High-Level Language Support
- Symbolic Debug
- RS-232-C and GPIB Communication Links
- Crystal Power Accessory
- Interface for Intel Performance Analysis Tool (iPAT)
- Interface for Optional General Purpose Logic Analyzer
- Tutorial Software
- Complete Intel Service and Support.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

September, 1988

Order Number: 280816-001

HIGHEST EMULATION SPEED AVAILABLE TODAY

The ICE-188 emulator supports development and debug of time-critical hardware and software using Intel's 12.5 MHz 80C188 microprocessor.

RETRACE SOFTWARE TRACKS

This emulator captures up to 2,048 frames of processor activity, including both execution and data bus activity. With this trace memory, large blocks of program code can be traced in real time and viewed for program flow and behavior characteristics.

HARDWARE BREAKPOINTS FOR COMPLEX DEBUG

User-defined "TIL-THEN" breakpoint statements stop emulation at specific execution addresses or bus events. During the hardware and software integration phase, breakpoint statements can be defined as execution addresses and/or bus addresses and/or bus access types such as memory and I/O reads or writes. Additionally, event counters provide another level of breakpoint control for sophisticated state machine constructs used to specify emulation breakpoints/tracepoints.

SMALL OR LARGE STEPS

A stepping command can be used to view program execution one instruction at a time or in preset instruction blocks. When used in conjunction with symbolic debug, code execution can be monitored quickly and precisely.

DEBUG CODE WITHOUT A PROTOTYPE

Even before prototype hardware is available, the ICE-188 emulator working in conjunction with the Crystal Power Accessory (CPA) creates a "virtual" application environment. 128K bytes of zero wait-state memory is available for mapped memory and I/O resource addressing in 4K increments. The CPA provides emulator diagnostics as well as the ability to use the emulator without a prototype.

DON'T LOSE MEMORY

The ICE-188 emulator continues DRAM refresh signals even when emulation has been halted, thus ensuring DRAM memory will not be lost. During interrogation mode the ICE-188 emulator will keep the timers functioning and correctly respond to interrupts in real-time.

HIGH LEVEL LANGUAGE SUPPORT OPTIMIZED FOR INTEL TOOLS

The ICE-188 supports emulation for programs written in Intel's ASM86 or any of Intel's high-level languages:

PL/M-86	Fortran-86
Pascal-86	C-86

These languages are optimized for the Intel 80186/80188 component architectures to deliver a tightly integrated, high performance development environment.

USER-FRIENDLY SYMBOLICS AID IN DEBUG

Symbolics allow access to program symbols by name rather than cumbersome physical addresses. Symbolic debug speeds the debugging process by reducing reliance on memory maps. In a dynamic development process, user variables can be used as parameters for ICE-188 commands resulting in a consistent debug environment.

SUPPORTS FAST BREAKS

"Fastbreaks" is a feature which allows the emulation processor to halt, access memory, and return to emulation as quickly as possible. A fastbreak never takes more than 5625 clock cycles (most types of fastbreaks are considerably less). This feature is particularly useful in embedded applications.

MULTIPLE HIGH-SPEED COMMUNICATION LINKS

Two communication links are available for use in conjunction with the host IBM PC AT. The ICE-188 emulator uses either serial (RS-232-C) or a parallel (GPIB) link. A user supplied National Instruments (IEEE-488) GPIB communication board provides parallel transfers at rates up to 300K bytes per second.

SOFTWARE ANALYSIS (IPAT)

Intel's Performance Analysis Tool (iPAT) is designed to increase team productivity with features like interrupt latency measurement, code coverage analysis and software module performance analysis. These features enable the user to design reliable, high performance embedded control products. The ICE-188 emulator has an external 60 pin connector for iPAT.

BUILT-IN SUPPORT FOR LOGIC ANALYSIS

General-purpose logic analyzers can be used in conjunction with the ICE-188 to provide detailed timing of specific events. The ICE-188 emulator provides an external sync signal for triggering logic analysis, making complex trigger sequence programming easy. An additional 60 pin connector is included for the logic analyzer.

WORLDWIDE SERVICE AND SUPPORT

The ICE-188 emulator is supported by Intel's worldwide service and support organization. Total hardware and software support is available including a hotline number when the need is there.

Note: This emulator does not support use of the 8087.

SPECIFICATIONS

PERSONAL COMPUTER REQUIREMENTS

The ICE-188 emulator is hosted on an IBM PC AT. The emulator has been tested and evaluated on an IBM PC AT. The PC AT must meet the following minimum requirements:

- 640K Bytes of Memory
- Intel Above Board with at Least 1M Byte of Expansion Memory
- One 360K Bytes or One 1.2M Bytes floppy Disk Drive
- One 20M Bytes Fixed-Disk Drive
- PC DOS 3.2 or Later
- A serial Port (COM1 or COM2) Supporting Minimally at 9600 Baud Data Transfers, or a National Instruments GPIB-PC2A board.
- IBM PC AT BIOS

PHYSICAL DESCRIPTION AND CHARACTERISTICS

The ICE-188 Emulator consists of the following components:

Unit	Width		Height		Length	
	Inches	Cm.	Inches	Cm.	Inches	Cm.
Emulator Control Unit	10.40	26.40	1.70	4.30	20.70	52.60
Power Supply	7.60	19.00	1.15	10.70	11.00	27.90
User Probe	3.70	9.40	.65	1.60	7.00	17.80
User Cable/ Plcc					22.00	55.90
Hinge Cable					3.40	8.60
Crystal Power Accessory	4.30	10.90	.60	1.50	6.70	17.00
CPA Power Cable					9.00	22.90

ELECTRICAL CONSIDERATIONS

Icc 1050mA

I_{IH} 70μA Max.

I_{IL} – 1.5mA Max

I_{OH} – 1.0mA Max.

TIMING CONSIDERATIONS

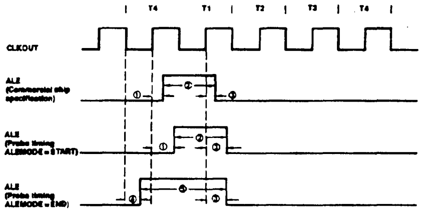
ICE-188 User AC Differences

Symbol	Parameter	COMPONENT SPEC		ICE-188 SPEC	
		Min.	Max.	Min.	Max.
T_{DVCL}	Data in Setup (A/D)	15		24	
T_{ARYCH}	Async Ready (ARDY) Resolution Transition Setup Time	15		23	
T_{SRYCL}	Synchronous Ready (SRDY) Transition Setup Time	15		25	
T_{HVCL}	HOLD Setup	15		32	
T_{INVCH}	NMI	15		32	
	/TEST	15		31	
	INTR,TIMERIN Setup Time	15		17	
T_{INVCL}	DRQ0, DRQ1, Setup Time	15		19	
T_{CLAZ}	Address Float Delay				
	READ Cycles	T_{CLAX}	25	5	36
	INTA cycles	T_{CLAX}	25	0	25
	HLDA	T_{CLAX}	25	10	50
T_{LHLL}	ALE Width (min)	T_{CLCL}-30		T_{CLCL}-32	
T_{CHLH}	ALE Active Delay*		25		42
(N/A)	CLKOUT Low to ALE Active**		(N/A)		19
T_{CHLL}	ALE Inactive Delay		25		40
T_{LLAX}	Address Hold to ALE Inactive (min)	T_{CHCL}-15		T_{CHCL}-28	
T_{CVCTX}	Control Inactive Delay	5	37	1	40
T_{AZRL}	Address Float to /RD Active	0		– 30	
T_{AVLL}	Address VAlid to ALE Low (min)	T_{CLCH}-15		T_{CLCH}-19	
T_{CHQSV}	Que Status Delay		28		35
T_{DXDL}	/DEN Inactive to DT /R Low	0		– 7	
T_{CICO}	CLKIN to CLKOUT Skew		21		37
Consult User Guide for Additional Specifications.					

*Applies only when the ALEMODE variable is set to START.

**Applies only when the ALEMODE variable is set to END.

ALE Signal START/END Timing


T4
T1
T2
T3
T4
CLKOUT
ALE (Commercial chip specification)
ALE (Probe timing ALEMODE = START)
ALE (Probe timing ALEMODE = END)
①
②
③
④
⑤

NOTE
① T_{CHLH} (ALE Active Delay)
② T_{LHLL} (ALE Width)
③ T_{CHLL} (ALE Inactive Delay)
④ CLKOUT low to ALE high for ALEMODE = END
⑤ Variable Width ALE remains high during T_{IDLE}, T_{HLDA} and ICE accesses

ORDERING INFORMATION

ICE188	ICE-188 System including ICE software (Requires DOS 3.XX PC AT with Above Board)
ICE 188AB	ICE 188 with Above Board included
D86ASM86NL	86 macro assembler 86 builder/binder/ mapper utilities for DOS 3.XX.
D86C86NL	86 C compiler and run time libraries for DOS 3.XX.
D86PAS86NL	86 Pascal Compiler for DOS 3.XX.
D86PLM86NL	86 PL/M compiler for DOS 3.XX.
D86FOR86NL	86 Fortran compiler for DOS 3.XX.
ICEPAT KIT	iPAT Kit (Performance Analysis Tool) for ICE 188
ICEXONCE	Adapter for on-circuit emulation
ICEXLCC	Adapter for LCC component
ICEXPGA	Adapter for PGA component
UP188	User probe to convert ICE-186 to support 80C188 component

ENVIRONMENTAL SPECIFICATIONS

Operating Temperature 10°C to 40°C Ambient
Storage Temperature −40°C to 70°C

I²ICE™ IN-CIRCUIT EMULATION SYSTEM

IN-CIRCUIT EMULATOR FOR THE 8086/80186/80286 FAMILY OF MICROPROCESSORS

The I²ICE™ In-Circuit Emulator is a high-performance, cost-effective debug environment for developing systems with the Intel 8086/80186/80286 family of microprocessors. With 10 MHz emulation, a window-oriented user interface, and compatibility with Intel's iPAT™ Performance Analysis Tool, the I²ICE Emulator gives you unmatched speed and control over all phases of hardware/software debug.

FEATURES

- Emulation speeds up to 10 MHz with 8086/88, 80186/188 and 80286 microprocessors
- 8087 and 80287 numeric coprocessor support
- Hosted on IBM PC AT*, AT BIOS, or compatibles
- ICEVIEW™ window-oriented user interface with pull-down menus and context-sensitive help
- Source and symbol display using all Intel languages
- 1K frame bus and execution trace buffer
- Symbolic debugging for flexible access to memory location and program variables
- Flexible breakpointing for quick problem isolation
- Memory expandable to 288K with zero wait states
- Worldwide service and support
- iPAT option for software speed tuning

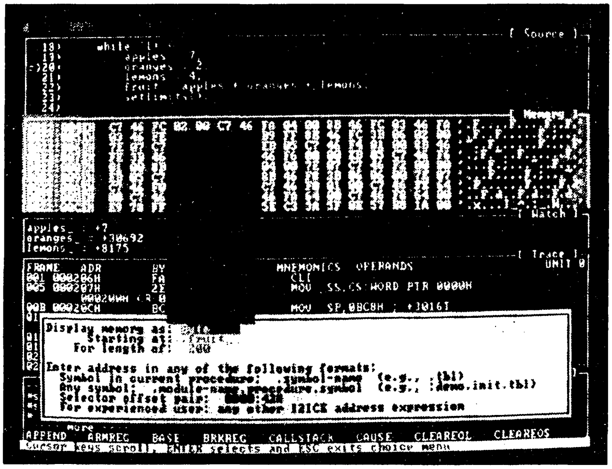
I²ICE, ICEVIEW, and iPAT are trademarks of Intel Corporation.

*IBM is a trademark of International Business Machines Corp.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

July, 1988

Order Number: 280800-001



[Source]
[Memory]
[Watch]
[Trace]
UNIT 0
more
APPEND ARMREG BASE BRKREG CALLSTACK CAUSE CLEAREOL CLEAREOS
Cursor keys scroll, ENTER selects and ESC exits choice menu

```
18>      while (1) {
19>          apples = 7;
=>20>          oranges = 2;
21>          lemons = 4;
22>          fruit = apples + oranges + lemons;
23>          setlimits();
24>
```

```
apples_ : +7
oranges_ : +30692
lemons_ : +8175
```

```
FRAME  ADR          BY              MNEMONICS   OPERANDS
001 000206H         FA              CLI
005 000207H         2E              MOV  SS,CS:WORD PTR 0000H
           00020AH  (R 0
00B 00020CH         BC              MOV  SP,0BC8H  ; +3016T
```

```
Display memory as:  Byte
      Starting at:  fruit
    For length of:  200

Enter address in any of the following formats:
  Symbol in current procedure:  .symbol-name  (e.g., :tbl)
  Any symbol:  :module-name.procedure.symbol  (e.g., :demo.init.tbl)
  Selector offset pair:  0000:438
  For experienced user: any other I2ICE address expression
```

Plate 1. An example of the ICEVIEW™ user interface showing source, memory, watch, and trace.

ONE TOOL FOR THE ENTIRE DEVELOPMENT PROCESS

The I²ICE Emulator allows hardware and software design to proceed simultaneously, so you can develop software even before prototype hardware is available. With 32K of zero wait-state mappable memory (and an additional 256K with optional memory boards), you can use the I²ICE Emulator to debug at any stage of the development cycle: hardware development, software development, system integration or system test.

HIGH-SPEED, REAL-TIME EMULATION

The I²ICE Emulator delivers full-speed, real-time emulation at speeds up to 10 MHz. Based on Intel's exclusive microprocessor technology, the I²ICE Emulator matches each chip's electrical and timing characteristics without memory or interrupt intrusions, ensuring design accuracy and eliminating surprises. The performance of your prototype is the performance you can expect from your final product.

EASY-TO-USE ICEVIEW™ INTERFACE

The ICEVIEW interface makes the I²ICE Emulator easy to learn and use by providing easy access to application information and ICE functions. Pull-down menus and windows boost productivity for both new and experienced users. Multiple on-screen windows allow you to access the source display, execution trace, register, and other important information, all at the same time. You can watch the information change as you modify and step through your program. You can even customize window size and screen positions.

A command line interface is also available with syntax checking and context-sensitive prompts. ICEVIEW works with monochrome, CGA and the latest EGA color displays.

SYMBOLIC DEBUG SPEEDS DEVELOPMENT

The extensive debug symbolics generated by the Intel 8086 and 80286 assemblers and compilers can increase your development productivity. Symbolics with automatic formatting are available for all primitive types, regardless of whether the variables are globals, locals (stack-resident) or pointers. The virtual symbol table supports all symbolics, even in very large programs. Aliasing can be used to reduce keystrokes and save time.

POWERFUL BREAK AND TRACE CAPABILITY FOR FAST PROBLEM ISOLATION

The I²ICE Emulator allows up to eight simultaneous break/trace conditions to be set (four execution, four bus), a timesaver when solving hardware/software integration problems. Break and trace points can be set on specified line numbers, on procedures, or on symbolic data events, such as writing a variable to a value or range of values. You can break or trace on specific hardware events, such as a read or write to a specific address, data or I/O port, or on a combination of events.

MULTIPROCESSOR, PROTECTED MODE, AND COPROCESSOR SUPPORT

Up to four I²ICE systems can be linked and controlled simultaneously from one PC host, enabling you to debug multiprocessor systems. The I²ICE Emulator with an 80286 probe supports all 80286 protected mode capabilities. It also supports the 8087 and 80287 numeric coprocessors.

iPAT™ FOR SOFTWARE PERFORMANCE AND CODE COVERAGE ANALYSIS

The I^{2}ICE Emulator interfaces to Intel's iPAT Performance Analysis Tool for examining software execution speeds and code coverage in real time. iPAT displays critical performance data about your code in easy-to-understand histograms and tables. Elusive bottlenecks are readily seen, allowing you to focus your attention to get the most performance out of your product.

iPAT also performs code execution coverage, letting you perform product evaluations faster and more effectively. iPAT pinpoints areas in your code either executed or not executed according to specific conditions, taking the guesswork out of software evaluations.

EASY INTERFACE TO EXTERNAL INSTRUMENTS

The I^{2}ICE system includes external emulation clips and software support for setting breakpoints, tracepoints and arm/disarm conditions on external events, making it easy to connect external logic analyzers and signal generators. You can debug complex hardware/software interactions with a high level of productivity.

WORLDWIDE SERVICE AND SUPPORT

The I^{2}ICE Emulator is supported by Intel's worldwide service and support organization. In addition to an extended warranty, you can choose from hotline support, on-site systems engineering assistance, and a variety of hands- on training workshops.

SPECIFICATIONS

HOST REQUIREMENTS

IBM PC/AT or 100% PC AT BIOS compatible
DOS 3.1 or later
640K bytes of memory
360K bytes or 1.2 MB floppy disk drive
Hard disk drive
Monochrome, CGA or EGA monitor (EGA recommended)

PHYSICAL DESCRIPTION

	Width		Height		Length	
Unit	cm	in	cm	in	cm	in
I^{2}ICE chassis	43.2	17.0	21.0	8.25	61.3	24.13
Probe base	21.6	8.5	7.6	3.0	25.4	10.0

Host/chassis cable 15 ft. (4.6 m)

ELECTRICAL CHARACTERISTICS

90-132 V or 180-264 V (selectable)
47-63 Hz
12 amps (AC)

ENVIRONMENTAL SPECIFICATIONS

Operating temperature: 0-40°C (32-104°F) ambient
Operating humidity: Maximum of 85% relative humidity, non-condensing

ORDERING INFORMATION

Kit Code	Contents
pIII010KITD	I^{2}ICE system 10 MHz 8086/8088 support kit for IBM PC host. Includes probe, chassis, and host interface module and software.
pIII111KITD	I^{2}ICE system 10 MHz 80186 support kit for IBM PC host. Includes probe, chassis, host interface module and software. Note: For 80188 support, the III198 option below must also be ordered.
III198	10 MHz 80188 support conversion kit to convert 80186 probe to 80188 probe.
pIII212KITD	I^{2}ICE system, 10 MHz 80286 support kit for IBM PC AT host. Includes probe, chassis, host interface module and software.
III010PATC86D	I^{2}ICE system 10 MHz 8086/8088 support kit with iPAT Performance Analysis Tool for PC AT host. Includes I^{2}ICE probe, chassis, host interface module, iPAT tool option, cables and software. Also includes iC-86 compiler, 86 Macro Assembler, utilities, and AEDIT text editor.
III111PATC86D	As above for 10 MHz 80186 support.
III212PATC86D	As above for 10 MHz 80286 support. Note: C-286 and RLL-286 and ASM-286 must be ordered separately.
954D	I^{2}ICE PC AT host software. Includes ICEVIEW™ windowed human interface.

Note: I^{2}ICE probes, chassis, software, cables and iPAT options are available separately.

ICE-286 IN-CIRCUIT EMULATOR

HIGH PERFORMANCE REAL-TIME EMULATION

Intel's ICE-286 emulator delivers real-time emulation for the 80286 microprocessor at speeds up to 12.5 MHz. The in-circuit emulator is a versatile and efficient tool for developing, debugging and testing products designed with the Intel 80286 microprocessor. The ICE-286 emulator provides real time, full speed emulation in a users system. Popular features such as symbolic debug, 2K bytes trace memory, and single-step program execution are standard on the ICE-286 emulator. Intel provides a complete development environment using assembler (ASM-286) as well as high-level languages such as Intel's iC286, PL/M-286 or Fortran 286 to accelerate development schedules.

Intel's ICE-286 emulator is hosted on IBM's Personal Computer AT, already available as a standard development solution in most of today's engineering environments. The ICE-286 emulator operates in prototype or standalone mode allowing software development and debug before a prototype system is available. The ICE-286 emulator is ideally suited for developing real time applications such as process control, machine control, communications, or other applications requiring the full power of the 12.5 MHz 80286 microprocessor.

ICE-286 FEATURES

- Full 12.5 MHz Emulation Speed
- 2K Bytes Deep Trace Memory
- Two-Level Breakpoints with Occurrence Counters
- Single-Step Capability
- 128K Bytes Zero Wait-State Mapped Memory
- Support For Protected and Real Modes
- High-Level Language Support
- Symbolic Debug
- Numeric Processor Extension Support
- RS-232-C and GPIB Communication Links
- Crystal Power Accessory
- Interface for Intel Performance Analysis Tool (iPAT)
- Interface for Optional General Purpose Logic Analyzer
- Tutorial Software
- Complete Intel Service and Support

intel

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

March, 1988
Order Number: 280728-002

HIGHEST EMULATION SPEED AVAILABLE TODAY

The ICE-286 emulator supports development and debug of time-critical hardware and software using Intel's 12.5 MHz 80286 microprocessor.

RETRACE SOFTWARE TRACKS

This emulator captures up to 2048 frames of processor activity, including both execution and data bus activity. With this trace memory, large blocks of program code can be traced in real time and viewed for program flow and behavior characteristics.

HARDWARE BREAKPOINTS FOR COMPLEX DEBUG

User-defined "TIL-THEN" breakpoint statements stop emulation at specific execution addresses or bus events. During the hardware and software integration phase, breakpoint statements can be defined as execution addresses and/or bus addresses and/or bus access types, such as memory and I/O reads or writes. Additionally, event counters provide another level of breakpoint control for sophisticated state machine constructs used to specify emulation breakpoints/tracepoints.

SMALL OR LARGE STEPS

A stepping command can be used to view program execution one frame at a time or in preset frame blocks. When used in conjunction with symbolic debug, code execution can be monitored quickly and precisely.

DEBUG CODE WITHOUT A PROTOTYPE

Even before prototype hardware is available, the ICE-286 emulator working in conjunction with the Crystal Power Accessory (CPA) creates a "virtual" application environment. 128K bytes of zero wait-state memory is available for mapped memory and I/O resource addressing in 4K increments. The CPA provides emulator diagnostics as well as the ability to use the emulator without a prototype.

PROTECTED AND REAL MODES

The ICE-286 emulator has full access to all protected-mode registers and permits modification of register contents. Protected mode of execution if beneficial for secure, multitasking applications.

HIGH-LEVEL LANGUAGE SUPPORT OPTIMIZED FOR INTEL TOOLS

The ICE-286 supports emulation for programs written in Intel's ASM 286 and ASM 86 or any of the Intel high-level languages:

PL/M-286/86	Fortran-286/86
Pascal-286/86	C-286/86

These languages are optimized for Intel component architectures to deliver a tightly integrated, high performance development environment.

USER-FRIENDLY SYMBOLICS AID IN DEBUG

Symbolics allow access to program symbols by name rather than cumbersome physical addresses. Symbolic debug speeds the debugging process by reducing reliance on memory maps. In a dynamic development process, user variables can be used as parameters for ICE-286 commands resulting in a consistent debug environment.

80287 NUMERICS SUPPORT

The ICE-286 emulator provides emulation support for the 80287 numerics processor. 80287 registers can be displayed and modified allowing full debug support for numerics.

MULTIPLE HIGH-SPEED COMMUNICATION LINKS

Two communication links are available for use in conjunction with the host IBM PC AT. The ICE-286 emulator uses either serial (RS-232-C) or a parallel (GPIB) link. A user supplied National Instruments (IEEE-488) GPIB communication board provides parallel transfers at rates up to 300K bytes per second.

SOFTWARE ANALYSIS (iPAT)

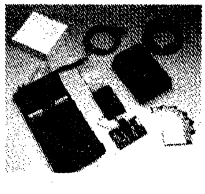
Intel's Performance Analysis Tool (iPAT) is designed to increase team productivity with features like interrupt latency measurement, code coverage analysis and software module performance analysis. These features enable the user to design reliable, high performance embedded control products. The ICE-286 emulator has an external 60 pin connector for iPAT.

BUILT-IN SUPPORT FOR LOGIC ANALYSIS

General-purpose logic analyzers can be used in conjunction with the ICE-286 to provide detailed timing of specific events. The ICE-286 emulator provides an external sync signal for triggering logic analysis, making complex trigger sequence programming easy. An additional 60 pin connector is included for the logic analyzer.

WORLDWIDE SERVICE AND SUPPORT

The ICE-286 emulator is supported by Intel's worldwide service and support organization. Total hardware and software support is available including a hotline number when the need is there.



SPECIFICATIONS

PERSONAL COMPUTER REQUIREMENTS

The ICE-286 emulator is hosted on an IBM PC AT. The emulator has been tested and evaluated on an IBM PC AT. The PC AT must meet the following minimum requirements:

- 640K Bytes of Memory
- Intel Above Board with at Least 1M Byte of Expansion Memory
- One 360K Bytes or One 1.2M Bytes Floppy Disk Drive
- One 20M Bytes Fixed-Disk Drive
- PC-DOS 3.2 or Later
- A Serial Port (COM1 or COM2) Supporting Minimally at 9600 Baud Data Transfers, or a National Instruments GPIB-PC2A Board.
- IBM PC AT BIOS

ELECTRICAL CONSIDERATIONS

Icc 1050mA

ENVIRONMENTAL SPECIFICATIONS

Temperature 10°C to 40°C Ambient
Storage Temperature −40°C to 70°C

TIMING/DC CONSIDERATIONS

ICE™-286 USER PIN DIFFERENCES					
		COMPONENT SPEC.		ICE-286 SPEC.	
	PARAMETER	Min	Max	Min	Max
2	System (CLK) low time	11	237	14	236
3	System (CLK) high time	13	239	14	236
8	Read Data Setup	5		7	
10	/Ready Setup	22		24	
12a	Status/peak * active delay	3	18	3	20
12b	Status/peak * inactive delay	3	20	3	22
13	Address valid delay	1	32	1	34
14	Write data valid delay	0	30	0	33
15	Address/status/data float	0	32	0	34
Consult User Guide for additional specifications.					

PHYSICAL DESCRIPTION AND CHARACTERISTICS

The ICE-286 Emulator consists of the following components:

	Width		Height		Length	
Unit	Inches	Cm.	Inches	Cm.	Inches	Cm.
Emulator Control Unit	10.40	26.40	1.70	4.30	20.70	52.60
Power Supply	7.60	19.00	4.15	10.70	11.00	27.90
User Probe	3.70	9.40	.65	1.60	7.00	17.80
User Cable/ Plcc					22.00	55.90
Hinge Cable					3.40	8.60
Crystal Power Accessory	4.30	10.90	.60	1.50	6.70	17.00
CPA Power Cable					9.00	22.90

ORDERING INFORMATION

ICE286	ICE-286 NMOS System including ICE S/W packages (Requires DOS 3.XX PC AT with Above Board)
ICE286AB	ICE-286 NMOS System including ICE S/W packages and Intel's 2M Byte Above Board (PCMB 4125) (Requires DOS 3.XX PC-AT)
ICE286PAT	ICE-286 NMOS System including ICE S/W Packages and the iPAT system (Requires DOS 3.XX PC AT with Above Board)
D86ASM286NL	286 macro assembler 286 builder/binder/mapper utilities for DOS 3.XX.
D86C286NL	286 C compiler and run time libraries for DOS 3.XX.
D86PLM286NL	286 PL/M compiler for DOS 3.XX.

INTEL386™ FAMILY DEVELOPMENT SUPPORT

COMPREHENSIVE DEVELOPMENT SUPPORT FOR THE INTEL386™ FAMILY OF MICROPROCESSORS

The perfect complement to the Intel386™ Family of microprocessors is the optimum development solution. From a single source, Intel, comes a complete, syncrgistic hardware and software development toolset, delivering full access to the power of the Intel386 architecture in a way that only Intel can.

Intel development tools are easy to use, yet powerful, with contemporary user interface techniques and productivity boosting features such as symbolic debugging. And you'll find Intel first to market with the tools needed to start development, and with lasting product quality and comprehensive support to keep development on-track.

If what interests you is getting the best product to market in as little time as possible, Intel is the choice.

FEATURES

- Comprehensive support for the full 32 bit Intel386 architecture, including protected mode and 4 gigabyte physical memory addressing
- Source line display and symbolics allow debugging in the context of the original program
- Architectural extensions in Intel high-level languages provides for manipulating hardware directly without assembly language routines
- A common object code format (OMF-386) supports the intermixing of modules written in various languages
- ROM-able code is output directly from the language tools, significantly reducing the effort necessary to integrate software into the final target system
- Support for the 80387 numeric coprocessor

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel and is subject to change without notice.

September, 1988
Order Number: 280808-001

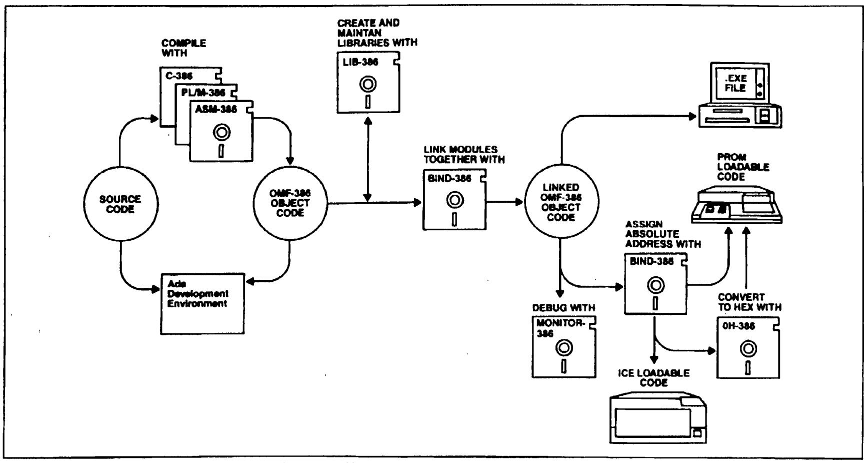


Figure 1. Intel Microprocessor Development Environment

ASM-386 MACRO ASSEMBLER

ASM-386 is a "high-level" macro assembler for the Intel386 Family. ASM-386 offers many features normally found only in high-level languages. The macro facility in ASM-386 saves development time by allowing common program sequences to be coded only once. The assembly language is strongly typed, performing extensive checks on the usage of variables and labels.

Other ASM-386 features include:

- "High-level" assembler mnemonics to simplify the language
- Structures and records for data representation
- Upward compatibility with ASM-286

PL/M-386 COMPILER

PL/M-386 is a structured high-level system implementation language for the Intel386 Family. PL/M-386 supports the implementation of protected operating system software by providing built-in procedures and variables to access the Intel386 architecture.

For efficient code generation, PL/M-386 features four levels of optimization, a virtual symbol table, and four models of program size and memory usage.

Other PL/M-386 features include:

- The ability to define a procedure as an interrupt handler as well as facilities for generating interrupts
- Direct support of byte, half-word, and word input and output from microprocessor ports
- Upward compatibility with PL/M-286 and PL/M-86 source code

PL/M-386 combines the benefits of a high-level language with the ability to access the Intel386 architecture. For the development of systems software, PL/M-386 is a cost-effective alternative to assembly language programming.

C-386 COMPILER

C-386 brings the C language to the Intel386 Family. For code efficiency, C-386 features two levels of optimization, three models of program size and memory usage, and an extremely efficient register allocator. The C-386 compiler eliminates common code, eliminates redundant loads and stores, and resolves span dependencies (shortens branches) within a program.

C-386 allows full access to the Intel386 architecture through control of bit fields, pointers, addresses, and register allocations.

Other C-386 features include:

- An interrupt directive defining a function as an interrupt function
- Built-in functions allow direct access to the microprocessor through the inline insertion of machine code
- Structure assignments, functions taking structure arguments, and returning structures, and the void and enum data types

The C-386 runtime library is implemented in layers. The upper layers include the standard I/O library (STDIO), memory management routines, conversion routines, and string manipulation routines. The lowest layer, operating system interface routines, is documented for adaptation to the target environment.

RLL-386™ RELOCATION, LINKAGE, AND LIBRARY TOOLS

The RLL-386™ relocation, linkage, and library tools are a cohesive set of utilities featuring comprehensive support of the full Intel386 architecture. RLL-386 provides for a variety of functions—from linking separate modules, building an object library, or linking in 80387 support, to building a task to execute under protected mode or the multi-tasking, memory protected system software itself.

The RLL-386 relocation, linkage, and library tools package includes a program binder for linking ASM, PL/M, and C modules together, a system builder for configuring protected, multi-task systems, a cross reference mapper, a program librarian, an 80387 numeric coprocessor support library, and a conversion utility for outputing hex format code for PROM programming.

Ada-386™ CROSS COMPILATION PACKAGE

The Ada-386™ Cross-Compilation Package is a complete development environment for embedded real-time Ada applications for the 386 microprocessor. The Ada cross compiler, which runs under VAX/VMS, generates code highly optimized for the 80386. The Ada-386 Cross-Compilation Package also features a VMS hosted and targeted compiler and tools to support software debugging before the target system is available. Sophisticated code generation tools, such as the Global Optimizer, help make the target code smaller and more efficient.

Ada-386 includes a source level symbolic debugger working in unison with a small debug monitor supplied in PROM. Code can also be downloaded and debugged using Intel's ICE™-386 In-Circuit Emulator.

Other Ada-386 features include:

- The ability to directly call Intel's iRMK real-time kernel
- An object module importer allows program modules written in other Intel386 Family languages to be linked with Ada modules
- Built in support for the 386, including machine code insertion and full representation specifications
- Highly optimized interrupt handling—fast execution of interrupt handlers without requiring a context switch

VALIDATED Ada

INTEL386™ FAMILY IN-CIRCUIT EMULATORS

Intel386 Family in-circuit emulators embody exclusive technology that gives the emulator access to internal processor states that are accessible in no other way. Intel386 microprocessors fetch and execute instructions in parallel; fetched instructions are not necessarily executed. Because of this, an emulator without this access to internal processor states is prone to error in determining what actually occurred inside the microprocessor. With Intel's exclusive technology, Intel386 Family emulators are one hundred percent accurate.

Other features of Intel386 Family in-circuit emulators include:

- Unparalleled support of the Intel386 architecture, notably the native protected mode
- Emulation at clock speeds to 25MHz and full-featured trigger and trace capabilities
- Non-intrusive operation
- Convertible to support any Intel386 microprocessor

With symbolic debugging, memory locations can be examined or modified using symbolic references to the original program, such as a procedure or a variable name, line number, or program label. Source code associated with a given line number can be displayed, as can the type information of variables, such as byte, word, record, or array. Microprocessor data structures, such as registers, descriptor tables, and page tables, can also be examined and modified using symbolic names. The symbolic debugging information for use with Intel development tools is produced only by Intel languages.

MONITOR-386™ SOFTWARE DEBUGGER

MONITOR-386™ is a software debugger for 386 and 386SX-based systems. MONITOR-386 provides program execution control and symbolic processor and memory interrogation and modification. Hardware and software breakpoints can be set at symbolic addresses and program execution can be single-stepped through assembly or high-level language instructions.

Other MONITOR-386 features include:

- Debug procedures, named user-definable sequences of MONITOR-386 commands, enable users to define macro commands that would otherwise take several lines of command entries to perform the same function
- A disassembler/single line assembler allows users to display memory as and patch memory with 80386/80387 mnemonics

MONITOR-386, used in conjunction with Intel single board computers iSBC® 386/22 and iSBC 386/116, can debug software before a functional prototype of the target system is available.

iPAT-386™ PERFORMANCE ANALYSIS TOOL

iPAT-386™ performance analysis tool provides analysis of real-time software executing on a 386-based target system. With iPAT-386, it is possible to speed-tune applications, optimize use of operating systems, determine response characteristics, and identify code execution coverage.

By examining iPAT-386 histogram and tabular information about procedure usage (with the option of including interaction with other procedures, hardware, the operating system, or interrupt service routines) for critical functions, performance bottlenecks can be identified. With iPAT-386 code execution coverage information, the completeness of testing can be confirmed. iPAT-386 can be used in conjunction with Intel's ICE-386™ in-circuit emulator to control test conditions.

iPAT-386 provides real-time analysis up to 20MHz, performance profiles of up to 125 partitions, and code execution coverage analysis over 252K.

SERVICE, SUPPORT, AND TRAINING

To augment its development tools, Intel offers a full array of seminars, classes, and workshops, field application engineering expertise, hotline technical support, and on-site service.

PRODUCT SUPPORT MATRIX

	Component			Host	
Product	**386**	**386sx**	**376**	**DOS**	**VAX/VMS**
ASM-386 Macro Assembler	✓	✓	✓	✓	✓
PL/M-386 Compiler	✓	✓	✓	✓	✓
C-386 Compiler	✓	✓	✓	✓	✓
RLL-386 Relocation, Linkage, and Library tools	✓	✓	✓	✓	✓
Ada-386 Cross Compilation Package	✓				✓
Intel386 Family In-Circuit Emulators	✓	✓	✓	✓	
Monitor-386 Software Debugger	✓	✓		✓	
iPAT-386 Performance Analysis Tool	✓			✓	

Intel386, 386, 386SX, 376, ICE, and iSBC are trademarks of Intel Corporation.
VAX and VMS are registered trademarks of Digital Equipment Corporation.

ORDERING INFORMATION

For direct information on Intel's Development Tools, or for the number of your nearest sales office or distributor, call 800-874-6835 (U.S.). For information or literature on additional Intel products, call 800-548-4725 (U.S. and Canada).

DOMESTIC SALES OFFICES

ALABAMA

†Intel Corp.
5015 Bradford Dr., #2
Huntsville 35805
Tel: (205) 830-4010

ARIZONA

†Intel Corp.
11225 N. 28th Dr.
Suite D-214
Phoenix 85029
Tel: (602) 869-4980

†Intel Corp.
1161 N. El Dorado Place
Suite 301
Tucson 85715
Tel: (602) 299-6815

CALIFORNIA

†Intel Corp.
21515 Vanowen Street
Suite 116
Canoga Park 91303
Tel: (818) 704-8500

†Intel Corp.
2250 E. Imperial Highway
Suite 218
El Segundo 90245
Tel: (213) 640-6040

†Intel Corp.
1510 Arden Way, Suite 101
Sacramento 95815
Tel: (916) 920-8096

†Intel Corp.
4350 Executive Drive
Suite 105
San Diego 92121
Tel: (619) 452-5880

†Intel Corp.*
400 N. Tustin Avenue
Suite 450
Santa Ana 92705
Tel: (714) 835-9642
TWX: 910-595-1114

†Intel Corp.*
San Tomas 4
2700 San Tomas Expressway
2nd Floor
Santa Clara 95051
Tel: (408) 986-8086
TWX: 910-338-0255
FAX: 408-727-2620

COLORADO

†Intel Corp.
4445 Northpark Drive
Suite 100
Colorado Springs 80907
Tel: (719) 594-6622

†Intel Corp.*
650 S. Cherry St., Suite 915
Denver 80222
Tel: (303) 321-8086
TWX: 910-931-2289

CONNECTICUT

†Intel Corp.
26 Mill Plain Road
2nd Floor
Danbury 06811
Tel: (203) 748-3130
TWX: 710-456-1199

FLORIDA

†Intel Corp.
6363 N.W. 6th Way, Suite 100
Ft. Lauderdale 33309
Tel: (305) 771-0600
TWX: 510-956-9407
FAX: 305-772-8193

†Intel Corp.
5850 T.G. Lee Blvd.
Suite 340
Orlando 32822
Tel: (407) 240-8000
FAX: 407-240-8097

†Intel Corp.
11300 4th Street North
Suite 170
St. Petersburg 33716
Tel: (813) 577-2413
FAX: 813-578-1607

GEORGIA

†Intel Corp.
3280 Pointe Parkway
Suite 200
Norcross 30092
Tel: (404) 449-0541

ILLINOIS

†Intel Corp.*
300 N. Martingale Road, Suite 400
Schaumburg 60173
Tel: (312) 605-8031
FAX: 312-605-9762

INDIANA

†Intel Corp.
8777 Purdue Road
Suite 125
Indianapolis 46268
Tel: (317) 875-0623

IOWA

Intel Corp.
1930 St. Andrews Drive N.E.
2nd Floor
Cedar Rapids 52402
Tel: (319) 393-5510

KANSAS

†Intel Corp.
10985 Cody St.
Suite 140, Bldg. D
Overland Park 66210
Tel: (913) 345-2727

MARYLAND

†Intel Corp.*
7321 Parkway Drive South
Suite C
Hanover 21076
Tel: (301) 796-7500
TWX: 710-862-1944

†Intel Corp.
7833 Walker Drive
Suite 550
Greenbelt 20770
Tel: (301) 441-1020

MASSACHUSETTS

†Intel Corp.*
Westford Corp. Center
3 Carlisle Road
2nd Floor
Westford 01886
Tel: (508) 692-3222
TWX: 710-343-6333

MICHIGAN

†Intel Corp.
7071 Orchard Lake Road
Suite 100
West Bloomfield 48322
Tel: (313) 851-8096

MINNESOTA

†Intel Corp.
3500 W. 80th St., Suite 360
Bloomington 55431
Tel: (612) 835-6722
TWX: 910-576-2867

MISSOURI

†Intel Corp.
4203 Earth City Expressway
Suite 131
Earth City 63045
Tel: (314) 291-1990

NEW JERSEY

†Intel Corp.*
Parkway 109 Office Center
328 Newman Springs Road
Red Bank 07701
Tel: (201) 747-2233

†Intel Corp.
280 Corporate Center
75 Livingston Avenue
First Floor
Roseland 07068
Tel: (201) 740-0111
FAX: 201-740-0626

NEW MEXICO

†Intel Corp.
8500 Menaul Boulevard N.E.
Suite B 295
Albuquerque 87112
Tel: (505) 292-8086

NEW YORK

Intel Corp.
127 Main Street
Binghamton 13905
Tel: (607) 773-0337
FAX: 607-723-2677

†Intel Corp.*
850 Cross Keys Office Park
Fairport 14450
Tel: (716) 425-2750
TWX: 510-253-7391

†Intel Corp.*
2950 Expressway Dr., South
Suite 130
Islandia 11722
Tel: (516) 231-3300
TWX: 510-227-6236

†Intel Corp.
Westage Business Center
Bldg. 300, Route 9
Fishkill 12524
Tel: (914) 897-3860
FAX: 914-897-3125

NORTH CAROLINA

†Intel Corp.
5800 Executive Center Dr.
Suite 105
Charlotte 28212
Tel: (704) 568-8966
FAX: 704-535-2236

†Intel Corp.
2700 Wycliff Road
Suite 102
Raleigh 27607
Tel: (919) 781-8022

OHIO

†Intel Corp.*
3401 Park Center Drive
Suite 220
Dayton 45414
Tel: (513) 890-5350
TWX: 810-450-2528

†Intel Corp.*
25700 Science Park Dr., Suite 100
Beachwood 44122
Tel: (216) 464-2736
TWX: 810-427-9298

OKLAHOMA

†Intel Corp.
6801 N. Broadway
Suite 115
Oklahoma City 73162
Tel: (405) 848-8086

OREGON

†Intel Corp.
15254 N.W. Greenbrier Parkway
Building B
Beaverton 97006
Tel: (503) 645-8051
TWX: 910-467-8741

PENNSYLVANIA

†Intel Corp.*
455 Pennsylvania Avenue
Suite 230
Fort Washington 19034
Tel: (215) 641-1000
TWX: 510-661-2077

Intel Corp.*
400 Penn Center Blvd., Suite 610
Pittsburgh 15235
Tel: (412) 823-4970

PUERTO RICO

†Intel Microprocessor Corp.
South Industrial Park
P.O. Box 910
Las Piedras 00671
Tel: (809) 733-8616

TEXAS

†Intel Corp.
313 E. Anderson Lane
Suite 314
Austin 78752
Tel: (512) 454-3628

†Intel Corp.*
12000 Ford Road
Suite 400
Dallas 75234
Tel: (214) 241-8087
FAX: 214-484-1180

†Intel Corp.*
7322 S.W. Freeway
Suite 1490
Houston 77074
Tel: (713) 988-8086
TWX: 910-881-2490

UTAH

†Intel Corp.
428 East 6400 South
Suite 104
Murray 84107
Tel: (801) 263-8051

VIRGINIA

†Intel Corp.
1504 Santa Rosa Road
Suite 108
Richmond 23288
Tel: (804) 282-5668

WASHINGTON

†Intel Corp.
155 108th Avenue N.E.
Suite 386
Bellevue 98004
Tel: (206) 453-8086
TWX: 910-443-3002

†Intel Corp.
408 N. Mullan Road
Suite 102
Spokane 99206
Tel: (509) 928-8086

WISCONSIN

†Intel Corp.
330 S. Executive Dr.
Suite 102
Brookfield 53005
Tel: (414) 784-8087
FAX: (414) 796-2115

CANADA

BRITISH COLUMBIA

Intel Semiconductor of Canada, Ltd.
4585 Canada Way, Suite 202
Burnaby V5G 4L6
Tel: (604) 298-0387
FAX: (604) 298-8234

ONTARIO

†Intel Semiconductor of Canada, Ltd.
2650 Queensview Drive
Suite 250
Ottawa K2B 8H6
Tel: (613) 829-9714
TLX: 053-4115

†Intel Semiconductor of Canada, Ltd.
190 Attwell Drive
Suite 500
Rexdale M9W 6H8
Tel: (416) 675-2105
TLX: 06983574
FAX: (416) 675-2438

QUEBEC

†Intel Semiconductor of Canada, Ltd.
620 St. John Boulevard
Pointe Claire H9R 3K2
Tel: (514) 694-9130
TWX: 514-694-9134

†Sales and Service Office
*Field Application Location

DOMESTIC DISTRIBUTORS

ALABAMA

Arrow Electronics, Inc.
1015 Henderson Road
Huntsville 35805
Tel: (205) 837-6955

†Hamilton/Avnet Electronics
4940 Research Drive
Huntsville 35805
Tel: (205) 837-7210
TWX: 810-726-2162

Pioneer/Technologies Group, Inc.
4825 University Square
Huntsville 35805
Tel: (205) 837-9300
TWX: 810-726-2197

ARIZONA

†Hamilton/Avnet Electronics
505 S. Madison Drive
Tempe 85281
Tel: (602) 231-5140
TWX: 910-950-0077

Hamilton/Avnet Electronics
30 South McKiemy
Chandler 85226
Tel: (602) 961-6669
TWX: 910-950-0077

Arrow Electronics, Inc.
4134 E. Wood Street
Phoenix 85040
Tel: (602) 437-0750
TWX: 910-951-1550

Wyle Distribution Group
17855 N. Black Canyon Hwy.
Phoenix 85023
Tel: (602) 249-2232
TWX: 910-951-4282

CALIFORNIA

Arrow Electronics, Inc.
10824 Hope Street
Cypress 90630
Tel: (714) 220-6300

Arrow Electronics, Inc.
19748 Dearborn Street
Chatsworth 91311
Tel: (213) 701-7500
TWX: 910-493-2086

†Arow Electronics, Inc.
521 Weddell Drive
Sunnyvale 94086
Tel: (408) 745-6600
TWX: 910-339-9371

Arrow Electronics, Inc.
9511 Ridgehaven Court
San Diego 92123
Tel: (619) 565-4800
TWX: 888-064

†Arrow Electronics, Inc.
2961 Dow Avenue
Tustin 92680
Tel: (714) 838-5422
TWX: 910-595-2860

†Avnet Electronics
350 McCormick Avenue
Costa Mesa 92626
Tel: (714) 754-6071
TWX: 910-595-1928

†Hamilton/Avnet Electronics
1175 Bordeaux Drive
Sunnyvale 94086
Tel: (408) 743-3300
TWX: 910-339-9332

†Hamilton/Avnet Electronics
4545 Ridgeview Avenue
San Diego 92123
Tel: (619) 571-7500
TWX: 910-595-2638

†Hamilton/Avnet Electronics
9650 Desoto Avenue
Chatsworth 91311
Tel: (818) 700-1161

†Hamilton Electro Sales
10950 W. Washington Blvd.
Culver City 20230
Tel: (213) 558-2458
TWX: 910-340-6364

Hamilton Electro Sales
1361B West 190th Street
Gardena 90248
Tel: (213) 217-6700

†Hamilton/Avnet Electronics
3002 'G' Street
Ontario 91761
Tel: (714) 989-9411

†Avnet Electronics
20501 Plummer
Chatsworth 91351
Tel: (213) 700-6271
TWX: 910-494-2207

CALIFORNIA (Cont'd.)

†Hamilton Electro Sales
3170 Pullman Street
Costa Mesa 92626
Tel: (714) 641-4150
TWX: 910-595-2638

†Hamilton/Avnet Electronics
4103 Northgate Blvd.
Sacramento 95834
Tel: (916) 920-3150

Wyle Distribution Group
124 Maryland Street
El Segundo 90254
Tel: (213) 322-8100

Wyle Distribution Group
7382 Lampson Ave.
Garden Grove 92641
Tel: (714) 891-1717
TWX: 910-348-7140 or 7111

Wyle Distribution Group
11151 Sun Center Drive
Rancho Cordova 95670
Tel: (916) 638-5282

†Wyle Distribution Group
9525 Chesapeake Drive
San Diego 92123
Tel: (619) 565-9171
TWX: 910-335-1590

†Wyle Distribution Group
3000 Bowers Avenue
Santa Clara 95051
Tel: (408) 727-2500
TWX: 910-338-0296

†Wyle Distribution Group
17872 Cowan Avenue
Irvine 92714
Tel: (714) 863-9953
TWX: 910-595-1572

Wyle Distribution Group
26677 W. Agoura Rd.
Calabasas 91302
Tel: (818) 880-9000
TWX: 372-0232

COLORADO

Arrow Electronics, Inc.
7060 South Tucson Way
Englewood 80112
Tel: (303) 790-4444

†Hamilton/Avnet Electronics
8765 E. Orchard Road
Suite 708
Englewood 80111
Tel: (303) 740-1017
TWX: 910-935-0787

†Wyle Distribution Group
451 E. 124th Avenue
Thornton 80241
Tel: (303) 457-9953
TWX: 910-936-0770

CONNECTICUT

†Arrow Electronics, Inc.
12 Beaumont Road
Wallingford 06492
Tel: (203) 265-7741
TWX: 710-476-0162

Hamilton/Avnet Electronics
Commerce Industrial Park
Commerce Drive
Danbury 06810
Tel: (203) 797-2800
TWX: 710-456-9974

†Pioneer Electronics
112 Main Street
Norwalk 06851
Tel: (203) 853-1515
TWX: 710-468-3373

FLORIDA

†Arrow Electronics, Inc.
400 Fairway Drive
Suite 102
Deerfield Beach 33441
Tel: (305) 429-8200
TWX: 510-955-9456

Arrow Electronics, Inc.
37 Skyline Drive
Suite 3101
Lake Marv 32746
Tel: (407) 323-0252
TWX: 510-959-6337

†Hamilton/Avnet Electronics
6801 N.W. 15th Way
Ft. Lauderdale 33309
Tel: (305) 971-2900
TWX: 510-956-3097

†Hamilton/Avnet Electronics
3197 Tech Drive North
St. Petersburg 33702
Tel: (813) 576-3930
TWX: 810-863-0374

FLORIDA (Cont'd.)

†Hamilton/Avnet Electronics
6947 University Boulevard
Winter Park 32792
Tel: (305) 628-3888
TWX: 810-853-0322

†Pioneer/Technologies Group, Inc.
337 S. Lake Blvd.
Alta Monte Springs 32701
Tel: (407) 834-9090
TWX: 810-853-0284

Pioneer/Technologies Group, Inc.
674 S. Military Trail
Deerfield Beach 33442
Tel: (305) 428-8877
TWX: 510-955-9653

GEORGIA

†Arrow Electronics, Inc.
3155 Northwoods Parkway
Suite A
Norcross 30071
Tel: (404) 449-8252
TWX: 810-766-0439

†Hamilton/Avnet Electronics
5825 D Peachtree Corners
Norcross 30092
Tel: (404) 447-7500
TWX: 810-766-0432

Pioneer/Technologies Group, Inc.
3100 F Northwoods Place
Norcross 30071
Tel: (404) 448-1711
TWX: 810-766-4515

ILLINOIS

Arrow Electronics, Inc.
1140 W. Thorndale
Itasca 60143
Tel: (312) 250-0500
TWX: 312-250-0916

†Hamilton/Avnet Electronics
1130 Thorndale Avenue
Bensenville 60106
Tel: (312) 860-7780
TWX: 910-227-0060

MTI Systems Sales
1100 W. Thorndale
Itasca 60143
Tel: (312) 773-2300

†Pioneer Electronics
1551 Carmen Drive
Elk Grove Village 60007
Tel: (312) 437-9680
TWX: 910-222-1834

INDIANA

†Arrow Electronics, Inc.
2495 Directors Row, Suite H
Indianapolis 46241
Tel: (317) 243-9353
TWX: 810-341-3119

Hamilton/Avnet Electronics
485 Gradle Drive
Carmel 46032
Tel: (317) 844-9333
TWX: 810-260-3966

†Pioneer Electronics
6408 Castleplace Drive
Indianapolis 46250
Tel: (317) 849-7300
TWX: 810-260-1794

IOWA

Hamilton/Avnet Electronics
915 33rd Avenue, S.W.
Cedar Rapids 52404
Tel: (319) 362-4757

KANSAS

Arrow Electronics
8208 Melrose Dr., Suite 210
Lenexa 66214
Tel: (913) 541-9542

†Hamilton/Avnet Electronics
9219 Quivera Road
Overland Park 66215
Tel: (913) 888-8900
TWX: 910-743-0005

Pioneer/Tec Gr.
10551 Lockman Rd.
Lenexa 66215
Tel: (913) 492-0500

KENTUCKY

Hamilton/Avnet Electronics
1051 D. Newton Park
Lexington 40511
Tel: (606) 259-1475

MARYLAND

Arrow Electronics, Inc.
8300 Guilford Drive
Suite H, River Center
Columbia 21046
Tel: (301) 995-0003
TWX: 710-236-9005

Hamilton/Avnet Electronics
6822 Oak Hall Lane
Columbia 21045
Tel: (301) 995-3500
TWX: 710-862-1861

†Mesa Technology Corp.
9720 Patuxent Woods Dr.
Columbia 21046
Tel: (301) 290-8150
TWX: 710-828-9702

†Pioneer/Technologies Group, Inc.
9100 Gaither Road
Gaithersburg 20877
Tel: (301) 921-0660
TWX: 710-828-0545

MASSACHUSETTS

Arrow Electronics, Inc.
25 Upton Dr.
Wilmington 01887
Tel: (617) 935-5134

†Hamilton/Avnet Electronics
10D Centennial Drive
Peabody 01960
Tel: (617) 531-7430
TWX: 710-393-0382

MTI Systems Sales
83 Cambridge St.
Burlington 01813

Pioneer Electronics
44 Hartwell Avenue
Lexington 02173
Tel: (617) 861-9200
TWX: 710-326-6617

MICHIGAN

Arrow Electronics, Inc.
755 Phoenix Drive
Ann Arbor 48104
Tel: (313) 971-8220
TWX: 810-223-6020

Hamilton/Avnet Electronics
2215 29th Street S.E.
Space A5
Grand Rapids 49508
Tel: (616) 243-8805
TWX: 810-274-6921

Pioneer Electronics
4504 Broadmoor S.E.
Grand Rapids 49508
FAX: 616-698-1831

†Hamilton/Avnet Electronics
32487 Schoolcraft Road
Livonia 48150
Tel: (313) 522-4700
TWX: 810-282-8775

†Pioneer/Michigan
13485 Stamford
Livonia 48150
Tel: (313) 525-1800
TWX: 810-242-3271

MINNESOTA

†Arrow Electronics, Inc.
5230 W. 73rd Street
Edina 55435
Tel: (612) 830-1800
TWX: 910-576-3125

†Hamilton/Avnet Electronics
12400 Whitewater Drive
Minnetonka 55434
Tel: (612) 932-0600

†Pioneer Electronics
7625 Golden Triange Dr.
Suite G
Eden Prairi 55343
Tel: (612) 944-3355

MISSOURI

†Arrow Electronics, Inc.
2380 Schuetz
St. Louis 63141
Tel: (314) 567-6888
TWX: 910-764-0882

†Hamilton/Avnet Electronics
13743 Shoreline Court
Earth City 63045
Tel: (314) 344-1200
TWX: 910-762-0684

NEW HAMPSHIRE

†Arrow Electronics, Inc.
3 Perimeter Road
Manchester 03103
Tel: (603) 668-6968
TWX: 710-220-1684

†Hamilton/Avnet Electronics
444 E. Industrial Drive
Manchester 03103
Tel: (603) 624-9400

NEW JERSEY

†Arrow Electronics, Inc.
Four East Stow Road
Unit 11
Marlton 08053
Tel: (609) 596-8000
TWX: 710-897-0829

†Arrow Electronics
6 Century Drive
Parsipanny 07054
Tel: (201) 538-0900

†Hamilton/Avnet Electronics
1 Keystone Ave., Bldg. 36
Cherry Hill 08003
Tel: (609) 424-0110
TWX: 710-940-0262

†Hamilton/Avnet Electronics
10 Industrial
Fairfield 07006
Tel: (201) 575-5300
TWX: 710-734-4388

†MTI Systems Sales
37 Kulick Rd.
Fairfield 07006
Tel: (201) 227-5552

†Pioneer Electronics
45 Route 46
Pinebrook 07058
Tel: (201) 575-3510
TWX: 710-734-4382

NEW MEXICO

Alliance Electronics Inc.
11030 Cochiti S.E.
Albuquerque 87123
Tel: (505) 292-3360
TWX: 910-989-1151

Hamilton/Avnet Electronics
2524 Baylor Drive S.E.
Albuquerque 87106
Tel: (505) 765-1500
TWX: 910-989-0614

NEW YORK

†Arrow Electronics, Inc.
3375 Brighton Henrietta Townline Rd.
Rochester 14623
Tel: (716) 275-0300
TWX: 510-253-4766

Arrow Electronics, Inc.
20 Oser Avenue
Hauppauge 11788
Tel: (516) 231-1000
TWX: 510-227-6623

Hamilton/Avnet
933 Motor Parkway
Hauppauge 11788
Tel: (516) 231-9800
TWX: 510-224-6166

†Hamilton/Avnet Electronics
333 Metro Park
Rochester 14623
Tel: (716) 475-9130
TWX: 510-253-5470

†Hamilton/Avnet Electronics
103 Twin Oaks Drive
Syracuse 13206
Tel: (315) 437-0288
TWX: 710-541-1560

†MTI Systems Sales
38 Harbor Park Drive
Port Washington 11050
Tel: (516) 621-6200

†Pioneer Electronics
68 Corporate Drive
Binghamton 13904
Tel: (607) 722-9300
TWX: 510-252-0893

Pioneer Electronics
40 Oser Avenue
Hauppauge 11787
Tel: (516) 231-9200

†Microcomputer System Technical Distributor Center

DOMESTIC DISTRIBUTORS (Cont'd.)

NEW YORK (Cont'd.)

†Pioneer Electronics
60 Crossway Park West
Woodbury, Long Island 11797
Tel: (516) 921-8700
TWX: 510-221-2184

†Pioneer Electronics
840 Fairport Park
Fairport 14450
Tel: (716) 381-7070
TWX: 510-253-7001

NORTH CAROLINA

†Arrow Electronics, Inc.
5240 Greensdairy Road
Raleigh 27604
Tel: (919) 876-3132
TWX: 510-928-1856

†Hamilton/Avnet Electronics
3510 Spring Forest Drive
Raleigh 27604
Tel: (919) 878-0819
TWX: 510-928-1836

Pioneer/Technologies Group, Inc.
9801 A-Southern Pine Blvd.
Charlotte 28210
Tel: (919) 527-8188
TWX: 810-621-0366

OHIO

Arrow Electronics, Inc.
7620 McEwen Road
Centerville 45459
Tel: (513) 435-5563
TWX: 810-459-1611

†Arrow Electronics, Inc.
6238 Cochran Road
Solon 44139
Tel: (216) 248-3990
TWX: 810-427-9409

†Hamilton/Avnet Electronics
954 Senate Drive
Dayton 45459
Tel: (513) 439-6733
TWX: 810-450-2531

Hamilton/Avnet Electronics
4588 Emery Industrial Pkwy.
Warrensville Heights 44128
Tel: (216) 349-5100
TWX: 810-427-9452

†Hamilton/Avnet Electronics
777 Brooksedge Blvd.
Westerville 43081
Tel: (614) 882-7004

†Pioneer Electronics
4433 Interpoint Boulevard
Dayton 45424
Tel: (513) 236-9900
TWX: 810-459-1622

†Pioneer Electronics
4800 E. 131st Street
Cleveland 44105
Tel: (216) 587-3600
TWX: 810-422-2211

OKLAHOMA

Arrow Electronics, Inc.
1211 E. 51st Street
Suite 101
Tulsa 74146
Tel: (918) 252-7537

†Hamilton/Avnet Electronics
12121 E. 51st St., Suite 102A
Tulsa 74146
Tel: (918) 252-7297

OREGON

†Almac Electronics Corp.
1885 N.W. 169th Place
Beaverton 97005
Tel: (503) 629-8090
TWX: 910-467-8746

†Hamilton/Avnet Electronics
6024 S.W. Jean Road
Bldg. C, Suite 10
Lake Oswego 97034
Tel: (503) 635-7848
TWX: 910-455-8179

Wyle Distribution Group
5250 N.E. Elam Young Parkway
Suite 600
Hillsboro 97124
Tel: (503) 640-6000
TWX: 910-460-2203

PENNSYLVANIA

Arrow Electronics, Inc.
650 Seco Road
Monroeville 15146
Tel: (412) 856-7000

Hamilton/Avnet Electronics
2800 Liberty Ave.
Pittsburgh 15238
Tel: (412) 281-4150

Pioneer Electronics
259 Kappa Drive
Pittsburgh 15238
Tel: (412) 782-2300
TWX: 710-795-3122

†Pioneer/Technologies Group, Inc.
Delaware Valley
261 Gibralter Road
Horsham 19044
Tel: (215) 674-4000
TWX: 510-665-6778

TEXAS

†Arrow Electronics, Inc.
3220 Commander Drive
Carrollton 75006
Tel: (214) 380-6464
TWX: 910-860-5377

†Arrow Electronics, Inc.
10899 Kinghurst
Suite 100
Houston 77099
Tel: (713) 530-4700
TWX: 910-880-4439

†Arrow Electronics, Inc.
2227 W. Braker Lane
Austin 78758
Tel: (512) 835-4180
TWX: 910-874-1348

†Hamilton/Avnet Electronics
1807 W. Braker Lane
Austin 78758
Tel: (512) 837-8911
TWX: 910-874-1319

TEXAS (Cont'd.)

†Hamilton/Avnet Electronics
2111 W. Walnut Hill Lane
Irving 75038
Tel: (214) 550-6111
TWX: 910-860-5929

†Hamilton/Avnet Electronics
4850 Wright Rd., Suite 190
Stafford 77477
Tel: (713) 240-7733
TWX: 910-881-5523

†Pioneer Electronics
18260 Kramer
Austin 78758
Tel: (512) 835-4000
TWX: 910-874-1323

†Pioneer Electronics
13710 Omega Road
Dallas 75234
Tel: (214) 386-7300
TWX: 910-850-5563

†Pioneer Electronics
5853 Point West Drive
Houston 77036
Tel: (713) 988-5555
TWX: 910-881-1606

Wyle Distribution Group
1810 Greenville Avenue
Richardson 75081
Tel: (214) 235-9953

UTAH

Arrow Electronics
1946 Parkway Blvd.
Salt Lake City 84119
Tel: (801) 973-6913

†Hamilton/Avnet Electronics
1585 West 2100 South
Salt Lake City 84119
Tel: (801) 972-2800
TWX: 910-925-4018

Wyle Distribution Group
1325 West 2200 South
Suite E
West Valley 84119
Tel: (801) 974-9953

WASHINGTON

†Almac Electronics Corp.
14360 S.E. Eastgate Way
Bellevue 98007
Tel: (206) 643-9992
TWX: 910-444-2067

Arrow Electronics, Inc.
19540 68th Ave. South
Kent 98032
Tel: (206) 575-4420

†Hamilton/Avnet Electronics
14212 N.E. 21st Street
Bellevue 98005
Tel: (206) 643-3950
TWX: 910-443-2469

Wyle Distribution Group
15385 N.E. 90th Street
Redmond 98052
Tel: (206) 881-1150

WISCONSIN

Arrow Electronics, Inc.
200 N. Patrick Blvd., Ste. 100
Brookfield 53005
Tel: (414) 767-6600
TWX: 910-262-1193

Hamilton/Avnet Electronics
2975 Moorland Road
New Berlin 53151
Tel: (414) 784-4510
TWX: 910-262-1182

CANADA

ALBERTA

Hamilton/Avnet Electronics
2816 21st Street N.E.
Calgary T2E 6Z3
Tel: (403) 230-3586
TWX: 03-827-642

Zentronics
Bay No. 1
3300 14th Avenue N.E.
Calgary T2A 6J4
Tel: (403) 272-1021

BRITISH COLUMBIA

†Hamilton/Avnet Electronics
105-2550 Boundary
Burmalay V5M 3Z3
Tel: (604) 437-6667

Zentronics
108-11400 Bridgeport Road
Richmond V6X 1T2
Tel: (604) 273-5575
TWX: 04-5077-89

MANITOBA

Zentronics
60-1313 Border Unit 60
Winnipeg R3H 0X4
Tel: (204) 694-1957

ONTARIO

Arrow Electronics, Inc.
36 Antares Dr.
Nepean K2E 7W5
Tel: (613) 226-6903

Arrow Electronics, Inc.
1093 Meyerside
Mississauga L5T 1M4
Tel: (416) 673-7769
TWX: 06-218213

†Hamilton/Avnet Electronics
6845 Rexwood Road
Units 3-4-5
Mississauga L4T 1R2
Tel: (416) 677-7432
TWX: 610-492-8867

Hamilton/Avnet Electronics
6845 Rexwood Road
Unit 6
Mississauga L4T 1R2
Tel: (416) 277-0484

ONTARIO (Cont'd.)

†Hamilton/Avnet Electronics
190 Colonnade Road South
Nepean K2E 7L5
Tel: (613) 226-1700
TWX: 05-349-71

†Zentronics
8 Tilbury Court
Brampton L6T 3T4
Tel: (416) 451-9600
TWX: 06-976-78

†Zentronics
155 Colonnade Road
Unit 17
Nepean K2E 7K1
Tel: (613) 226-8840

Zentronics
60-1313 Border St.
Winnipeg R3H 0I4
Tel: (204) 694-7957

QUEBEC

†Arrow Electronics Inc.
4050 Jean Talon Quest
Montreal H4P 1W1
Tel: (514) 735-5511
TWX: 05-25590

Arrow Electronics, Inc.
909 Charest Blvd.
Quebec J1N 2C9
Tel: (418) 687-4231
TWX: 05-13388

Hamilton/Avnet Electronics
2795 Halpern
St. Laurent H2E 7K1
Tel: (514) 335-1000
TWX: 610-421-3731

Zentronics
817 McCaffrey
St. Laurent H4T 1M3
Tel: (514) 737-9700
TWX: 05-827-535

†Microcomputer System Technical Distributor Center

EUROPEAN SALES OFFICES

DENMARK

Intel
Glentevej 61, 3rd Floor
2400 Copenhagen NV
Tel: (45) (01) 19 80 33
TLX: 19567

FINLAND

Intel
Ruosilantie 2
00390 Helsinki
Tel: (358) 0 544 644
TLX: 123332

FRANCE

Intel
1, Rue Edison-BP 303
78054 St. Quentin-en-Yvelines Cedex
Tel: (33) (1) 30 57 70 00
TLX: 699016

Intel
4, Quai des Etroits
69321 Lyon Cedex 05
Tel: (33) (16) 78 42 40 89
TLX: 305153

WEST GERMANY

Intel*
Dornacher Strasse 1
8016 Feldkirchen bei Muenchen
Tel: (49) 089/90992-0
TLX: 5-23177
FAX: 904-3948

Intel
Hohenzollern Strasse 5
3000 Hannover 1
Tel: (49) 0511/344081
TLX: 9-23625

Intel
Abraham Lincoln Strasse 16-18
6200 Wiesbaden
Tel: (49) 06121/7605-0
TLX: 4-186183

Intel
Zettachring 10A
7000 Stuttgart 80
Tel: (49) 0711/728728-0
TLX: 7-254826

ISRAEL

Intel*
Atidim Industrial Park-Neve Sharet
P.O. Box 43202
Tel-Aviv 61430
Tel: (972) 03-498080
TLX: 371215

ITALY

Intel*
Milanofiori Palazzo E
20090 Assago
Milano
Tel: (39) (02) 824 40 71
TLX: 341286

NETHERLANDS

Intel*
Marten Meesweg 93
3068 AV Rotterdam
Tel: (31) 10.407.11.11
TLX: 22283

NORWAY

Intel
Hvamveien 4-PO Box 92
2013 Skjetten
Tel: (47) (6) 842 420
TLX: 78018

SPAIN

Intel
Zurbaran, 28
28010 Madrid
Tel: (34) 410 40 04
TLX: 46880

SWEDEN

Intel*
Dalvagen 24
171 36 Solna
Tel: (46) 8 734 01 00
TLX: 12261

SWITZERLAND

Intel
Zuerichstrasse
8185 Winkel-Rueti bei Zuerich
Tel: (41) 01/860 62 62
TLX: 825977

UNITED KINGDOM

Intel*
Pipers Way
Swindon, Wiltshire SN3 1RJ
Tel: (44) (0793) 696000
TLX: 444447/8

EUROPEAN DISTRIBUTORS/REPRESENTATIVES

AUSTRIA

Bacher Electronics G.m.b.H.
Rotenmuehlgasse 26
1120 Wien
Tel: (43) (0222) 83 56 46-0
TLX: 131532

BELGIUM

Inelco Belgium S.A.
Av. des Croix de Guerre 94
1120 Bruxelles
Oorlogskruisenlaan, 94
1120 Brussel
Tel: (32) (02) 216 01 60
TLX: 64475

DENMARK

ITT-Multikomponent
Naverland 29
2600 Glostrup
Tel: (45) (0) 2 45 66 45
TLX: 33 355

FINLAND

OY Fintronic AB
Melkonkatu 24A
00210 Helsinki
Tel: (358) (0) 6926022
TLX: 124224

FRANCE

Generim
Z.A. de Courtaboeuf
Av. de la Baltique-BP 88
91943 Les Ulis Cedex
Tel: (33) (1) 69 07 78 78
TLX: 691700

Jermyn
73-79, rue des Solets
Silic 585
94663 Rungis Cedex
Tel: (33) (1) 45 60 04 00
TLX: 260967

Metrologie
Tour d'Asnieres
4, av. Laurent-Cely
92606 Asnieres Cedex
Tel: (33) (1) 47 90 62 40
TLX: 611448

Tekelec-Airtronic
Rue Carle Vernet - BP 2
92315 Sevres Cedex
Tel: (33) (1) 45 34 75 35
TLX: 204552

WEST GERMANY

Electronic 2000 AG
Stahlgruberring 12
8000 Muenchen 82
Tel: (49) 089/42001-0
TLX: 522561

ITT Multikomponent GmbH
Postfach 1265
Bahnhofstrasse 44
7141 Moeglingen
Tel: (49) 07141/4879
TLX: 7264472

Jermyn GmbH
Im Dachsstueck 9
6250 Limburg
Tel: (49) 06431/508-0
TLX: 415257-0

Metrologie GmbH
Meglingerstrasse 49
8000 Muenchen 71
Tel: (49) 089/78042-0
TLX: 5213189

Proelectron Vertriebs GmbH
Max Planck Strasse 1-3
6072 Dreieich
Tel: (49) 06103/3040
TLX: 417903

IRELAND

Micro Marketing Ltd.
Glenageary Office Park
Glenageary
Co. Dublin
Tel: (21) (353) (01) 85 63 25
TLX: 31584

ISRAEL

Eastronics Ltd.
11 Rozanis Street
P.O.B. 39300
Tel-Aviv 61392
Tel: (972) 03-475151
TLX: 33638

ITALY

Intesi
Divisione ITT Industries GmbH
Viale Milanofiori
Palazzo E/5
20090 Assago
Milano
Tel: (39) 02/824701
TLX: 311351

Lasi Elettronica S.p.A.
V. le Fulvio Testi, 126
20092 Cinisello Balsamo
Milano
Tel: (39) 02/2440012
TLX: 352040

NETHERLANDS

Koning en Hartman
1 Energieweg
2627 AP Delft
Tel: (31) 15609906
TLX: 38250

NORWAY

Nordisk Elektronikk (Norge) A/S
Postboks 123
Smedsvingen 4
1364 Hvalstad
Tel: (47) (02) 84 62 10
TLX: 77546

PORTUGAL

Ditram
Avenida Marques de Tomar, 46-A
1000 Lisboa
Tel: (351) (1) 73 48 34
TLX: 14182

SPAIN

ATD Electronica, S.A.
Plaza Ciudad de Viena, 6
28040 Madrid
Tel: (34) 234 40 00
TLX: 42754

ITT-SESA
Calle Miguel Angel, 21-3
28010 Madrid
Tel: (34) 419 09 57
TLX: 27461

SWEDEN

Nordisk Elektronik AB
Huvudstagatan 1
Box 1409
171 27 Solna
Tel: (46) 08-734 97 70
TLX: 105 47

SWITZERLAND

Industrade A.G.
Hertistrasse 31
8304 Wallisellen
Tel: (41) (801) 83 05 04 0
TLX: 56788

TURKEY

EMPA Electronic
Lindwurmstrasse 95A
8000 Muenchen 2
Tel: (49) 089/53 80 570
TLX: 528573

UNITED KINGDOM

Accent Electronic Components Ltd.
Jubilee House, Jubilee Road
Letchworth, Herts SG6 1TL
Tel: (44) (0462) 686666
TLX: 826293

Bytech-Comway Systems
3 The Western Centre
Western Road
Bracknell RG12 1RW
Tel: (44) (0344) 55333
TLX: 847201

Jermyn
Vestry Estate
Otford Road
Sevenoaks
Kent TN14 5EU
Tel: (44) (0732) 450144
TLX: 95142

MMD
Unit 8 Southview Park
Caversham
Reading
Berkshire RG4 0AF
Tel: (44) (0734) 48 16 66
TLX: 846669

Rapid Silicon
Rapid House
Denmark Street
High Wycombe
Buckinghamshire HP11 2ER
Tel: (44) (0494) 442266
TLX: 837931

Rapid Systems
Rapid House
Denmark Street
High Wycombe
Buckinghamshire HP11 2ER
Tel: (44) (0494) 450244
TLX: 837931

YUGOSLAVIA

Rapido Electronic Components S.p.a.
Via C. Beccaria, 8
34133 Trieste
Italia
Tel: (39) 040/360555
TLX: 460461

*Field Application Location

intel®

INTERNATIONAL SALES OFFICES

AUSTRALIA

Intel Australia Pty. Ltd.*
Spectrum Building
200 Pacific Hwy., Level 6
Crows Nest, NSW, 2065
Tel: (2) 957-2744
TLX: AA 20097
FAX: (2) 923-2632

BRAZIL

Intel Semicondutores do Brasil LTDA
Av. Paulista, 1159-CJS 404/405
01311 - Sao Paulo - S.P.
Tel: 55-11-287-5899
TLX: 1153146 SAPI BR
FAX: 55-11-212-7631

CHINA/HONG KONG

Intel PRC Corporation
15/F, Office 1, Citic Bldg.
Jian Guo Men Wai Street
Beijing, PRC
Tel: (1) 500-4850
TLX: 22947 INTEL CN
FAX: (1) 500-2953

Intel Semiconductor Ltd.*
10/F East Tower
Bond Center
Queensway, Central
Hong Kong
Tel: (5) 8444-555
TLX: 63869 ISHLHK HX
FAX: (5) 8681-989

JAPAN

Intel Japan K.K.
5-6 Tokodai, Tsukuba-shi
Ibaraki, 300-26
Tel: 029747-8511
TLX: 3656-160
FAX: 029747-8450

Intel Japan K.K.*
Daiichi Mitsugi Bldg.
1-8889 Fuchu-cho
Fuchu-shi, Tokyo 183
Tel: 0423-60-7871
FAX: 0423-60-0315

Intel Japan K.K.*
Flower-Hill Shin-machi Bldg.
1-23-9 Shinmachi
Setagaya-ku, Tokyo 154
Tel: 03-426-2231
FAX: 03-427-7620

Intel Japan K.K.*
Bldg. Kumagaya
2-69 Hon-cho
Kumagaya-shi, Saitama 360
Tel: 0485-24-6871
FAX: 0485-24-7518

Intel Japan K.K.*
Mitsui-Seimei Musashi-kosugi Bldg.
915 Shinmaruko, Nakahara-ku
Kawasaki-shi, Kanagawa 211
Tel: 044-733-7011
FAX: 044-733-7010

JAPAN (Cont'd.)

Intel Japan K.K.
Nihon Seimei Atsugi Bldg.
1-2-1 Asahi-machi
Atsugi-shi, Kanagawa 243
Tel: 0462-29-3731
FAX: 0462-29-3781

Intel Japan K.K.*
Ryokuchi-Eki Bldg.
2-4-1 Terauchi
Toyonaka-shi, Osaka 560
Tel: 06-863-1091
FAX: 06-863-1084

Intel Japan K.K.
Shinmaru Bldg.
1-5-1 Marunouchi
Chiyoda-ku, Tokyo 100
Tel: 03-201-3621
FAX: 03-201-6850

Intel Japan K.K.
Green Bldg.
1-16-20 Nishiki
Naka-ku, Nagoya-shi
Aichi 450
Tel: 052-204-1261
FAX: 052-204-1285

KOREA

Intel Technology Asia, Ltd.
Business Center 16th Floor
61, Yoido-Dong, Young Deung Po-Ku
Seoul 150
Tel: (2) 784-8186, 8286, 8386
TLX: K29312 INTELKO
FAX: (2) 784-8096

SINGAPORE

Intel Singapore Technology, Ltd.
101 Thomson Road #21-06
Goldhill Square
Singapore 1130
Tel: 250-7811
TLX: 39921 INTEL
FAX: 250-9256

TAIWAN

Intel Technology (Far East) Ltd.
Taiwan Branch
10/F, No. 205, Tun Hua N. Road
Taipei, R.O.C.
Tel: 886-2-716-9660
TLX: 13159 INTELTWN
FAX: 886-2-717-2455

INTERNATIONAL DISTRIBUTORS/REPRESENTATIVES

ARGENTINA

DAFSYS S.R.L.
Chacabuco, 90-4 PISO
1069-Buenos Aires
Tel: 54-1-334-1871
54-1-334-7726
TLX: 25472

AUSTRALIA

Total Electronics
15-17 Hume Street
Huntingdale, 3166,
Victoria, Australia
Tel: 03-544-8244
TLX: AA 30895
FAX: 03-543-8179

BRAZIL

Elebra Microelectronica
R. Geraldo Flausina Gomes, 78
9 Andar
04575 - Sao Paulo - S.P.
Tel: 011-55-11-534-9637
TLX: 3911125131 ELBR BR
FAX: 55-11-534-9424

CHILE

DIN Instruments
Suecia 2323
Casilla 6055, Correo 22
Santiago
Tel: 56-2-225-8139
TLX: 440422 RUDY CZ

CHINA/HONG KONG

Novel Precision Machinery Co., Ltd.
Flat D, 20 Kingsford Ind. Bldg.
Phase 1, 26 Kwai Hei Street
N.T., Kowloon
Hong Kong
Tel: 852-0-223-222
TWX: 39114 JINMI HX
FAX: 852-0-261-602

INDIA

Micronic Devices
Arun Complex
No. 65 D.V.G. Road
Basavanagudi
Bangalore 560 004
Tel: 91-812-600-631
011-91-812-621-455
TLX: 0845-8332 MD BG IN

Micronic Devices
Flat 403, Gagan Deep
12, Rajendra Place
New Delhi 110 008
Tel: 91-58-97-71
011-91-57-23509
TLX: 03163235 MDND IN

Micronic Devices
No. 516 5th Floor
Swastik Chambers
Sion, Trombay Road
Chembur
Bombay 400 071
Tel: 91-52-39-63
TLX: 9531 171447 MDEV IN

S&S Corporation
Camden Business Center
Suite 6
1610 Blossom Hill Rd.
San Jose, CA 95124
U.S.A.
Tel: (408) 978-6216
TLX: 820281

JAPAN

Asahi Electronics Co. Ltd.
KMM Bldg. 2-14-1 Asano
Kokurakita-ku
Kitakyushu-shi 802
Tel: 093-511-6471
FAX: 093-551-7861

C. Itoh Techno-Science Co., Ltd.
4-8-1 Dobashi, Miyamae-ku
Kawasaki-shi, Kanagawa 213
Tel: 044-852-5121
FAX: 044-877-4268

JAPAN (Cont'd.)

Dia Semicon Systems, Inc.
Wacore 64, 1-37-8 Sangenjaya
Setagaya-ku, Tokyo 154
Tel: 03-487-0386
FAX: 03-487-8088

Okaya Koki
2-4-18 Sakae
Naka-ku, Nagoya-shi 460
Tel: 052-204-2916
FAX: 052-204-2901

Ryoyo Electro Corp.
Konwa Bldg.
1-12-22 Tsukiji
Chuo-ku, Tokyo 104
Tel: 03-546-5011
FAX: 03-546-5044

KOREA

J-Tek Corporation
6th Floor, Government Pension Bldg.
24-3 Yoido-Dong
Youngdeungpo-ku
Seoul 150
Tel: 82-2-782-8039
TLX: 25299 KODIGIT
FAX: 82-2-784-8391

Samsung Semiconductor &
Telecommunications Co., Ltd.
150, 2-KA, Tafpyung-ro, Chung-ku
Seoul 100
Tel: 82-2-751-3987
TLX: 27970 KORSST
FAX: 82-2-753-0967

MEXICO

Dicopel S.A.
Tochtli 368 Fracc. Ind. San Antonio
Azcapotzalco
C.P. 02760-Mexico, D.F.
Tel: 52-5-561-3211
TLX: 1773790 DICOME

NEW ZEALAND

Switch Enterprises
36 Olive Road
Penrose, Auckland
ATTN: Dean Danford
Tel: 64-9-591155
FAX: 64-9-592681

SINGAPORE

Electronic Resources Pte, Ltd.
17 Harvey Road #04-01
Singapore 1336
Tel: 283-0888, 289-1618
TWX: 56541 FRELS
FAX: 2895327

SOUTH AFRICA

Electronic Building Elements, Pty. Ltd.
P.O. Box 4609
Pine Square, 18th Street
Hazelwood, Pretoria 0001
Tel: 27-12-469921
TLX: 3-227786 SA
FAX: 0927-012-46-9221

TAIWAN

Micro Electronics Corporation
No. 585, Ming Shen East Rd.
Taipei, R.O.C.
Tel: 886-2-501-8231
FAX: 886-2-501-4265

Sertek
5FL, 135 Sec. 2
Chien-Kuo N. Rd.
Taipei 10479
R.O.C.
Tel: (02) 5010055
FAX: (02) 5012521
(02) 5058414

VENEZUELA

P. Benavides S.A.
Avilanes a Rio
Residencia Kamarata
Locales 4 A17
La Candelaria, Caracas
Tel: 58-2-571-0396
TLX: 28450 PBVEN VC
FAX: 58-2-572-3321

*Field Application Location

DOMESTIC SERVICE OFFICES

ALABAMA

Intel Corp.
5015 Bradford Dr., #2
Huntsville 35805
Tel: (205) 830-4010

ARIZONA

Intel Corp.
11225 N. 28th Dr.
Suite D-214
Phoenix 85029
Tel: (602) 869-4980

Intel Corp.
500 E. Fry Blvd., Suite M-15
Sierra Vista 85635
Tel: (602) 459-5010

Intel Corp.
1161 N. El Dorado Place
Suite 301
Tucson 85715
Tel: (602) 299-6815

CALIFORNIA

Intel Corp.
21515 Vanowen Street
Suite 116
Canoga Park 91303
Tel: (818) 704-8500

Intel Corp.
2250 E. Imperial Highway
Suite 218
El Segundo 90245
Tel: (213) 640-6040

Intel Corp.
1900 Prairie City Rd.
Folsom 95630-9597
Tel: (916) 351-6143

Intel Corp.
1510 Arden Way, Suite 101
Sacramento 95815
Tel: (916) 920-8096

Intel Corp.
4350 Executive Drive
Suite 105
San Diego 92121
Tel: (619) 452-5880

Intel Corp.*
400 N. Tustin Avenue
Suite 450
Santa Ana 92705
Tel: (714) 835-9642
TWX: 910-595-1114

Intel Corp.*
San Tomas 4
2700 San Tomas Expressway
2nd Floor
Santa Clara 95051
Tel: (408) 986-8086
TWX: 910-338-0255

COLORADO

Intel Corp.
4445 Northpark Drive
Suite 100
Colorado Springs 80907
Tel: (303) 594-6622

Intel Corp.*
650 S. Cherry St., Suite 915
Denver 80222
Tel: (303) 321-8086
TWX: 910-931-2289

CONNECTICUT

Intel Corp.
26 Mill Plain Road
2nd Floor
Danbury 06811
Tel: (203) 748-3130
TWX: 710-456-1199

FLORIDA

Intel Corp.
6363 N.W. 6th Way
Suite 100
Ft. Lauderdale 33309
Tel: (305) 771-0600
TWX: 510-956-9407
FAX: 305-772-8193

Intel Corp.
5850 T.G. Lee Blvd.
Suite 340
Orlando 32822
Tel: (305) 240-8000
FAX: 305-240-8097

Intel Corp.
11300 4th Street North
Suite 170
St. Petersburg 33716
Tel: (813) 577-2413
FAX: 813-578-1607

GEORGIA

Intel Corp.
3280 Pointe Parkway
Suite 200
Norcross 30092
Tel: (404) 449-0541

ILLINOIS

Intel Corp.*
300 N. Martingale Road
Suite 400
Schaumburg 60173
Tel: (312) 310-8031

INDIANA

Intel Corp.
8777 Purdue Road
Suite 125
Indianapolis 46268
Tel: (317) 875-0623

IOWA

Intel Corp.
1930 St. Andrews Drive N.E.
2nd Floor
Cedar Rapids 52402
Tel: (319) 393-5510

KANSAS

Intel Corp.
8400 W. 110th Street
Suite 170
Overland Park 66210
Tel: (913) 345-2727

MARYLAND

Intel Corp.*
7321 Parkway Drive South
Suite C
Hanover 21076
Tel: (301) 796-7500
TWX: 710-862-1944

Intel Corp.
7833 Walker Drive
Suite 550
Greenbelt 20770
Tel: (301) 441-1020

MASSACHUSETTS

Intel Corp.*
Westford Corp. Center
3 Carlisle Road
2nd Floor
Westford 01886
Tel: (508) 692-3222
TWX: 710-343-6333

MICHIGAN

Intel Corp.
7071 Orchard Lake Road
Suite 100
West Bloomfield 48033
Tel: (313) 851-8096

MINNESOTA

Intel Corp.
3500 W. 80th St., Suite 360
Bloomington 55431
Tel: (612) 835-6722
TWX: 910-576-2867

MISSOURI

Intel Corp.
4203 Earth City Expressway
Suite 131
Earth City 63045
Tel: (314) 291-1990

NEW JERSEY

Intel Corp.
Raritan Plaza III
Raritan Center
Edison 08817
Tel: (201) 225-3000

Intel Corp.
385 Sylvan Avenue
Englewood Cliffs 07632
Tel: (201) 567-0821
TWX: 710-991-8593

Intel Corp.*
Parkway 109 Office Center
328 Newman Springs Road
Red Bank 07701
Tel: (201) 747-2233

†Intel Corp.
280 Corporate Center
75 Livingston Avenue
First Floor
Roseland 07068
Tel: (201) 740-0111
FAX: 201-740-0626

NEW MEXICO

Intel Corp.
8500 Menaul Boulevard N.E.
Suite B 295
Albuquerque 87112
Tel: (505) 292-8086

NEW YORK

Intel Corp.
127 Main Street
Binghamton 13905
Tel: (607) 773-0337
FAX: 607-723-2677

Intel Corp.*
850 Cross Keys Office Park
Fairport 14450
Tel: (716) 425-2750
TWX: 510-253-7391

Intel Corp.*
300 Motor Parkway
Hauppauge 11787
Tel: (516) 231-3300
TWX: 510-227-6236

Intel Corp.
Westage Business Center
Bldg. 300, Route 9
Fishkill 12524
Tel: (914) 897-3860
FAX: 914-897-3125

NORTH CAROLINA

Intel Corp.
5700 Executive Drive
Suite 213
Charlotte 28212
Tel: (704) 568-8966

Intel Corp.
2306 W. Meadowview Road
Suite 206
Greensboro 27407
Tel: (919) 294-1541

Intel Corp.
2700 Wycliff Road
Suite 102
Raleigh 27607
Tel: (919) 781-8022

OHIO

Intel Corp.*
3401 Park Center Drive
Suite 220
Dayton 45414
Tel: (513) 890-5350
TWX: 810-450-2528

Intel Corp.*
25700 Science Park Dr.
Suite 100
Beachwood 44122
Tel: (216) 464-2736
TWX: 810-427-9298

OKLAHOMA

Intel Corp.
6801 N. Broadway
Suite 115
Oklahoma City 73162
Tel: (405) 848-8086

OREGON

Intel Corp.
15254 N.W. Greenbrier Parkway
Building B
Beaverton 97006
Tel: (503) 645-8051
TWX: 910-467-8741

Intel Corp.
5200 N.E. Elam Young Parkway
Hillsboro 97123
Tel: (503) 681-8080

PENNSYLVANIA

Intel Corp.*
455 Pennsylvania Avenue
Suite 230
Fort Washington 19034
Tel: (215) 641-1000
TWX: 510-661-2077

Intel Corp.*
400 Penn Center Blvd.
Suite 610
Pittsburgh 15235
Tel: (412) 823-4970

PUERTO RICO

Intel Microprocessor Corp.
South Industrial Park
P.O. Box 910
Las Piedras 00671
Tel: (809) 733-8616

TEXAS

Intel Corp.
313 E. Anderson Lane
Suite 314
Austin 78752
Tel: (512) 454-3628

TEXAS (Cont'd.)

Intel Corp.*
12000 Ford Road
Suite 400
Dallas 75234
Tel: (214) 241-8087
FAX: 214-484-1180

Intel Corp.*
7322 S.W. Freeway
Suite 1490
Houston 77074
Tel: (713) 988-8086
TWX: 910-881-2490

UTAH

Intel Corp.
428 East 6400 South
Suite 104
Murray 84107
Tel: (801) 263-8051

VIRGINIA

Intel Corp.
1504 Santa Rosa Road
Suite 108
Richmond 23288
Tel: (804) 282-5668

WASHINGTON

Intel Corp.
155 108th Avenue N.E.
Suite 386
Bellevue 98004
Tel: (206) 453-8086
TWX: 910-443-3002

Intel Corp.
408 N. Mullan Road
Suite 102
Spokane 99206
Tel: (509) 928-8086

WISCONSIN

Intel Corp.
330 S. Executive Dr.
Suite 102
Brookfield 53005
Tel: (414) 784-8087
FAX: (414) 796-2115

CANADA

BRITISH COLUMBIA

Intel Semiconductor of Canada, Ltd.
4585 Canada Way, Suite 202
Burnaby V5G 4L6
Tel: (604) 298-0387
FAX: (604) 298-8234

ONTARIO

Intel Semiconductor of Canada, Ltd.
2650 Queensview Drive
Suite 250
Ottawa K2B 8H6
Tel: (613) 829-9714
TLX: 053-4115

Intel Semiconductor of Canada, Ltd.
190 Attwell Drive
Suite 500
Rexdale M9W 6H8
Tel: (416) 675-2105
TLX: 06983574
FAX: (416) 675-2438

QUEBEC

Intel Semiconductor of Canada, Ltd.
620 St. John Boulevard
Pointe Claire H9R 3K2
Tel: (514) 694-9130
TWX: 514-694-9134

CUSTOMER TRAINING CENTERS

CALIFORNIA

2700 San Tomas Expressway
Santa Clara 95051
Tel: (408) 970-1700

ILLINOIS

300 N. Martingale, #300
Schaumburg 60173
Tel: (312) 310-5700

MASSACHUSETTS

3 Carlisle Road
Westford 01886
Tel: (508) 692-1000

MARYLAND

7833 Walker Dr., 4th Floor
Greenbelt 20770
Tel: (301) 220-3380

SYSTEMS ENGINEERING OFFICES

CALIFORNIA

2700 San Tomas Expressway
Santa Clara 95051
Tel: (408) 986-8086

ILLINOIS

300 N. Martingale, #300
Schaumburg 60173
Tel: (312) 310-8031

NEW YORK

300 Motor Parkway
Hauppauge 11788
Tel: (516) 231-3300

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 PM
Microprocessor/Peripheral

ISBN 1-55512-041-5